Int. J. Nonlinear Anal. Appl. 14 (2023) 1, 753–761
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.6874

Evolving trees for detecting android malware using evolutionary learning

Waheed Fadel Waheed, Hasanen Alyasiri*

Faculty of Computer Science and Mathematics, University of Kufa, Iraq

(Communicated by Ehsan Kozegar)

Abstract

Android smartphone platforms have grown rapidly and become ubiquitous due to the fact that they have opened up new possibilities in every aspect of modern life. However, as a consequence of their widespread, such systems suffer an increasing frequency of malware attacks. New malware or modifications to existing malware are increasingly being used by attackers in such operations. Security researchers deal with a continually evolving threat environment that is broad, complicated, and ever-changing. Conventional detection methods failed to adequately protect such systems and so the security industry has started to detect systems that use machine learning techniques are becoming more common. In this work, we used an evolutionary algorithm to identify malware targeting Android smartphones. In order to evaluate our system, we compared it to different state-of-the-art algorithms. Finally, we demonstrated our proposed method's ability to detect zero-day malware. Obtained results show that our method achieved 99.11% detection accuracy, and is well-suited for detecting zero-day malware.

Keywords: Android, smartphone, malware
2020 MSC: 68M25

1 Introduction

In this era, there is no denying that smartphones have become pervasive and bring huge opportunities and promises ease of life. Compared to PCs from a decade ago, modern mobile devices are far more powerful. In addition, their size as compared to in-person computers contributes significantly to their expanding appeal. As the latest Statista report [14] shows, smartphones users reached more than 6 billion in 2022 and are expected to exceed 7 billion in 2027 (see Figure 1).

The most widely used operating systems are Android and iOS, and their simple usage and processing capacity to operate a broad range of apps draw millions of users globally [18]. Android also seems to be more popular than other operating systems. According to a recent report, Android represents 71.7% of the operating systems market share in 2022 [17]. Such widespread acceptance brings with it major security and privacy risks, as well as a slew of other malicious behaviors [9]. Malware assaults have increased in frequency. A recent report revealed that the total of mobile malware and new malware were doubled between 2019 and 2020 [6]. As a result, malware detection has become increasingly important for mobile systems. Furthermore, when cybercriminals know their malware will

*Corresponding author

Email addresses: waheedf.alkaaby@student.uokufa.edu.iq (Waheed Fadel Waheed), hasanen.alyasiri@uokufa.edu.iq (Hasanen Alyasiri)

Received: April 2022 *Accepted:* July 2022

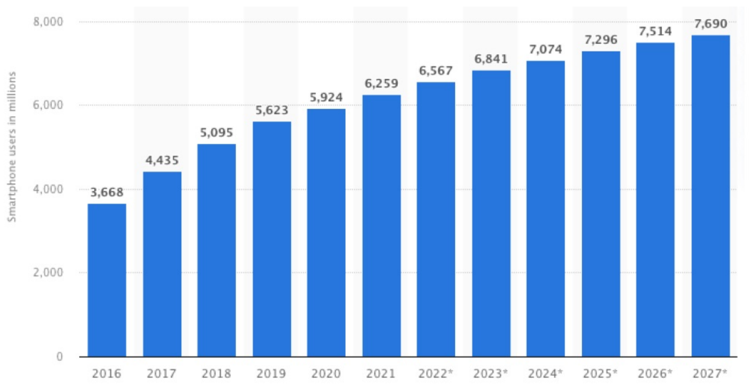

Figure 1: The growth of smartphone users from 2016 to 2027 [14]

be identified, cybercriminals use a variety of concealment tactics to trick anti-malware scanners. Thus, there is a need for adopting solutions that could identify new malicious applications. Although numerous malware detection methods have been developed, there are common drawbacks to the current solutions. Such as the static analysis uses statically extracted signatures (behaviors not at runtime) to detect malware. Polymorphism (changing code to avoid detection by handwriting tools) and concealment are advanced strategies used by advanced malware programs to evade detection by user defined malware detection (hiding evidence of malicious activities) [12]. Using the activities taken by the malware after it has been executed in a sandbox environment, a behavior-based method known as dynamic analysis determines the functioning of the virus. Traditional strategies must be supplemented with new ones in order to safeguard mobile systems adequately. This research's focus is on Machine Learning (ML), which has improved significantly over the previous several years and is particularly relevant to the work discussed in this paper. ML algorithms are now being used by several researchers to enhance the security of many systems, including mobile. As shown below, our research takes a specific methodology.

A subset of Machine Learning (ML) algorithms, Evolutionary Algorithms (EAs) have created intelligent solutions in several domains, including software testing, computer networks, medicine, and art [16]. Additionally, EA-based approaches have been successfully applied for research areas on security [2, 3, 16]. EA-created solutions have a number of desirable properties, including being easy to understand, lightweight, and utilising fewer features than conventional ML techniques [16]. These properties are especially important when dealing with security issues. Evtree, an evolutionary algorithm, is used in this article to develop solutions for identifying malware targeting Android devices. To the best of our knowledge, this is the first work to investigate how well the evtree algorithm protects mobile devices. In addition, a recent report showed that mobile devices have been targeted through new ways of attacks to bypass security screening [6]. Consequently, we'll examine how the evtree method performs in the context of this problem.

The following is the structure of this work: Section 2 provides a background of the study including an overview of previous research; Section 3 discusses the methodology; Section 4 outlines the experiments undertaken and the outcomes; and Section 5 summarizes the findings and includes some future research directions.

2 Background of study

History of Android

Google's Android operating system was developed by Rich Miner, Nick Sears, Chris White, and Andy Rubin in 2003 as part of the Android Inc firm [5]. Android's origins date back to its days as a camera OS, designed to improve wireless communication between cameras and personal computers (PCs). The operating system was tailored to make mobile phones smarter within a few months of the company's inception [1]. In August 2005, Google bought Android. The Linux kernel was used to build the Android operating system, which is free for mobile phone makers to use [5]. Android OS 1.0 was launched in November 2007 and the first Android phone went on sale in September of that year. With each passing year, Android OS has progressed to better and better versions. However, the closed framework provides more data protection and is more pristine than the open-source ones.

Security in Android

From a simplistic SMS sending Trojan, Android malware has advanced to more futuristic codes that can corrupt other applications, encrypt user data, gain root privileges, download, and install more applications that are malicious bypassing the knowledge of the users, and load a payload from a remote server to cellular phones. These threats could lead to personal information theft, loss of privacy, monitoring of the users, financial loss through ransomware, and remote operation of the phones. Several works have been carried out by researchers in detecting android malware detection including static analysis, dynamic Analysis, machine learning. A recent study showed that ML achieved a better approach with higher detection accuracy than other approaches [15].

A considerable amount of literature has been conducted in the detection of malware using ML algorithms, part of which includes:

Murad et al. (2014) used feature selection with a genetic algorithm (GA), a method for detecting Android malware. was presented (GA). For recognizing and investigating malicious software on Android, three deferent classier techniques were created using different characteristic subsets which were identified using GA. This dataset included 1740 samples, including 1119 malwares as well as 621 benigns, and the highest accuracy was achieved using a combination of SVM and GA, at 98% [13].

M. Arif et al. (2018) An algorithm based on biological principles was used to select trustworthy permission characteristics that are capable of detecting malware attacks. A static analysis method with classifiers has been developed to identify potentially harmful applications. This method makes use of permission capabilities offered by Smartphone mobile device. During the evaluation, 5000 Drebin botnets and 3500 innocuous samples were employed, and also the technique achieved great detection performance, with a predictive performance of 95.6% [4].

Mahdavifar et al. (2020) proposed an effectual Mobile malware categorization classification system made use of a quasi-pseudo-label recurrent neural network. Because there were just a few training examples datasets available, their technique outperformed training data that employed a learning algorithm. Their data set included 11,598 inter applications with features that were spread throughout all five types of malware. (i.e., banking, adware, short message service (SMS), and benign, riskware) Their approach obtained an accuracy level of 96.7% in the detection of malware by using a predetermined hidden layer in addition to hidden neurons [12].

Y. Xu et al. (2021) It is suggested that a group Convolution layer that is built upon feature interactions be used in addition to learning characteristics and reconstruct vulnerability information. First, factor loadings were generated to quantify the relationships between the characteristics. Next, the values obtained were sorted in highest to the lowest, and also the data was categorized by columns. Second, the 1D cluster CNN model is built. This model consists of multiple 1D convolutional with multiple 1D pooled filters. Finally, the rebuilt characteristics are used as input to shadowing machine learning techniques that anticipate possible threats to the network. Its FLOP, variables, and runtime of the group CNN was lowered as compared to the standard 1D CNN [19].

E. T. Elkabbash et al. (2021) unique detecting strategy that leverages the synthetic Jellyfish Searching (JS) optimization to enhance the spontaneous vector copying of iPhone information (RVFL). Using 11,598 inter programs with 471 kinematical factors, RVFL+JS seeks to reduce the completion times of optimized classes with the highest performance measures [8].

T. Dong et al. (2022) They construct FEDFOREST, an innovative collaborative training algorithm-based NIDS, by incorporating FL as well as GBDT. FEDFOREST can accurately identify the cyberwar for various tasks by offering its exact classification, and zero-day threats could be quickly included within the FEDFOREST structure. Additionally, two new confidentiality techniques have been suggested to further safeguard the private information inside the FL scenario. FEDFOREST can provide effectiveness, interpretability, as well as expandability for various cyber warfare detection methods, which outperform established DNNs-based NIDS according to our numerous tests [7].

3 Proposed approach

Using evolutionary algorithms, we synthesize detection systems using supervised learning modules. Using a collection of predictor factors, trees are used to predict a class (i.e., extracted features). Other researchers have previously extracted characteristics from the dataset used in their efforts to create an Android malware detector, which comprises benign and infected Android samples. Consequently, we'll begin with the learning phase and evaluate the performance of each developed rule. All either sample have been successfully identified or the maximum number of generations has been achieved, at which point the training phase comes to an end. after that, the best-evolved programs are sent into the testing phase for evaluation. The structure of our approach is shown in Figure 2.

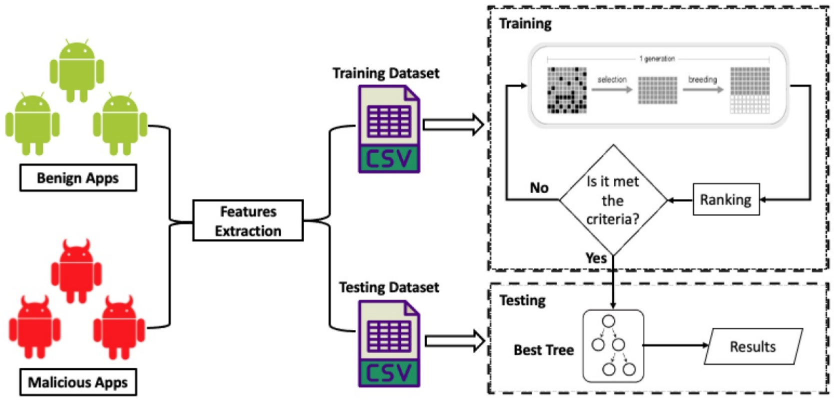

Figure 2: General Overview of The Proposed Methodology

3.1 Classification using evtree Algorithm

This work aims to create a classification tree that can better identify Android malware. evtree, an evolutionary algorithm for classifying trees, is used. The evolutionary approach influenced by Darwinian evolution is used to implement the evtree population algorithm. It was first introduced by Grubinger et al. in 2014 [10]. An EA's typical workflow is shown in Figure 3:

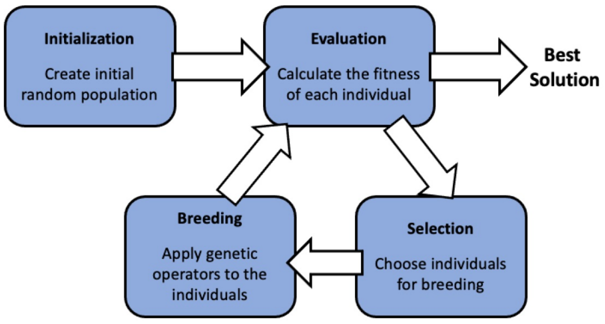
Initialization
Create initial random population
Evaluation
Calculate the fitness of each individual
Best Solution
Breeding
Apply genetic operators to the individuals
Selection
Choose individuals for breeding

Figure 3: General Overview of The Evolutionary Algorithm

It produces trees (i.e., candidate solutions) and then alters them regularly after each generation to develop the classification tree for the target environment. Root nodes are seeded with a random, valid split rule in the early stages of population growth. The population's needs are reflected in the evaluation function. Individuals are judged based on their misclassification rate and complexity using the evtree algorithm (i.e., terminal nodes used). The objective is to increase accuracy on the training data and reduce complexity. Genetic operators are applied to a different tree once every iteration. To create new individuals, crossover and mutation operators are used. Our approach makes use of four different kinds of mutation operators as well as one crossover operator. When two randomly picked trees are crossed over, the result is two new trees. When using the crossover operator, a random member of the remaining population is chosen to serve as the second parent. Split picks a random terminal-node and applies a valid, randomly generated split rule to it in the event of mutation. Consequently, the terminal node chosen becomes an internal node and two additional terminal nodes are created. To prune, Prune selects a random internal node that has two terminal nodes as its successors then prunes it. In a major split rule mutation, a random internal node is selected, and the split rule, described by the related split variable, and the split point are changed. When compared to the operator for major split rule mutation, a minor split rule mutation is quite similar. Only slightly alters and does not impact, however it does shift the split point by a little amount. The resulting offspring are then reintroduced into the general population. Each step of the process is repeated until a predetermined stopping point is reached, such as the completion of the specified number of generations.

It is possible to imitate evtree by altering various (hyper)parameters. In general, the best parameters for a given dataset are determined by the specifics of that dataset. Parameter options that have been tested in our work are discussed here. For instance, setting the probability of genetic operators to c40m30sp30. Utilizing this setting, the crossover operator has a 40% chance of being selected, the mutation operators have a 30% chance of being selected (the minor split rule mutation has a 15% chance, and the major split rule mutation has a 15% chance), and a 30% probability for selecting one of the split (with 15% probability) or the prune operators (with 15% probability). We tested five probability settings including: c0m50sp50, c20m20sp60, c20m40sp40, c20m60sp20, and c40m30sp30. To test the usefulness of the variable operation parameters, we run it with various of iteration (100, 500, and 1000). Figure 4 shows evtree's accomplishment with adopted settings probabilities using various iterations. Using 100 iteration, c0m50sp50 should a better performance than other settings. However, all settings showed similar and better performance when the iteration was set to 1000. With a search over 1000 iterations gave no improvement in the performance for all variation operator settings. Additionally, we tested various population of trees to check their effect on the results. Figure 5 depicts outcomes of evtree algorithm with population sizes of 250, 500, and 1000 trees. It can be seen that, all adopted setting were similar and better when the population was 1000, however, there is no increase in the performance when the population size is larger than 1000. Finally, for used dataset, due to no difference among adopted settings, we used c40m30sp30 with 1000 for iteration to report results in this work.

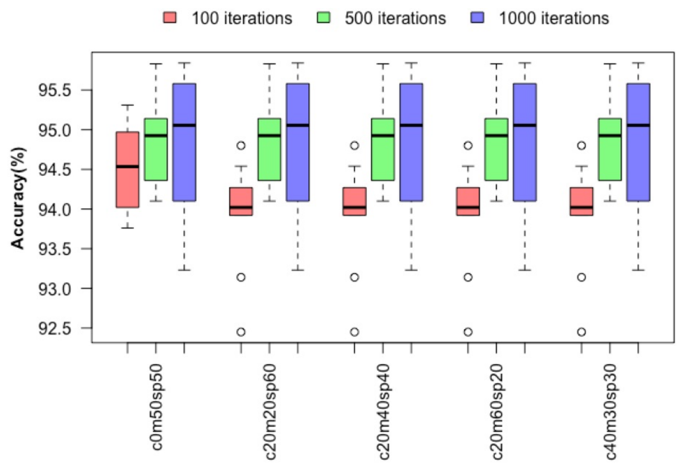

Figure 4: Results of evtree with different variation operator settings and number of iterations

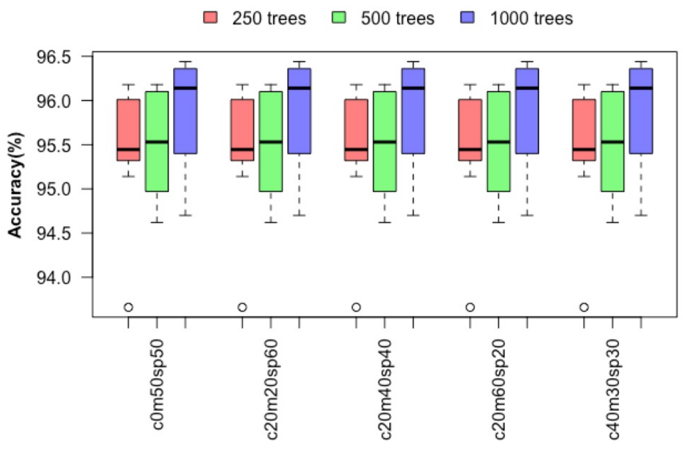

Figure 5: Results of evtree with different variation operator settings and population sizes

3.2 Experimental Evaluation

An Intel Core i7-based computer with 8 GB of RAM and macOS Catalina was used for these tests. To develop the technique, a R package named evtree was used [10].

3.3 Evaluation Metrics

evtree performance can be measured according to how Android applications (i.e., malicious, and benign) are classified. A true positive (TP) refers to malware application being classified as a malware, whereas true negative

(TN) indicates benign application being classified as benign. False positives (FP) and false negatives (FN) are the opposite of each other. A malicious program may alternatively be categorized as a ”benign” application due to false positives (FP) and false negatives (FN). We adopted the following metrics:

The detection rate indicates the proportion of values predicted as malware that were malware:

$$\text{Detection rate } (DR) = TP/(TP + FN)$$

The accuracy shows the percentage of correct classification decisions:

$$\text{Accuracy } (Acc) = (TP + TN)/(TN + TP + FN + FP)$$

The Precision describes the fraction of raised malware that are correct:

$$\text{Precision} = TP/(TP + FP)$$

Finally, the F1 score is a harmonic average of accuracy and recall:

$$F1 - score = (2 \times Recall \times precision)/(Recall + precision)$$

3.4 Description of CICMalDorid2020 Dataset

Data from the University of New Brunswick’s Canadian Institute for Cybersecurity website was used in this study (https://www.unb.ca/cic/datasets/maldroid-2020.html). This data collection contains the following characteristics:

1. Big. It has more than 17,341 Android samples.
2. Recent. It includes recent and sophisticated Android samples until 2018.
3. Diverse. It has samples spanning between five distinct categories: Adware, Banking malware, SMS malware, Riskware, and Benign.
4. Comprehensive. It includes the most complete captured static and dynamic features.

Each category in CICMalDorid2020 dataset is described briefly in the following [11, 12]:

Adware: When a genuine app is infected with malware and infected Mobile Adware, the advertising content (i.e., adverts) often hides inside the legitimate app (available on the third-party market). In order to keep the advertisements running, the ad library employed by the virus performs a sequence of processes repeatedly (even if the victim tries to force-close the app). For example, malware might infect a device, making it vulnerable to Adware and letting hackers steal sensitive data.

Banking Malware: Is malware that mimics the actual banking applications or web interface in order to get access to a user’s online banking accounts. This virus is mostly Trojan based, with the goal of getting into devices, stealing sensitive information like bank logins and passwords, then sending the data back to a central server for further use by the criminals behind it.

SMS Malware: Use the SMS service to execute attacks by intercepting the SMS payload. Malware on the attackers’ hosting servers is required before they can connect the SMS. There are several ways that the C&C server is put to use by the bad guys to carry out their harmful activities.

Mobile Riskware: Riskware is a word used to describe lawful applications that have the potential to do damage if bad persons use them. So, it may evolve into adware or ransomware, which boosts functionality by installing additional infected software on the victim’s computer. There is just one version of this category, which is normally classified as ”Riskware” by Virus Total.

Benign: All additional software that does not fall into one of the categories listed above is considered benign, indicating that it is not harmful.

Android samples were obtained from a variety of sources, including VirusTotal, the Contagio security blog, AMD, MalDozer, and other recent contributions to research. We used the dataset file with 470 extracted features form Android based applications. Components in this file (i.e., features) include the description of system call frequencies, bindings, and composite behaviours. The total number of samples is 11,598 that distributed as detailed in Fig. 6. Benign samples represent around 15% of the dataset while SMS malware is with the most sample around 33%.

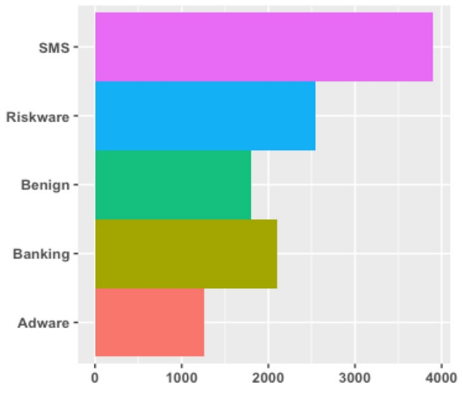

Figure 6: CICMalDorid2020 Dataset Sample Distribution

4 Results

To eliminate dependency on the random seed and biases estimation in performance evaluation, we used 10-fold cross-validation approach. The average of the best-evolved rules on the testing dataset are reported. Figure 7 shows the achieved outcomes of evtree algorithm using different number of trees (i.e., 250, 500, and 1000) in term of precision, detection rate, F1-score, and accuracy. Furthermore, each barplot, which shows the average of 10 runs, with an error bar on top which indicates standard error around the average. The standard error reports the change in average with various experiments carried each time. It can be calculated using $(\sigma/\sqrt{N})$ where N is set to 10 (i.e., total number of runs). As noted, evtree performance where similar when executed using different number of trees, however, with favour in some cases towards evolved rules with population of 1000.

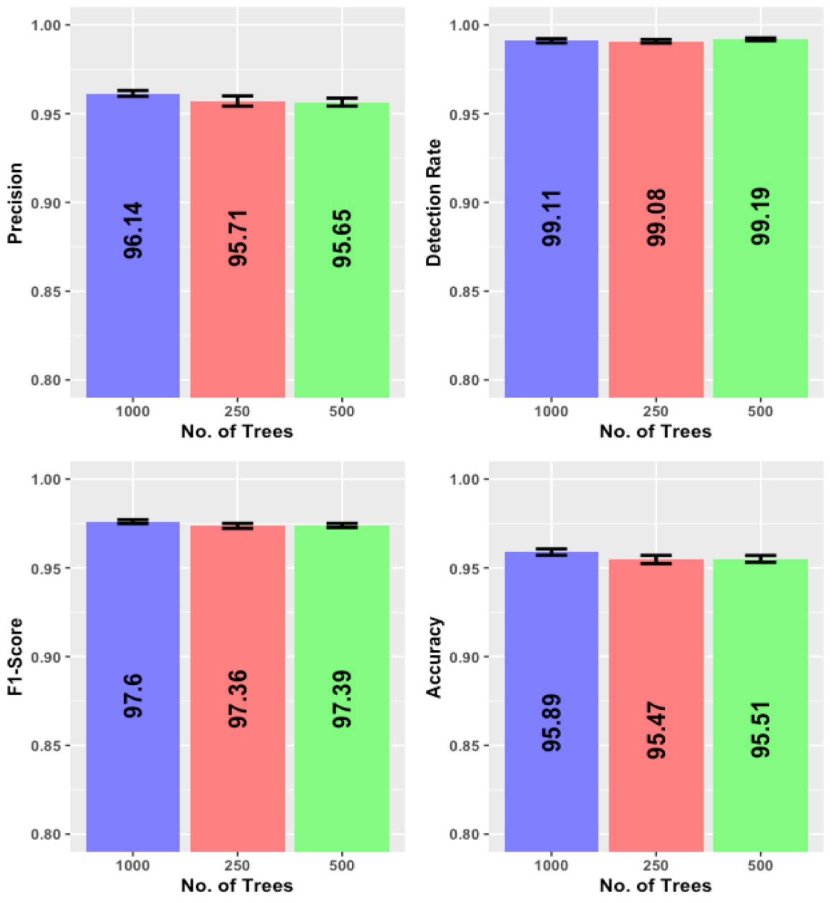

Figure 7: evtree performance results

When the result of this study is compared with the previous works' shown in Table 1, the results showed that evtree algorithm was effective, leading to better or comparable results accomplished by other approaches. Furthermore, evtree is far less complex than other proposed techniques and it produced lightweight solutions (i.e., reduced tree complexity).

Table 1: Comparing Results Against Existing Security Systems using CICMalDorid2020 Dataset

Reference	Classifier	Precision	DR	F1-score	Acc
Mahdavifar et al. (2020) [12]	Pseudo-Label Deep Neural Network	99.16	96.54	97.84	96.7
Xu et al. (2021) [19]	Group Convolutional Neural Networks	NA	NA	77	81.11
Elkabbash et al. (2021) [8]	Random vector functional link and artificial Jellyfish Search optimizer	NA	97.29	97.51	97.22
Dong et al. (2022) [7]	Gradient Boosting Decision Tree and Federated Learning	NA	NA	88.59	89.63
Proposed Approach	evtree	96.14	99.11	97.60	95.89

4.1 evtree Algorithm for Detecting Unknown Malware

Most Android defence systems have been built employing signature-based or standard behaviour of used Apps. Nevertheless, these are insufficient for detecting unknown malware. We evaluated evtree algorithm's capability to learn attributes of known malware which are usually modifications of earlier malware. The CICMalDorid2020 dataset was used for this investigation. This dataset has 4 types of malware (see Figure 8). Different malware has been included into our training and testing sets. Thus, we deleted its training samples and put them to the testing component for each form of malware that mimics an unknown malware. We ran the algorithm ten times for each investigation, a total of 40 independent runs for all malwares found in the utilised dataset. Figure 8 shows the range of reactions to the detection rate of each unknown malwares.

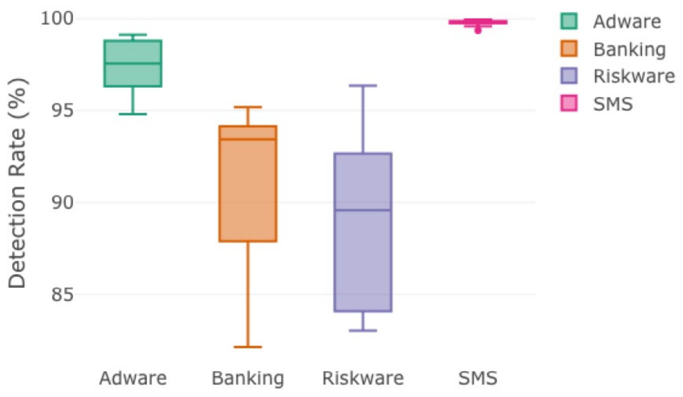

Figure 8: CICMalDorid2020 Dataset Sample Distribution

The boxplots show that evtree algorithm was capable of detecting unknown malware with high and steady performance, specifically Adware, and SMS malware. However, it has accomplished the most inferior detection for Banking and Riskware malware compared to other unknown malware. Therefore, in the future, we require to include different features that may contain malware behaviour. Overall, these outcomes indicate that evtree algorithm has successfully categorised unknown malware that were excluded from training.

5 Conclusions and Future Work

This study provides a preliminary examination of the evtree algorithm for Android malware detection. Such system has become a favoured target of cyber-attacks. Results show the ability of the proposed method to classify malware threats. Our proposed method was able to convert extracted features, which describe App behaviour, into a decision space that distinguishes benign from malware Apps. The results indicated that evtree algorithm demonstrated a strong performance in identifying malware that was absent from the training phase. Future work includes leveraging evtree to develop an ensemble model (i.e., a group of detectors) to identify risks to Android environments. Additionally, uses evtree to classify Android malware into different categories.

References

[1] J. Alabaster, *Android founder: We aimed to make a camera OS*, PC World, Apr. Accessed: Jul. **17** (2013), 2020.

[2] H. Alyasiri, *Evolving rules for detecting cross-site scripting attacks using genetic programming*, Int. Conf. Adv. Cyber Secur., 2020, pp. 642–656.

[3] H. Alyasiri, J.A. Clark and D. Kudenko, *Evolutionary computation algorithms for detecting known and unknown attacks*, Int. Conf. Secur. Info. Tech. Commun., 2018, pp. 170–184.

[4] M. Arif, I. Naseem, M. Moinuddin and U.M. Al-Saggaf, *Design of an intelligent q-LMS algorithm for tracking a non-stationary channel*, Arab. J. Sci. Eng. **43** (2018), no. 6, 2793–2803.

[5] J. Callaham, *The history of Android OS: Its name, origin and more*, Android Auth. **18** (2019).

[6] C. Castillo and R. Samani, *McAfee Mobile Threat Report 2021*, (2021), https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf

[7] T. Dong, S. Li, H. Qiu and J. Lu, *An interpretable federated learning-based network intrusion detection framework*, arXiv Prepr.arXiv2201.03134, (2022).

[8] E.T. Elkabbash, R.R. Mostafa and S.I. Barakat, *Android malware classification based on random vector functional link and artificial jellyfish search optimizer*, PLoS One **16** (2021), no. 11.

[9] A. Feizollah, N.B. Anuar, R. Salleh and A.W.A. Wahab, *A review on feature selection in mobile malware detection*, Digit. Invest. **13** (2015), 22–37.

[10] T. Grubinger, A. Zeileis and K.-P. Pfeiffer, *evtree: Evolutionary learning of globally optimal classification and regression trees in R*, J. Stat. Softw. **61** (2014), 1–29.

[11] S. Mahdavifar, D. Alhadidi and A.A. Ghorbani, *Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder*, J. Network Syst. Manag. **30** (2022), no. 1, 1–34.

[12] S. Mahdavifar, A.F.A. Kadir, R. Fatemi, D. Alhadidi and A.A. Ghorbani, *Dynamic android malware category classification using semi-supervised deep learning*, IEEE Intl. Conf. Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020, pp. 515–522.

[13] M.H. Murad and S. Fatema, *Some static relativistic compact charged fluid spheres in general relativity*, Astrophys. Space Sci. 350 (2014), no. 1, 293–305.

[14] S. O'Dea, *Number of smartphone subscriptions worldwide from 2016 to 2027*, https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, (2022).

[15] M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius and R. Maskeliunas, *Android malware detection: A survey*, Int. Conf. Appl. Inf., 2018, pp. 255–266.

[16] S. Sen, *A survey of intrusion detection systems using evolutionary computation*, Bio-inspired Comput. Telecommun., Elsevier, (2015), pp. 73–94.

[17] S.G. Stats, *Mobile operating system market share worldwide*, https://gs.statcounter.com/os-market-share/mobile/worldwide, 2019.

[18] L. Tung, *Bigger than windows, bigger than iOS: Google now has 2.5 billion active Android devices*, https://www.zdnet.com/article/bigger-than-windows-bigger-than-ios-google-now-has-2-5-billion-active-android-devices-after-10-years/, 2020.

[19] Y. Xu, X. Zhang, C. Lu, Z. Qiu, C. Bi, Y. Lai, J. Qiu and H. Zhang, *Network threat detection based on group CNN for privacy protection*, Wirel. Commun. Mob. Comput. **2021** (2021).