EDMDroid:Ensuring Diversity to improve Android malware detection based on permissions

房浩 朱会娟
江苏大学计算机科学与通信工程学院

三等奖

Abstract

Android system has become the main target of malicious software on account of its open source and popularity. The rapid growth of malicious software, including tariff consumption, privacy theft and remote control, has brought great harm to users. Therefore, the task of detecting malware carries great significance. This article proposes a framework EDMDroid based on Android permission frequency to detect malware. The purpose of this framework is to improve the detection rate by ensuring the diversity of ensemble base classifiers. The realization of diversity is to generate data subsets through random feature subspace and bootstrapping technology, then use non-negative matrix factorization (NMF) technology to generate new data sets on the basis of data subsets. The final prediction results using the integrated strategy voting method come from the integrated predicted results of the model trained by decision tree algorithm. We verified the proposed method on two datasets. The first dataset only extracts the permissions officially defined by Android (OFC dataset). The second dataset contains custom and officially defined permissions (CBA dataset). Experimental results prove that EDMDroid is an effective way for detecting Android malware.

Proposed method

1. FRAMEWORK

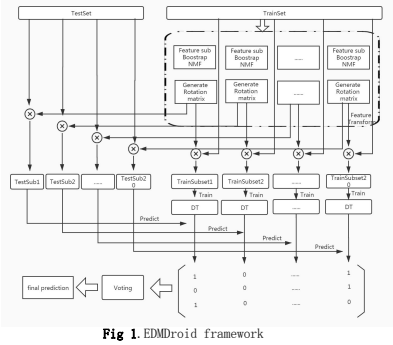


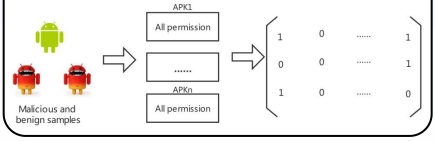
Fig 1. EDMDroid framework

2. Implementation process

In order to achieve the diversity of base classifiers to better detect Android malware, referring to the algorithm of rotating forest, rotating forest is the dimensionality reduction of the data set through Principal Component Analysis (PCA).Based on the characteristics of PCA ,which allowing negative decomposition and linear dimensionality reduction, we use NMF to replace PCA, compared with PCA, NMF has two advantages, the first advantage is non-negativity, which can make data more effective. Of course, the author who proposed NMF believes that non-negativity will cause sparseness and that non-negativity will cause the calculation process to enter partial decomposition, resulting in a reduction in calculation time. The second advantage is the stronger ability of nonlinear fitting. Therefore, this article proposes a new framework EDMDroid, the original data set is five-fold cross-validated, and then a data subset is generated through feature subspace and bootstrapping sample technology, and then a new data subset is formed after each data subset is subjected to non-negative matrix factorization. Since ensemble learning is to combine multiple weak learners through a certain integration strategy to get a better learner, with the widespread application of ensemble learning, this paper introduces ensemble learning methods in order to improve the performance of Android malware detection. After the new data subset is formed, the model is trained through the decision tree method, and the obtained prediction results are integrated by voting to obtain the final prediction results.

3. Feature Extraction

Android uses permissions to restrict applications access and operation to sensitive resources. Therefore, we use the decompilation tool GDA to decompile the APK file, GDA is a new C++-based reverse analysis tool for Android, can extract the requested permissions for samples. But the permissions we extracted are not represented by binary numbers (0 and 1 respectively indicate whether the permission is requested),Expressed by frequency. For example: In the Manifest file, the user-permission tag declares the READ_LOGS permission, and other components also declare the permission, so the frequency of the permission is the number of times it appears in the Manifest file. Using the frequency of permissions as a feature can highlight the importance of permissions. For example: INTERNET permission almost every application requests, but the frequency of each application request is different. If it is purely expressed in binary, each application requests permissions , which is no difference, and the frequency of the permissions can make the difference of characteristics more obvious. Therefore, after decompiling the application, we obtain the Manifest file of the application, analyze the Manifest file, and extract the permissions declared by the user-permission. At the same time, all the permissions requested by the components in the Manifest are also extracted. Finally, we extracted the frequency of all permissions as features to detect Android malware



4. Constructing the training set for learner

Algorithm 1 Constructing the training set for learner

Input:
X: training samples
Y:lable set
M:number of feature in each subset
Function:
1. F←feature set in X
2. F_{num}←size of feature set F
3. S←the number of subsets
4. Randomly split the feature set F into S subsets $F_j (1 \le j \le S)$
5. R←zero(F_{num}*F_{num})
6. For j=1:S
   F_j←the feature in j_{th} subset
7. X_j←get the corresponding columns of F_j in X to build a training set
8. X_j' ←draw a bootstrap from X_j(with 75% of sample size of X_j)
9. D_j←run NMF on X_j' to obtain the coefficients $r_j^{(1)},\ldots,r_j^{M}$
10. End for
11. R= $r_1^{(1)},\ldots,r_S^{M}$
12. R' ←rearrange the columns of R with respect to the order of features in F to construct a new rotation matrix
13. N_{train}←generate a new training set with(X*R')
14. Return N_{train}

Experiment

1.Dataset

1)OFA data set
We first collected 166 permissions from the documentation of Android developers as the official standard for defining permissions. As long as it is officially defined, we will keep it, and delete other permissions, and finally get a data set with only officially defined permissions. This data set has 145 features (permissions)
2)CBA data set
We take into account the permissions defined by the APP itself. If there is no call between applications, the permission will only be requested once. This will cause the redundancy of features to affect the classification effect, Therefore, we delete the permission with a frequency of 1, and finally get a data set combining the official permissions and the permissions defined by the APP itself. This data set has 367 features.
Both datasets have 1994 malicious samples and 841 benign samples

2.Experiment Result

Dataset	Accuracy	Precision	Recall	F1-score	Specificity	MCC	AUC	FPR
OFA	0.8437	0.8878	0.8906	0.8890	73.18	0.6248	0.8187	0.2681
CBA	0.8613	0.9036	0.8985	0.9010	77.28	0.6691	0.8320	0.2271

The results of the first set of experiments are in the OFA data set and the CBA data set.The experimental results show that the official permissions and the permissions defined by the APP can indeed improve the performance of malware detection.

Dataset	Accuracy	Precision	Recall	F1-score	Specificity	MCC	AUC	FPR
OFA PCA	0.8906	0.8988	0.9510	0.9241	74.73	0.7319	0.8577	0.2526
OFA NMF	0.8959	0.9066	0.9496	0.9275	76.83	0.7455	0.8636	0.2316
CBA PCA	0.9022	0.9108	0.9543	0.9320	77.88	0.7604	0.8605	0.2211
CBA NMF	0.9068	0.9130	0.9589	0.9354	78.29	0.7713	0.8623	0.2170

The second set of experiments is to use the EDMDroid framework for the OFA and CBA data sets. It can be seen from TABLE that the results obtained in the CBA data set are better than the results in the OFA data set.so prove again that the combination of officially defined permissions and APP's own defined permissions can indeed improve the performance of malware detection. on the same data set, the effect of the subset processed by the NMF method is better than the subset processed by the PCA method. It proves that the EDMDroid framework is an effective method for malware detection.

Conclusion

To resist Android malware, we collected the most recent samples to form a new data set and extract the frequency of sample request permission as a feature. We proposed the EDMDroid framework, which increases the diversity of base classifiers by using random feature subspace and bootstrapping sample technology. At the same time, non-negative matrix factorization (NMF) is performed on each subset. The new data set was obtained by multiplying the rotation matrix of each subset with the training set. In this paper, experiments have proved that the official defined permissions and user-defined permissions can better detect Android malware.

Main References

[1] A. Arora, S.K. Peddoju, M. Conti, PermPair: Android Malware Detection Using Permission Pairs, IEEE Transactions on Information Forensics and Security 15 (2020) 1968-1982.
[2] M.N.R.S.S.H.V.S.P. Thangaraj, Comparative Analysis of Feature Selection Methods and Machine Learning Algorithms in Permission based Android Malware Detection, (2018).
[3] V.G.S.G.S.M.S.G.V.L.M. Conti, AndroTaint: An Efficient Android Malware Detection Framework using Dynamic Taint Analysis, ISEA ASIA SECURITY AND PRIVACY (ISEASP) (2017).
[4] H. Cai, N. Meng, B. Ryder, D. Yao, DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling, IEEE Transactions on Information Forensics and Security 14(6) (2019) 1455-1470.
[5] Z.L.R.W.X.J.S.Z.C. Wu, Classifying Android Malware with Dynamic Behavior Dependency Graphs, Trustcom/BigDataSE/ISPA 2016 IEEE Trustcom/BigDataSE/ISPA (2016).

Contract

Author：Hao Fang
Phone：15751011553
E-mail：1287947452@qq.com