Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning

Anam Fatima*, Ritesh Maurya*, Malay Kishore Dutta*, Radim Burget† and Jan Masek†
*Computer Science and Engineering, Centre for Advanced Studies, Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India
†Department of Telecommunications, Brno University of Technology, Brno, Czech Republic
Email: 17mcs01@gmail.com, maurya123ritesh47@gmail.com, malaykishoredutta@gmail.com, burgetrm@feec.vutbr.cz, masek@feec.vutbr.cz

***Abstract*—Android platform due to open source characteristic and Google backing has the largest global market share. Being the world's most popular operating system, it has drawn the attention of cyber criminals operating particularly through wide distribution of malicious applications. This paper proposes an effectual machine-learning based approach for Android Malware Detection making use of evolutionary Genetic algorithm for discriminatory feature selection. Selected features from Genetic algorithm are used to train machine learning classifiers and their capability in identification of Malware before and after feature selection is compared. The experimentation results validate that Genetic algorithm gives most optimized feature subset helping in reduction of feature dimension to less than half of the original feature-set. Classification accuracy of more than 94% is maintained post feature selection for the machine learning based classifiers, while working on much reduced feature dimension, thereby, having a positive impact on computational complexity of learning classifiers.**

***Keywords*—Android malware analysis; feature selection; Genetic algorithm; machine learning; reverse-engineering**

I. INTRODUCTION

Android Apps are freely available on Google Playstore, the official Android app store as well as third-party app stores for users to download. Due to its open source nature and popularity, malware writers are increasingly focusing on developing malicious applications for Android operating system. In spite of various attempts by Google Playstore to protect against malicious apps, they still find their way to mass market and cause harm to users by misusing personal information related to their phone book, mail accounts, GPS location information and others for misuse by third parties or else take control of the phones remotely. Therefore, there is need to perform malware analysis or reverse-engineering of such malicious applications which pose serious threat to Android platforms.

Broadly speaking, Android Malware analysis is of two types: Static Analysis and Dynamic Analysis. Static analysis basically involves analyzing the code structure without executing it while dynamic analysis is examination of the runtime behavior of Android Apps in constrained environment. Given in to the ever-increasing variants of Android Malware posing zero-day threats, an efficient mechanism for detection of Android malwares is required. In contrast to signature-based approach which requires regular update of signature database, machine-learning based approach in combination with static and dynamic analysis can be used to detect new variants of Android Malware posing zero-day threats. In [1], broad yet lightweight static analysis has been performed achieving a decent detection accuracy of 94% using Support Vector Machine algorithm. Nikola Milosevic et al. [2] presented static analysis based classification through two methodologies: one was permissions based while the other involved representation of the source code as a bag of words. Another approach based on identifying most significant permissions and applying machine learning on it for evaluation has been proposed in [3].

MADAM[4] provides an multi-level analysis framework where behavior of Android Apps is captured upto four levels: from package, user, application to kernel level, achieving detection accuracy as high as 96% with one disadvantage being that it could run only on rooted devices, making it incapable for commercial use. SAMADroid[5] proposed a three-way novel host server based methodology for improved performance as far as asset usage is concerned for malware detection on mobile devices. The current drift in malware detection has shifted towards deep learning applications where it requires least human intervention as proposed in [6]–[8].

An important step in all machine learning based approaches is feature selection. Obtaining optimal feature set will not only help in improving experimentation results but will also help in reducing the curse of dimensionality associated with most machine learning based algorithms. Fest [9] proposed a novel and efficient algorithm for feature selection to improve overall detection accuracy. In [10], a review of various feature selection algorithms for malware detection has been presented providing guidelines for selection.

In the proposed work, Genetic algorithm has been used because of its capabilities in finding a feature subset selected from original feature vector such that it gives the best

Acknowledgement: The authors would like to thank Ministry of Industry of the Czech Republic by the grant FV20044 and the National Sustainability Program under grant LO1401 for funding the work and C3i Center (Interdisciplinary Center for Cyber Security and Cyber Defense of Critical Infrastructures), IIT Kanpur, India for sharing their Android Applications dataset. For the research, infrastructure of the SIX Center was used.

accuracy for classifiers on which they are trained. It has been used, previously also, in combination with machine learning and deep learning algorithms to obtain the most optimal feature subset as in [7], [11].

The main contribution of the work is reduction of feature dimension to less than half of original feature-set using Genetic Algorithm such that it can be fed as input to machine learning classifiers for training with reduced complexity while maintaining their accuracy in malware classification. In contrast to exhaustive method of feature selection which requires testing for 2^N different combinations, where N is the number of features, Genetic Algorithm, a heuristic searching approach based on fitness function has been used for feature selection. The optimized feature set obtained using Genetic algorithm is used to train two machine learning algorithms: Support Vector Machine and Neural Network. It is observed that a decent classification accuracy of more than 94% is maintained while working on a much lower feature dimension, thereby, reducing the training time complexity of classifiers.

The remaining paper is structured as follows: Section II discusses about the proposed methodology used. Section III presents experimentation results obtained by applying genetic algorithm for feature selection to train machine learning algorithms. Section IV briefs about the general conclusions drawn from the experimentations.

II. PROPOSED METHODOLOGY

Two set of Android Apps or APKs: Malware/Goodware are reverse engineered to extract features such as permissions and count of App Components such as Activity, Services, Content Providers, etc. These features are used as feature-vector with class labels as Malware and Goodware represented by 0 and 1 respectively in CSV format.

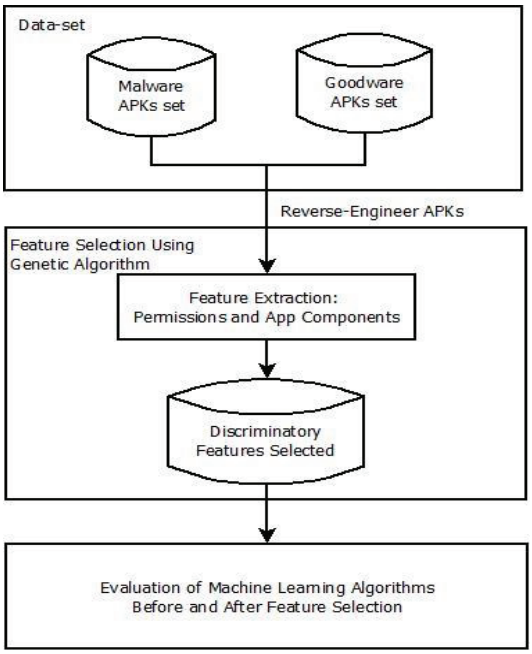


Fig. 1. Proposed Methodology

To reduce dimensionality of feature-set, the CSV is fed to Genetic Algorithm to select the most optimized set of features. The optimized set of features obtained is used for training two machine learning classifiers: Support Vector Machine and Neural Network.

Figure 1 gives a brief about proposed methodology, basically involving two units: feature extraction using Androguard tool and feature selection using Genetic Algorithm. Finally, the selected features are fed as input to machine learning algorithms for evaluation purpose.

A. *Reverse-Engineering of Android APKs*

In the proposed methodology, static features are obtained from AndroidManifest.xml which contains all the important information needed by any Android platform about the Apps. Androguard tool has been used for disassembling of the APKs and getting the static features.

B. *Feature Vector*

Features are extracted and mapped to a feature vector as follows:

- App Components: The counts of App components such as Activity, Services, Content Providers and Broadcast Receivers are used as a feature vector.
- Permissions: The permissions feature-set are mapped to a |S| dimensional vector space such that a dimension is set to 1 if the app x contains the feature and 0 otherwise. In this way, a vector $\psi(x)$ is constructed for each feature extracted from app x with the respective dimension is set to 1 and all other dimensions to 0 [5]. It can be summarized in equation (1):

$$\psi: X \rightarrow \{0;1\}^{|S|} \quad (1)$$

C. *Discriminatory Feature Selection*

In malware detection, selecting most significant features is an important step as it has a significant impact on quality of experimental results. Also, working on low-dimensional feature vector consisting of only discriminatory features will help in reducing computational complexity of learning classifier.

The CSV consisting of all features is fed into Genetic algorithm which gives best subset of features for the machine learning based classifier.

Features selected are represented by binary form called chromosomes such that if the feature is included it is represented by 1 and if it is excluded it is represented by 0 in the chromosome. The genetic algorithm maintains a subset of features or chromosome called population along with their fitness scores such that chromosome with better fitness scores are given more chance to reproduce. The fitness function of genetic algorithm is defined such that the chromosome that gives high accuracy on the machine learning based classifier is assigned a larger value in comparison to features that give lower accuracy for it. The chromosomes with best fitness score are selected as parent to produce next generation of offspring using the process of crossover and mutation [11].

The steps involved in feature selection using Genetic Algorithm can be summarized as below:

Step 1: Initialize the algorithm using feature subsets which are binary encoded such that if the feature is included it is represented by 1 and if it is excluded it is represented by 0 in the chromosome.

Step 2: Start the algorithm defining an initial set of population generated randomly.

Step 3: Assign a fitness score calculated by the defined fitness function for genetic algorithm.

Step 4: Selection of Parents: Chromosomes with good fitness scores are given preference over others to produce next generation of off-springs.

Step 5: Perform crossover and mutation operations on the selected parents with the given probability of crossover and mutation for generation of off-springs.

Repeat the Steps 3 to 5 iteratively till the convergence is met and fittest chromosome from population, that is, the optimal feature subset is resulted.

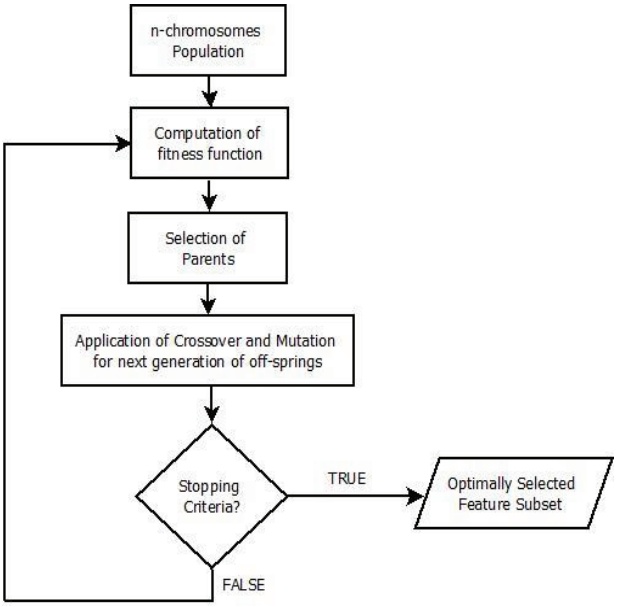


Fig. 2. Feature Selection using Genetic Algorithm

Figure 2 diagrammatically gives a brief about the feature selection done using Genetic Algorithm.

D. Machine Learning Based Classification

Given in to the ever-increasing variants of Android Malware posing zero-day threat, machine learning based techniques are being preferred over traditional signature-based approach which required regular update of signature database. The selected features using Genetic Algorithm are used to train and test the classifiers with following algorithms: Support Vector Machine (SVM) and Neural Network (NN).

III. EXPERIMENTAL RESULTS

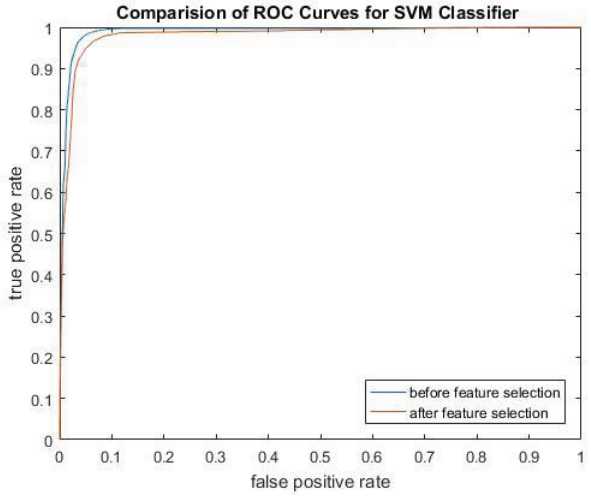
The proposed work has been performed on a dataset of around 40,000 APKs consisting of two categories: 20, 000 Malware or malicious applications and 20,000 Goodware or benign applications. The APKs are reverse-engineered to extract features. A CSV is generated consisting of 99 features with class labels as Malware (represented by 0) and Goodware (represented by 1). The primary purpose of the work is selection of optimized feature subset for which Genetic Algorithm has been used. The discriminatory features selected by Genetic Algorithm are fed as input to train Support Vector Machine and Neural Network classifiers. The parameters for Support Vector Machine are set as follows: Radial Basis Function (RBF) as kernel function and number of folds for cross-validation as 10. The number of hidden layers used for the feed-forward Neural Network is one of size 40.

The algorithms are tested on Intel(R) Xeon(R) Silver 4114 CPU@ 2.20GHz 2.19GHz, 64GB RAM, 64-bit operating system. The performance of these two classifiers in distinguishing between Malware and Goodware is compared before and after feature selection.

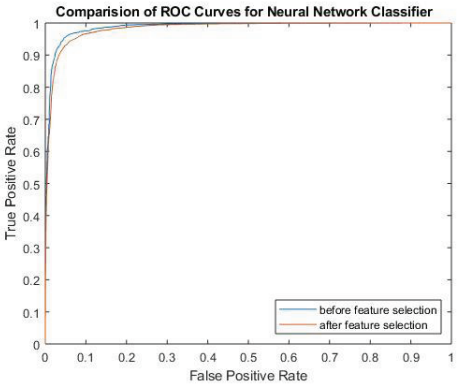
TABLE I. FEATURE SELECTED BY GENETIC ALGORITHM FOR DIFFERENT CLASSIFIERS AND ACCURACY OBTAINED WITH SELECTED FEATURES

Algorithm	**No of features before feature selection**	**AUC**	**Features Selected**	**AUC**
Support Vector Machine	99	.9891	33	.9803
Neural Network	99	.9876	40	.9828

Table I shows the features selected by the Genetic algorithm for different classifiers and classification accuracy of classifier with the selected subset of features obtained from Genetic algorithm. It can be analyzed from table I that the AUC for both classifiers is preserved to quite an extent with significant reduction in number of features.



(a)



(b)

Fig. 3. Comparison of ROC curves for (a) Support Vector Machine (b) Neural Network Classifier Before and After Feature Selection

Fig. 3 shows the ROC curve for different classifiers before and after feature selection. ROC curves for the Support Vector Machine and Neural Network classifiers are shown in fig. 3 (a) and fig. 3 (b) respectively. It can be deduced from the ROC curve that classifiers perform well with the selected features.

TABLE II. PERFORMANCE METRICS OF SUPPORT VECTOR MACHINE CLASSIFIER

Performance Metrics	With 99 features	With 33 features (post feature selection)
Sensitivity (%)	95.5	94.6
Specificity (%)	97.6	95.4
Accuracy (%)	96.6	95.0
Training Time Complexity (secs)	22.92	10.20

TABLE III. PERFORMANCE METRICS OF NEURAL NETWORK CLASSIFIER

Performance Metrics	With 99 features	With 40 features (post feature selection)
Sensitivity (%)	95.6	94.3
Specificity (%)	94.9	93.9
Accuracy (%)	95.2	94.1
Training Time Complexity (secs)	8.57	3.76

Tables II and III show performance metrics before and after feature selection for Support Vector Machine and Neural Network classifiers respectively. As can be observed from ROC curves and performance metrics, both Support Vector Machine and Neural Network when used in conjunction with Genetic Algorithm for feature selection perform significantly well without compromising much in accuracy while working in much reduced feature vector space (less than half of original feature-set), thereby, reducing the classifier training time complexity.

IV. CONCLUSION

As the number of threats posed to Android platforms is increasing day to day, spreading mainly through malicious applications or malwares, therefore it is very important to design a framework which can detect such malwares with accurate results. Where signature-based approach fails to detect new variants of malware posing zero-day threats, machine learning based approaches are being used. The proposed methodology attempts to make use of evolutionary Genetic Algorithm to get most optimized feature subset which can be used to train machine learning algorithms in most efficient way. From experimentations, it can be seen that a decent classification accuracy of more than 94% is maintained using Support Vector Machine and Neural Network classifiers while working on lower dimension feature-set, thereby reducing the training complexity of the classifiers. Further work can be enhanced using larger datasets for improved results and analyzing the effect on other machine learning algorithms when used in conjunction with Genetic Algorithm.

REFERENCES

[1] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proceedings 2014 Network and Distributed System Security Symposium*, 2014.

[2] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided Android malware classification," *Comput. Electr. Eng.*, vol. 61, pp. 266–274, 2017.

[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," *IEEE Trans. Ind. Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[4] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," *IEEE Trans. Dependable Secur. Comput.*, vol. 15, no. 1, pp. 83–97, 2018.

[5] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," *IEEE Access*, vol. 6, pp. 4321–4339, 2018.

[6] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A Multimodal Deep Learning Method for Android Malware Detection using Various Features," vol. 6013, no. c, 2018

[7] A. Martin, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho, "Evolving Deep Neural Networks architectures for Android malware classification," *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, pp. 1659–1666, 2017.

[8] X. Su, D. Zhang, W. Li, and K. Zhao, "A Deep Learning Approach to Android Malware Feature Learning and Detection," 2016 IEEE Trust., pp. 244–251, 2016.

[9] K. Zhao, D. Zhang, X. Su, and W. Li, "Fest : A Feature Extraction and Selection Tool for Android Malware Detection," *2015 IEEE Symp. Comput. Commun.*, pp. 714–720, 4893.

[10] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," *Digit. Investig.*, vol. 13, pp. 22–37, 2015.

[11] A. Firdaus, N. B. Anuar, A. Karim, M. Faizal, and A. Razak, "Discovering optimal features using static analysis and a genetic search based method for Android malware detection *," vol. 19, no. 6, pp. 712–736, 2018.

[12] A. V. Phan, M. Le Nguyen, and L. T. Bui, "Feature weighting and SVM parameters optimization based on genetic algorithms for classification problems," *Appl. Intell.*, vol. 46, no. 2, pp. 455–469, 2017.