The feature selection based on AndroidManifest.xml

Jun Guan\(^1\), Bao lei Mao\(^2\)* and Xu Jiang\(^1\)

\(^1\) School of Automation, Northwestern Polytechnical University, Xi’an, Shanxi Province, 710072, PR China
\(^2\) Cooperative Innovation Center of Internet Healthcare, Zhengzhou University, Zhengzhou, Henan Province, 450000, PR China

*Corresponding author’s e-mail: maobaolei@zzu.edu.cn

Abstract. Nowadays, Android malicious applications are rampant, and information security is facing serious challenges. To prevent being decompiled, Android malicious applications often use software packers to protect themselves. After decompilation, the only available file is the AndroidManifest.xml file. Therefore, when detecting shell applications, the researchers can only rely on the features provided by the AndroidManifest.xml file, including permissions and intent filters. How to select effective features from the AndroidManifest.xml file is the key to detect Android malicious applications. This paper proposes the feature selection method for Android malware detection based on AndroidManifest.xml.

1. Introduction

According to the IDC report[1], Android’s smartphone share is expected to hover around 86% in 2020. Such high market share inevitably results in the attack from the hackers. Therefore, the applications need to be inspected before installation. Nevertheless, both malicious applications and benign applications always protect themselves by software packers against decompilation. For malicious application researchers, AndroidManifest.xml is the only file to study. Every application must have an AndroidManifest.xml file, which describes the app’s name, the components of the application, the permissions that the applications need to request in order to access protected resources, and the hardware and software features[2].

In these features, the permissions and intent filters are the most used static feature to detect Android malware. But if all the features are used for detection, it will inevitably lead to dimensional disaster. Feature selection can reduce the complexity of the model and make it easier to interpret. There are three main feature selection methods, namely filter, wrapper, and embedded methods. The features selected by filter methods are based on the scores in various statistical tests for their correlation with the outcome variable. Wrapper methods depend on machine learning algorithms and use cross-validation to train the models on the subset of features. Compared with filter methods, wrapper methods are usually computationally very expensive and make the model more prone to overfitting. Embedded methods combine the qualities of filter and wrapper methods. It’s implemented by algorithms that have their own built-in feature selection methods [3]. Bhattacharya et al. [4] proposed a feature selection method based on permissions and Particle Swarm Optimization (PSO) algorithm to reduce the high dimensional. Zhao et al.[5] applied the mutual information to select permissions and APIs and the result demonstrates the feature set including permissions and APIs is better than the permission set or the APIs alone for detection. Li et al. [6] extracted only 22 significant
permissions as the features of the classifiers to detect Android malware. The above research methods are all for selecting feature subsets to improve the performance of the detection.

Based on these insights, we propose and implement an Android malware detection that uses the static method to extract the 118 permissions frequently requested by applications and intent filters applied by malware from the AndroidManifest.xml file. Then the method uses Information Gain and PSO algorithm to select the effective feature subsets respectively. By comparing the detection results, we choose the optimal feature subset for Android malware detection.

2. Methodology
To select the effective feature subset, our method needs to use IG and PSO. The following is a brief introduction to the two algorithms.

2.1. Information Gain
The information gain algorithm is based on information entropy and measures the impact of this feature on classification. The more information, the more important the characteristics. This information is entropy. In information theory, assuming that the probability of occurrence of each category $x_i$ is $p(x_i)$, the entropy of the classification system is shown in the formula (1).

$$H(Y) = \sum_{i=1}^{k} p(x_i) \log(p(x_i))$$  \hspace{1cm} (1)

The information gain of term $T$ is shown in the formula (2):

$$H(Y|T) = p(t)H(Y|t) + p(\bar{t})H(Y|\bar{t})$$  \hspace{1cm} (2)

The information gain value is defined as the difference between $H(Y)$ and $H(Y|T)$, as shown in the formula (3).

$$IG(T) = H(Y) - H(Y|T)$$

$$= \sum_{i=1}^{k} p(x_i) \log(p(x_i)) + p(t) \sum_{i=1}^{k} p(x_i|t) \log(p(x_i|t))$$

$$+ p(\bar{t}) \sum_{i=1}^{k} p(x_i|\bar{t}) \log(p(x_i|\bar{t}))$$  \hspace{1cm} (3)

The larger the difference, the greater the amount of classification information contained in the $T$, and the more important it is for the classifier. The information gain algorithm belongs to the filtering feature selection method.

2.2. PSO
PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. These candidate solutions are called "particles". Suppose the number of particles is $S$, the search space is $N$-dimensional, and the position of particle $i$ at the number of iterations $t$ is expressed as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$, the search speed is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iN})$. Each particle needs to consider two factors when searching, one is its local best known position $p_{i,pbest}$, the other is the best-known positions $p_{gbest}$ in the search-space. In the search-space, the particle velocity and position are updated to the formula (4).

$$v_{i1}^{t+1} = \omega v_{i1}^t + c_1 \times \text{rand}(\cdot) \times (p_{i,pbest}^t - x_{i1}^t) + c_2 \times \text{rand}(\cdot) \times (p_{gbest}^t - x_{i1}^t)$$

$$x_{i1}^{t+1} = x_{i1}^t + v_{i1}^{t+1}$$  \hspace{1cm} (4)

The PSO algorithm is a wrapper feature selection algorithm. When the number of features is large, the calculation is very time-consuming.
2.3. Evaluation Metrics

The confusion matrix is a tool for analyzing the classifier to distinguish different tuples. As shown in Table 1, the confusion matrix can reflect the "good or bad" of the model in more detail than the evaluation of the model accuracy.

We explain the parameters in Table 1, and define malicious applications as positive tuples and benign applications as negative tuples.

| Table 1 confusion matrix |
|--------------------------|
| Predicted               | Y | N |
| Actual                  |   |   |
| Y                       | TP| TN|
| N                       | FP| FN|

True Positive (TP) is the number of malicious applications correctly identified as malicious application. False Negative (FN) is the number of malicious applications incorrectly identified as benign applications. True Negative (TN) is the number of benign applications correctly identified as benign applications, and False Positive (FP) is the number of benign applications incorrectly identified as malicious applications. The formula for calculating the corresponding classifier evaluation parameters from the confusion matrix are:

\[
\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \tag{5}
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{6}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{7}
\]

\[
\text{F-measure} = \frac{(\alpha^2 + 1) \cdot \text{Precision} \cdot \text{Recall}}{\alpha^2 \cdot (\text{Precision} + \text{Recall})} \tag{8}
\]

When parameter \(\alpha=1\), the F-measure index is F1, as shown in formula (9). When it is higher, the classification effect is better.

\[
F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}
\]

The AUC value refers to the area covered under the ROC curve. When there are multiple ROC curves, the larger the area under the ROC curve and the larger the AUC value, the better the classification effect. Ideally, the model area is 1.

3. Evaluation

3.1. Dataset

The dataset contains 1280 malware from AndroMalShare[7] and MalGenome[8] and 1200 benign applications from the Xiaomi application market. In these applications, some are secondary-repacking malicious applications, so we delete the same samples after decompilation. The features are extracted...
by the program written in Python language. All the experiments about machine learning are completed by WEKA, using the ten-fold cross-validation.

Based on the same training set sample, the J48, KNN, NB, and SVM classifiers with large differences are used to learn the training set separately. Accuracy, F-measure, and AUC are used to evaluate the detection model.

3.2. the features subset selection
First, we use the permissions as the features, and the detection results are shown in Table 2. SVM classifier performs best with an accuracy of 0.943. In the same condition, we use the information from the Intent Filters as the features, the detection results are shown in Table 3. Among these results, the NB classifier performs best, but the accuracy is only 0.873.

| Table 2 | Permission-based detection results |
|---------|-----------------------------------|
| J48     | 0.928                             |
| KNN     | 0.934                             |
| SVM     | 0.943                             |
| NB      | 0.915                             |

| Table 3 | Intent-based detection results |
|---------|--------------------------------|
| J48     | 0.782 |
| KNN     | 0.855 |
| SVM     | 0.836 |
| NB      | 0.873 |

From Table 2 and Table 3, we can see that permission-based detection is more conducive to the detection of malicious applications. But it does not mean that the intent filters are not important. Therefore, we combine the two types of features as the set of full features for detection, and the results are shown in Table 4.

| Table 4 | Detection results based on permissions and Intent |
|---------|-----------------------------------------------|
| J48     | 0.931 |
| KNN     | 0.952 |
| SVM     | 0.952 |
| NB      | 0.923 |

Through the comparison of Table 2 and Table 4, it can be found that no matter which classifier, the detection indicators have been significantly improved. It demonstrates that choosing different types of features can improve the performance of classifiers. But the feature set already contains 132-dimensional features, which affects the efficiency of classifiers. Therefore, we employ the feature selection method to achieve the purpose of improving efficiency and accuracy.

We use the information gain and PSO algorithm respectively to select 54-dimensional features for Android malware detection. The results of the feature subset selected by IG are shown in Table 5. As can be seen from the results, except for the NB classifier, the indicators of other classifiers have been improved. These features are selected by IG including the features from the intent filters. For space reasons, we show the features of the top 20, as shown in Table 6. It can be seen from Table 6 that...
although most of the top 20 are permission features, there are still 4 features from the intent filters, which means they contribute a lot to the Android malware detection.

Table 5 Detection results based on the feature subset selected by IG

|       | Accuracy | Precision | Recall | F-measure | AUC  |
|-------|----------|-----------|--------|-----------|------|
| J48   | 0.944    | 0.944     | 0.944  | 0.944     | 0.951|
| KNN   | 0.961    | 0.963     | 0.961  | 0.961     | 0.976|
| SVM   | 0.959    | 0.961     | 0.959  | 0.959     | 0.958|
| NB    | 0.918    | 0.927     | 0.918  | 0.917     | 0.974|

Table 6 the features of the top 20 selected by IG

| feature                              | IG  |
|--------------------------------------|-----|
| SYSTEM_ALERT_WINDOW                  | 0.52020627 |
| ACCESS_WIFI_STATE                   | 0.25753235 |
| CAMERA                               | 0.469762   |
| VIBRATE                              | 0.24333042 |
| android.net.conn.CONNECTIVITY_CHANGE | 0.4430524  |
| MODIFY_AUDIO_SETTINGS                | 0.23213255 |
| BLUETOOTH                            | 0.35554563 |
| FLASHLIGHT                           | 0.2195303  |
| WRITE_SETTINGS                       | 0.34320919 |
| CHANGE_WIFI_STATE                   | 0.21783869 |
| android.intent.action.USER_PRESENT  | 0.34246605 |
| android.intent.action.SIG_STR        | 0.20235891 |
| RECORD_AUDIO                         | 0.31355958 |
| BLUETOOTH_ADMIN                      | 0.19632864 |
| WAKE_LOCK                            | 0.28092635 |
| ACCESS_COARSE_LOCATION               | 0.19032345 |
| CHANGE_NETWORK_STATE                 | 0.2746238  |
| ACCESS_FINE_LOCATION                 | 0.18954152 |
| MOUNT_UNMOUNT_FILESYSTEMS            | 0.26431345 |
| READ_LOGS                            | 0.18853336 |

The results of the subset selected by PSO are shown in Table 7. From the results, the indicators of all the classifiers have been improved than Table 3. Compared with IG, the feature subset selected by PSO is better and the highest detection accuracy gets 0.966. Therefore, we choose permissions and Intent as features and then use the PSO algorithm to select a feature subset. The results also show that the detection results of the feature subset can be close to or even exceed the set of all features and the key is how to select features.

Table 7 Detection results based on the feature subset selected by PSO

|       | Accuracy | Precision | Recall | F-measure | AUC  |
|-------|----------|-----------|--------|-----------|------|
| J48   | 0.939    | 0.941     | 0.939  | 0.939     | 0.955|
| KNN   | 0.961    | 0.961     | 0.961  | 0.961     | 0.979|
| SVM   | 0.966    | 0.968     | 0.966  | 0.966     | 0.967|
| NB    | 0.956    | 0.960     | 0.956  | 0.956     | 0.982|

4. Conclusion
More and more shell applications make malware detection more difficult. To solve the problem, we proposed the method to extract the features from AndroidManifest.xml and use the PSO algorithm to select the feature subset. Our method not only reduces the dimension of features but also improves the
detection efficiency. Although the method proposed in this paper has achieved good results, it still has some limitations. In future work, we need to add more features from AndroidManifest.xml and further study the algorithms.

Reference
[1] Smartphone Market Share[EB/OL]. https://www.idc.com/promo/smartphone-market-share/os.
[2] Intent and Intent Filter[EB/OL]. https://developer.android.com/guide/components/intents-filters.
[3] Introduction to Feature Selection methods with an example[EB/OL]. https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/.
[4] Bhattacharya A, Goswami R T, Mukherjee K. A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission based detection of Android malwares[J]. International Journal of Machine Learning and Cybernetics, 2019, 10(7): 1893-1907.
[5] Zhao C, Wang C, Zheng W. Android Malware Detection Based on Sensitive Permissions and APIs[J]//Proceedings of the 2nd Security and Privacy in New Computing Environments, Tianjin: IEEE, 2019.
[6] Li J, Sun L, Yan Q, et al. Significant permission identification for machine-learning-based Android malware detection[J]. IEEE Transactions on Industrial Informatics, 2018,14(7), 3216-3225.
[7] AndroMalshare [EB/OL]. http://andromalshare.org:8080/#overview.
[8] Malgenomeproject [EB/OL]. http://www.malgenomeproject.org.