RESEARCH ARTICLE

Analyzing and comparing the effectiveness of various machine learning algorithms for Android malware detection

Muhammad Shoaib Akhtar

School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China

Correspondence to: Muhammad Shoaib Akhtar, School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China; Email: 13.cs.194@gmail.com

Received: October 4, 2022;
Accepted: December 10, 2022;
Published: December 14, 2022.

Citation: Akhtar, M. S. (2022). Analyzing and comparing the effectiveness of various machine learning algorithms for Android malware detection. *Advances in Mobile Learning Educational Research*, 3(1), 570-578. https://doi.org/10.25082/AMLER.2023.01.005

Abstract: Android is the most extensively adopted mobile operating system in the world. The free third-party programmes that may be downloaded and installed contribute to this success by offering a wide range of features and functionalities. However, the freedom to utilize any third-party programme has spawned a never-ending tide of ever-evolving malicious software intending to harm the user in some way, shape, or form. In this work, we propose and show many methods for detecting malware on Android. An in-process detection system is built, including data analytics. It may use the detection system to look over your current app set and find any malicious software so you can remove it. Models based on machine learning allow for this to be accomplished. It has been investigated how well the models perform with two distinct feature sets: permissions and signatures. Initially, each dataset undergoes exploratory data analysis and feature engineering to narrow down the vast array of attributes. The next step is to determine if an application is malicious or safe using one of many supervised classification models derived from data mining. Different models' performance metrics are examined to find the method that provides the best outcomes for this malware detection task. Ultimately, it is seen that the signatures-based method is superior to the permissions-based. Classification methods such as k-nearest neighbours (kNN), logistic regression, support vector machines (SVM), and random forests (RF) are all equivalent in their efficacy.

Keywords: Android, cyber security, cyber warfare, malware detection

1 Introduction

The prevalence of malware designed to infect Android smartphones continues to rise, becoming increasingly complex and covert. Machine learning approaches have allowed for the modelling of patterns in both the static features and the dynamic behaviours of Android malware. By analyzing the correlation between classification accuracy and feature quality in machine learning classifiers, we link the characteristics provided by applications with those needed to provide the functionality associated with their category. Instead of identifying potentially harmful trends, our categorization strategy generates reliable static features for safe apps within a given category. We utilize the features of the most popular apps in a given category to train a malware detection classifier for that category. One of the largest selections of Android apps can be found in the Google Play Store, which is divided into 26 unique categories. While each app type serves a unique purpose, they all share standard static and dynamic features. In general, harmless applications of a given kind share several commonalities.

On the other hand, malicious apps often have unusual requirements for their category. The findings of this study recommend enhancing classification models' capability to identify harmful apps by employing category-based machine learning classifiers. Extensive machine learning studies demonstrate that classifiers that use categories to make their predictions outperform those that do not go by a significant margin.

The permission-based procedure that restricts the functionality of unofficial Android apps is the backbone of Android security. When installing an app, the user must give it the permissions it requests. The goal is to ensure the user is well-informed about the risks involved with installing and running a particular piece of software. In reality, two problems exist. First, most people need to learn more about cybersecurity, so they blindly trust the app store or the popularity of a piece of software and download it without looking into the developer's intentions. The second is that Android needs to make it evident during installation what rights and resources an app would require. Instead, it provides details on a collection of resources and makes them available in many forms. The categories include uses of resources that require special approval. User misunderstanding over rights management leads to unintended over-granting access, making it more difficult to identify malicious programmes and laying the groundwork for various attacks.

The study details a technique for spotting potentially harmful Android apps by analyzing their requests for access to private data. Particular attention is being paid to 222 permissions, some of which are set aside for use by other applications. It is an approach to static analysis that combines two established techniques. The first method looks at how often permissions are being asked and how many requests are coming from harmful software. The second strategy relies on possible security weaknesses in granting access. Through comparison, it is shown that Google's four permission protection levels are excessively coarse-grained, hiding the underlying meaning of permissions. The first strategy is more precise and polished in terms of the semantics of permits. A total of 6783 malicious applications and 1993 simple applications were included in our collection. Profiles for each sample based on both methods have been developed and will be used as input for training and learning procedures. Seven different classifiers were used to determine the models' efficiencies. After selecting the most promising ones, we use them to define our classifier, which subsequently demonstrates superb detection and prediction skills. We also made a list of correlations about permission-to-weight that could be used in other researches. (see Figure 1)

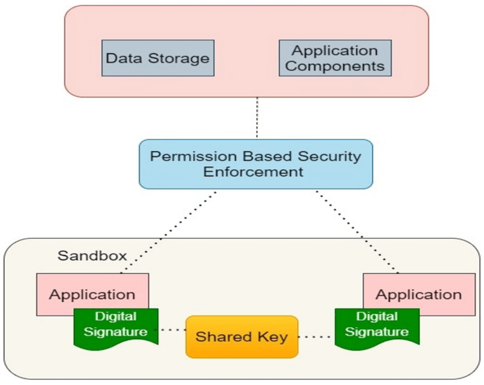


Figure 1 Security Model of Android [28]

Analyses show that our model is one of the most valuable tools, with the given permissions being the only distinguishing attribute. It has the potential to identify almost 99.20% of the 1260 samples of malware supplied by the Genome project, a statistic that is indicative of current malware in terms of its behaviour. This research proposes a strategy for balancing permissions to maintain excellent classification performance when using a dataset of unlabeled samples. This model has an approximately 97% actual positive rate for detecting Android malware and a 95% success rate for forecasting Android malware. This means it can distinguish between virtually any type of malware and accurately forecast when new malware will appear. An AUC between 97 and 99% shows that the system is very good at finding malware.

We also propose a mechanism that can be implemented on Android mobile devices for real-time tracking. Our system outperforms three of the best antivirus tools in two of the three categories we tested.

2 Literature review

Anam (2018) elaborate on the security structure and its open-source availability and has Google's backing. Android has the largest share of the worldwide market. As the most used OS, it is a prominent target for cybercriminals, who often utilize popular programmes to propagate malware. This study presents an effective machine-learning-based technique for Android malware detection by employing a genetic algorithm for developing discriminating feature selection. Machine learning classifiers are trained on the chosen features to evaluate the genetic algorithm's effectiveness in detecting malware. The results are compared to those obtained before and after feature selection. According to the results of the experiments, the genetic algorithm yields the most compelling feature subset, which reduces the feature dimension to less than half of its original value. The classification accuracy of machine learning-based classifiers can remain over 94% even after feature selection, even when working with dramatically reduced feature dimensions. This helps reduce the computational overhead associated with training classifiers (Fatima et al., 2019; Schultz et al., 2001; Firdausi et al., 2010).

Ravi (2018) compares the efficacy of many machine learning methods for detecting android malware, such as naive Bayes, j48, Random Forest, Multi Class Classifier, and Multilayer Perceptron (Hahn et al., 2016). By employing machine learning techniques, we developed a system to classify Android applications according to whether or not they include malware by employing machine learning techniques. To implement this, a user will need a plethora of apps from the Android Market and extract their permissions and functionalities. Classification accuracy and total training time will be used to evaluate the models we have built from the 3258 android app samples our system collected for validation. The experimental findings demonstrate that the multiclass classifier attains the best levels of classification accuracy when compared to other methods. After evaluating many different classifiers for their computational complexity, we found that the Naive Bayes classifier was the most efficient for determining how to classify the malware data set. The classification precision attained in this work surpasses that of other works published in this area. Using feature reduction, classification accuracy may improve significantly (Schultz et al., 2001; Firdausi et al., 2010; Hahn et al., 2016; Siddiqui et al., 2009; Anderson et al., 2012). (see Table 1)

Table 1 Comparison of ML algorithms (Anderson et al., 2012)

S.no	Author	Algorithm	Accuracy
1	Schultz et al.	Navies Bayes	97.11%
2	Firdausi et al.	J48	97.00%
3	Sebastian et al.	Random forest	96.02%
4	Siddiqui et al.	Random forest	96.60%
5	Anderson et al.	SVM	98.70%
6	This work	Multiclass classifier	99.90%

This study aims to ascertain whether or not the Android manifest file has adequate information to decide about an app's potential danger. In particular, it compares the benefits of exchanging intent between applications with asking for permission. Static malware detection is also improved by using the manifest file dataset to test and fine-tune several machine-learning techniques. A support vector machine (SVM) approach is the most efficient classifier across the board (91.7%). A fully effective classifier may be built using the manifest file alone. While the 1.2% boost in classifier performance is not huge, it warrants more research into whether or not intent filters, in addition to permissions, may be beneficial. This is significant as modern malware often uses implied intent to influence unintended programmes and avoid detection by traditional means. The methodology and the data collection may use tweaking (Fatima et al., 2019; Zhao et al., 2015; Varma et al., 2017).

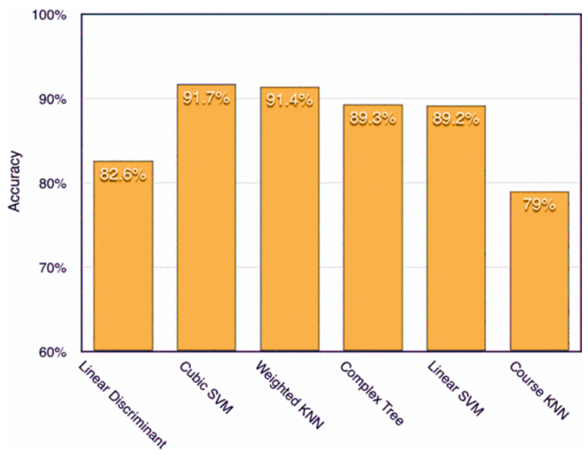


Figure 2 Comparing Algorithms on Combined Dataset (Varma et al., 2017)

As shown in Figure 2, the SVM has the best success rate (91.7%). The following best algorithms only improved accuracy by 91.4% (KNN) and 89.3% (Complex Tree), respectively; this is a small gap.

This work proposes a strategy to scan Android devices for malware based on Application Programming Interface classes. Here, we use machine learning to identify potentially harmful software updates. We also compare the precision of several machine learning methods. In this research, we employ cross-validation, a percentage split test, and the Random Forest, J48, and Support Vector Machine algorithms to classify 51 API package classes from 16 API classes as either benign or harmful. A total of 412 representative application examples are utilized. To put it another way, we discover that our classifications are, on average, 91.9% accurate (Kumaran & Li, 2016; Westyarian et al., 2015). (see Table 2)

In this study, we demonstrate the potential of machine learning for malware detection by utilizing preexisting classes and packages with three different algorithms (Support Vector

Table 2 Precision rate of algorithms

Categories	SVM	Tree J48	Tree Random Forest	Highest
Cross Validation Percentage Split	92.10% 91.40%	S9 90% 90.30%	92.40% 90.50%	92.40% 91.40%

Machine, J48, and Random Forest). The cross-validation test accuracy percentage is 92.4% using the random forest methodology but only 91.4% with the SVM method (Kumaran & Li, 2016; Westyarian et al., 2015).

As the number of people who rely on mobile devices grows, so does the need for sophisticated virus detection algorithms. This paper describes preliminary research on analyzing Android app permission requests to detect malware. With the help of an AI-driven system, we can distinguish between benign and malicious programmes. Once we found the sweet spot for the suggested model's parameters, we saw our dataset classification accuracies of 75% to 80%. Future studies may consider increasing the size of the training dataset, incorporating more characteristics, and experimenting with alternative machine-learning methods. Increased dependence on mobile devices underscores the importance of developing and implementing sophisticated methods for detecting viruses. This research demonstrated preliminary steps toward developing an Android malware detection system by analyzing app permission requests. This model achieved 80% accuracy on the dataset in classifying over 3000 Android apps (Tahtaci & Canbay, 2020; Leeds & Atkison, 2016).

3 Research problem

Android has been a target for cybercriminals because of its widespread use and the positive reception its user-friendly design and other features have received. Malware detection techniques for Android that depend on signatures or monitoring power use may miss the most current threats. So, we develop a new way to find malware that uses machine learning.

4 Methodology

This section will discuss our techniques to distinguish safe programmes from harmful ones. The entirety of the system's architecture is seen in Figure 3. We compile a database to determine which apps are trustworthy and which are not. The entire set engages in feature extraction. A feature vector is created for classification based on the retrieved features. We also provide a report examining how often permits are claimed and how often they go unclaimed. Last but not least, we guarantee that the software may be downloaded from Google Play. Our research required a huge dataset consisting of both clean and malicious applications. Applications from the Google Play Store were used to ensure user safety, while the Mahindr (2008) dataset was used ethically to investigate harmful apps.

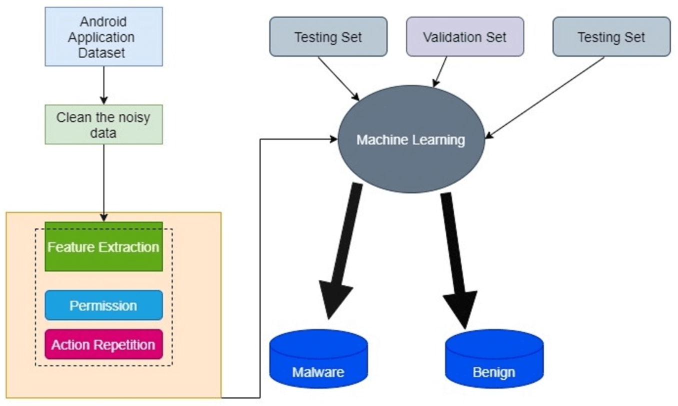


Figure 3 (a) Proposed method

In this paper, I demonstrate the effectiveness of both a permission-based and signature-based method using two datasets. Researchers construct a wide range of classification approaches using publicly available labelled data sources to distinguish harmful from security software. Data about the permissions granted to programmes are saved in one database, while API call signatures are kept in another. Several trained data mining models have their performance evaluated and compared using metrics like accuracy and recall. First, we evaluate classification models using the same approach. The most reliable results from each method are compared to determine which tool is better at finding malware.

Final categorization is accomplished using dynamic analysis based on machine learning, which has access to the collected dynamic characteristics. Questionable apps are evaluated and labelled as harmful or safe during this step. Malicious programmes are added to the list of malicious programmes. In contrast, benign programmes are added to the list, which can be used as references in the future. (see Figure 4)

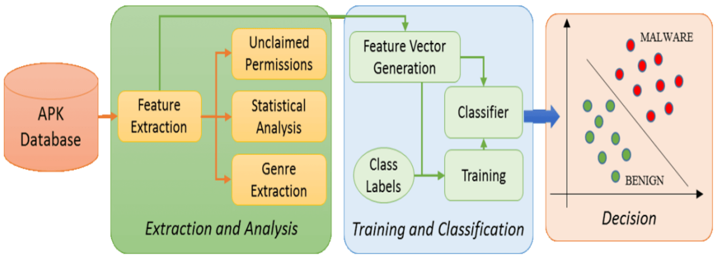


Figure 4 (b) Proposed method

The six components of this study's framework are depicted in the following figures. A component reads apk files and translates the code into.xml and.java classes. The second section is permission, broadcast receivers, and the APIs analysis module. The third module involves translating the extracted features into a binary vector that may be significant for machine learning algorithms and representing each app as a single instance with a binary vector of features and a class label indicating whether the app is benign or dangerous. Finally, we use the binary vectors from the example applications to train three machine learning algorithms and model the classifiers. The bought models determine if an app poses any security risks.

4.1 Dataset

Mahindru and Arvind (2008) obtained the data with legal permission. There are 176 rows in all. Most of these fields are connected to access controls. In what follows, we will look closely at both the permissions and signatures datasets. After a descriptive analysis has been done to remove unnecessary information from a dataset and feature selection techniques have been used to reduce the number of features in the dataset, the dataset is now ready to be processed by a classification algorithm.

```
data_1 = pd.read_csv('dataset_1.csv') # dataset_1 is the permissions data
```

```
print('The permissions data has {} row and {} columns'.format(data_1.shape[0], data_1.shape[1]))
```

4.2 Features extraction

Hundreds of thousands of attributes in a dataset are becoming the norm rather than the exception. In recent years, it has been evident that the resultant machine learning model will become overfit as the number of features increases. To address this problem, we generate a smaller subset of features from the whole set; this approach is commonly used to maintain the same degree of accuracy while reducing the feature set. This study aims to develop a new set of dynamic and static features by picking the most important ones from the existing set and removing the ones that are not as important.

4.3 Features selection

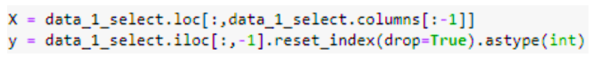
After completing feature extraction and uncovering more features, the next stage is feature selection. Feature selection is essential in selecting features from a pool of newly identified characteristics to enhance accuracy, simplify the model, and reduce overfitting. Researchers have employed various feature classification approaches to identify malicious code in the software. Due to its ability to pick the right attributes needed to build the malware detection models, the feature rank approach is heavily leveraged in this study.

5 Results and discussions

At the end of the descriptive analysis step, a clean dataset was available for further analysis.

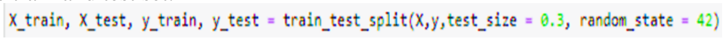
```
data_1_select = pd.read_csv('dataset_1_select.csv')
```

The separation between features and classes is required. The sample's classification as harmful software is the class variable, while the features are independent qualities that help make predictions.



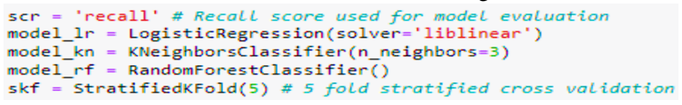
```
X = data_1_select.loc[:,data_1_select.columns[:-1]]
y = data_1_select.iloc[:,-1].reset_index(drop=True).astype(int)
```

The dataset is split into two parts: a training set and a validation set. The training set is used to fine-tune the models, while the test set is used to assess their efficacy, both of which are essential in data mining. I pick a random percentage and split the data in two, allocating 70% to the train and test set.



```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3, random_state = 42)
```

The recall score is one metric I use when comparing and contrasting models. The reason is that an efficient risk-mitigation strategy requires a malware detector capable of identifying as much malware as feasible. The model is run five times, with a new subset of the training set serving as the validation set in each iteration. Using this strategy, I can calculate the mean recall score, which is more accurate than the result of a single model run.



```
scr = 'recall' # Recall score used for model evaluation
model_lr = LogisticRegression(solver='liblinear')
model_kn = KNeighborsClassifier(n_neighbors=3)
model_rf = RandomForestClassifier()
skf = StratifiedKFold(5) # 5 fold stratified cross validation
```

5.1 KNN

As a first step, we will review the results produced by the kNN classifier. Attempt the model with k equal to 3. Figure 4 shows the test set confusion matrix and performance statistics. The model correctly recognized 78% of malware samples from the validation collection. This section will look at how well the kNN algorithm works and compare its results to those of other models. (see Figure 5)

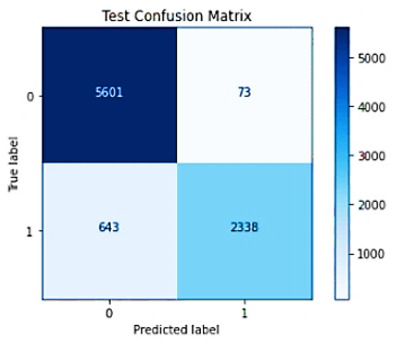


Figure 5 KNN result visualize in confusion matrix

5.2 Logistic regression

Validation data is used to assess the performance of Logistic Regression. Figure 5 displays the confusion matrix and performance indicators for the test set. Logistic regression's recall of 77% is similar to kNN's recall of 78%. This suggests that logistic regression is not the best model to utilize. (see Figure 6)

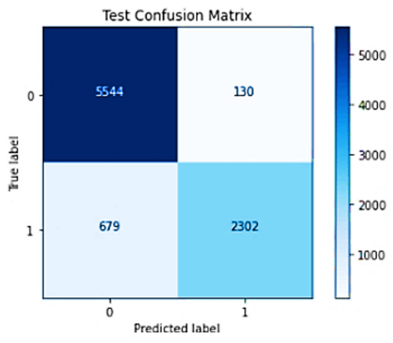


Figure 6 LR result visualize in confusion matrix

5.3 Random forest

We finally agreed on the Random Forest model as our preferred classifier. Thus, with a recall score of 79%, the Random Forest model greatly surpasses the kNN model. Unfortunately, there is no best model for the Permissions data set. Each has a recall rate between 79% and 81%. Even though the Random Forest model is not much better than the others, it is recommended to use permissions data to find malware. (see Figure 7)

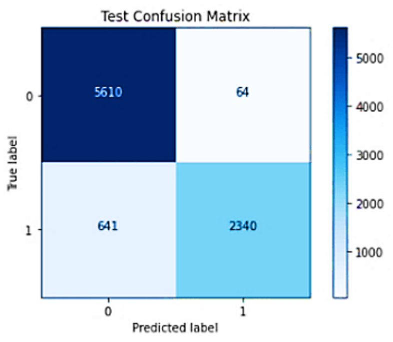


Figure 7 RF result visualize in confusion matrix

These are qualities that we also found to be particularly noteworthy in the permissions-based model. It is safe to assume that an app is malicious if it can access the device's unique identifier and the ability to use the Transact API without user intervention. According to the data presented above, the signatures-based method is superior to the permissions-based method for identifying malware (KNN=78.43%, LR=77.22%, and RF=78.5%) (Table 3). None of the models significantly outperformed the others. Both the Random Forest and kNN algorithms might be helpful in this situation. Figure 7 summarizes the results of testing on both methods for each model.

Table 3 Classifiers results comparisons

		kNN Classifier	Logistic Regression	Random Forest
Permissions based	Precision	96.97%	94.65%	97.34%
	Recall	78.43%	77.22%	78.5%
	FI -Score	86.72	85.05	86.91

Through the statistical analysis, we can deduce that Random Forest (Presidion 97.34%, Recall= 78.5%, F1-Score=86.91%) is the optimal model for the authorization strategy. In contrast, kNN Classifier (Presidion 96.97%, Recall= 78.43%, F1-Score=86.72%) is the second optimal model for the permission-based strategy, and Logistic Regression (Presidion 94.65%, Recall= 77.22%, F1-Score=85.05%) is the third optimal model for the permission-based strategy. We may presume, however, that the Random Forest and kNN Classifier have comparable performance. Using signature-based characteristics to find malware is better than the permissions-based method, which has a much lower recall rate.

6 Conclusion

This research aims to use data mining (ML) techniques to determine whether an Android app is malicious. After testing two approaches, I determined which was the most effective in correcting the issue. I tried three different supervised classification algorithms and settled on the best one. When conducting an inquiry, the permission-based strategy yields accurate results. The Permission-based approach takes advantage of the API Permission mechanisms defined in the application's code. A wide range of coding conventions is used for distinct software projects.

Consequently, signs for telling malicious software from standard software are included in the code. Meanwhile, the Random Forest and the RF Classifier provide the best performance. The project scored 97% on both the precision and recall metrics. The effectiveness of this strategy in identifying malware has been demonstrated at 97%. In conclusion, I developed a methodology for detecting Android malware, a significant concern for governments worldwide. A system like this could be used by law enforcement in UK, USA, UAE, France and other places to protect people from dangerous mobile malware.

Conflicts of interest

The author declares that they have no conflict of interest.

References

Alfalqi, K., Alghamdi, R., & Waqdan, M. (2015). Android platform malware analysis. International Journal of Advanced Computer Science and Applications (IJACSA), 6, 140-146. https://doi.org/10.14569/IJACSA.2015.060120

Alqahtani, E. J., Zagrouba, R., & Almuhaideb, A. (2019). A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms. In 2019 Sixth International Conference on Software Defined Systems (SDS) (pp. 110-117). IEEE. https://doi.org/10.1109/SDS.2019.8768729

Altaher, A. (2016). Classification of android malware applications using feature selection and classification algorithms. VAWKUM Transactions on Computer Sciences, 10(1), 1-5. https://doi.org/10.21015/vtcs.v10i1.412

Anderson, B., Storlie, C., & Lane, T. (2012). Improving malware classification: bridging the static/dynamic gap. In Proceedings of the 5th ACM workshop on Security and artificial intelligence (pp. 3-14). https://doi.org/10.1145/2381896.2381900

Arshad, S., Shah, M. A., Khan, A., & Ahmed, M. (2016). Android malware detection & protection: a survey. International Journal of Advanced Computer Science and Applications, 7(2). https://doi.org/10.14569/IJACSA.2016.070262

Arshad, S., Shah, M. A., Wahid, A., Mehmood, A., Song, H., & Yu, H. (2018). SAMADroid: a novel 3-level hybrid malware detection model for android operating system. IEEE Access, 6, 4321-4339. https://doi.org/10.1109/ACCESS.2018.2792941

Barsiya, T. K., Gyanchandani, M., & Wadhwani, B. (20016). Android malware analysis: A survey. International Journal of Control, Automation, Communication and Systems (IJCACS), 1(1), 35-42. https://doi.org/10.5121/ijcacs.2016.1105

Chang, W. L., Sun, H. M., & Wu, W. (2016). An android behavior-based malware detection method using machine learning. In 2016 IEEE International conference on signal processing, communications and computing (ICSPCC) (pp. 1-4). IEEE. https://doi.org/10.1109/ICSPCC.2016.7753624

Damshenas, M., Dehghantanha, A., & Mahmoud, R. (2013). A survey on malware propagation, analysis, and detection. International Journal of Cyber-Security and Digital Forensics, 2(4), 10-30.

Fatima, A., Maurya, R., Dutta, M. K., Burget, R., & Masek, J. (2019). Android malware detection using genetic algorithm based optimized feature selection and machine learning. In 2019 42nd International conference on telecommunications and signal processing (TSP) (pp. 220-223), IEEE. https://doi.org/10.1109/TSP.2019.8769039

Feizollah, A., Anuar, N. B., Salleh, R., & Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. Digital investigation, 13, 22-37. https://doi.org/10.1016/j.diin.2015.02.001

Firdaus, A., Anuar, N. B., Karim, A., & Razak, M. F. A. (2018). Discovering optimal features using static analysis and a genetic search based method for Android malware detection. Frontiers of Information Technology & Electronic Engineering, 19(6), 712-736. https://doi.org/10.1631/FITEE.1601491

Firdausi, I., Erwin, A., & Nugroho, A. S. (2010, December). Analysis of machine learning techniques used in behavior-based malware detection. In 2010 second international conference on advances in computing, control, and telecommunication technologies (pp. 201-203). IEEE. https://doi.org/10.1109/ACT.2010.33

Hahn, S., Protsenko, M., & Müller, T. (2016). Comparative evaluation of machine learning-based malware detection on android. Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit.

Kim, T., Kang, B., Rho, M., Sezer, S., & Im, E. G. (2018). A multimodal deep learning method for android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14(3), 773-788. https://doi.org/10.1109/TIFS.2018.2866319

Kumaran, M., & Li, W. (2016). Lightweight malware detection based on machine learning algorithms and the android manifest file. In 2016 IEEE MIT Undergraduate Research Technology Conference (URTC) (pp. 1-3). IEEE. https://doi.org/10.1109/URTC.2016.8284090

Leeds, M., & Atkison, T. (2016). Preliminary Results of Applying Machine Learning Algorithms to Android Malware Detection. 2016 International Conference on Computational Science and Computational Intelligence (CSCI), 2016, pp. 1070-1073. https://doi.org/10.1109/CSCI.2016.0204

Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics, 14(7), 3216-3225. https://doi.org/10.1109/TII.2017.2789219

Martín, A., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2017). Evolving deep neural networks architectures for android malware classification. In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 1659-1666). IEEE. https://doi.org/10.1109/CEC.2017.7969501

Sawle, P. D., & Gadicha, A. B. (2014). Analysis of malware detection techniques in android. International Journal of Computer Science and Mobile Computing, 3(3), 176-182.

Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2016). Madam: Effective and efficient behavior-based android malware detection and prevention. IEEE Transactions on Dependable and Secure Computing, 15(1), 83-97.
https://doi.org/10.1109/TDSC.2016.2536605

Schultz, M. G., Eskin, E., Zadok, F., & Stolfo, S. J. (2000). Data mining methods for detection of new malicious executables. In Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001 (pp. 38-49). IEEE.
https://doi.org/10.1109/SECPRI.2001.924286

Siddiqui, M., Wang, M. C., & Lee, J. (2009). Detecting internet worms using data mining techniques. Journal of Systemics, Cybernetics and Informatics, 6(6), 48-53.

Soni, H., Arora, P., & Rajeswari, D. (2020). Malicious Application Detection in Android using Machine Learning. In 2020 International Conference on Communication and Signal Processing (ICCSP) (pp. 0846-0848). IEEE.
https://doi.org/10.1109/ICCSP48568.2020.9182170

Su, X., Zhang, D., Li, W., & Zhao, K. (2016). A deep learning approach to android malware feature learning and detection. In 2016 IEEE Trustcom/BigDataSE/ISPA (pp. 244-251). IEEE.
https://doi.org/10.1109/TrustCom.2016.0070

Tahtaci, B., & Canbay, B. (2020). Android Malware Detection Using Machine Learning. 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), 1-6.
https://doi.org/10.1109/ASYU50717.2020.9259834

Tarar, N., Sharma, S., & Krishna, C. R. (2018). Analysis and Classification of Android Malware using Machine Learning Algorithms. In 2018 3rd International Conference on Inventive Computation Technologies (ICICT) (pp. 738-743). IEEE.
https://doi.org/10.1109/ICICT43934.2018.9034337

Urooj, B., Shah, M. A., Maple, C., Abbasi, M. K., & Riasat, S. (2022). Malware detection: a framework for reverse engineered android applications through machine learning algorithms. IEEE Access, 10, 89031-89050.
https://doi.org/10.1109/ACCESS.2022.3149053

Utku, A., & Doğru, İ. A. (2017). Malware detection system based on machine learning methods for Android operating systems. In 2017 25th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
https://doi.org/10.1109/SIU.2017.7960231

Vanjire, S., & Lakshmi, M. (2021). Behavior-Based Malware Detection System Approach For Mobile Security Using Machine Learning. In 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV) (pp. 1-4). IEEE.
https://doi.org/10.1109/AIMV53313.2021.9671009

Varma, P. R. K., Kumari, V. V., & Kumar, S. S. (2015). A novel rough set attribute reduction based on ant colony optimisation. International Journal of Intelligent systems Technologies and applications, 14(3-4), 330-353.
https://doi.org/10.1504/IJISTA.2015.074333

Varma, P. R. K., Raj, K. P., & Raju, K. S. (2017). Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 294-299). IEEE.
https://doi.org/10.1109/I-SMAC.2017.8058358

Westyarian, W., Rosmansyah, Y., & Dabarsyah, B. (2015). Malware detection on Android smartphones using API class and machine learning. 2015 International Conference on Electrical Engineering and Informatics (ICEEI), 294-297.
https://doi.org/10.1109/ICEEI.2015.7352513

Zhao, K., Zhang, D., Su, X., & Li, W. (2015). Fest: A feature extraction and selection tool for Android malware detection. In 2015 IEEE symposium on computers and communication (ISCC) (pp. 714-720). IEEE.
https://doi.org/10.1109/ISCC.2015.7405598