SUIP: An Android malware detection method based on data flow features

SUIP: An Android malware detection method based on data flow features

X R Chen[1], S S Shi[1], C L Xie[1], Z Yang[1], Y J Guo[1], Y Fang[1] and W P Wen[1]

[1]School of Software and Microelectronics, Peking University, Beijing 102600, China

E-mail: weipingwen@pku.edu.cn

Abstract. Currently static detection is the most commonly used in Android malware detection. Among them, the extraction of various features is particularly important. In analysing the data flow features of applications, researchers usually use taint analysis method to extract. However, this method lack intermediate process features. So in this paper, we analyse the features of Android components to obtain application data transfer features for complementing the application data flow features and build a more complete combination of data flow features. Based on this, we propose a new Android malicious application detection method—SUIP. This method complements the missing features based on taint analysis, and combines the LightGBM algorithm to build a detection model. Finally, we use the sample set in Virusshare for experiments. Compared with the traditional static detection method of Android malicious code, the result shows that our detection method has a high detection accuracy of 98.50%.

1. Introduction

With the continuous development of the Internet, smart phones have gradually become an indispensable part of people's daily lives. However, security threats have also followed. There are many types of malware for Android like viruses, worms, Trojan horses, hijacks, and backdoors. Except those, other malicious software such as spyware, ransomware, adware and tracking software have appeared recently [1]. The number of malicious programs targeting Android phones has grown rapidly in recent years. According to 360's "China's Mobile Phone Security Report in 2018", the 360 Internet Security Center intercepted 4.342 million malicious smartphone samples during 2018, with an average growth rate of about 12,000 per day. In recent years, the center has detected about 110 million mobile malware infections. How to efficiently and accurately detect Android malware has become a hot potato in mobile Internet security research in recent years.

In terms of Android malicious code detection, various technologies have been proposed at home and abroad in recent years, which can be divided into static detection technology and dynamic detection technology according to whether the detection requires program execution. Because of the low efficiency and the difficulty of analyzing all execution paths completely by dynamic analysis, static detection technology is more commonly used in the industry. In static detection technology, researchers extract various features of Android applications and use combinations of various features to train classification models to achieve the purpose of detection. Among them, for the data flow feature extraction of Android applications, researchers have developed mature method based on taint analysis. However, because of the drawbacks of the taint analysis itself, we found that if the results of the taint analysis are directly used as the data flow features of the Android application, the intermediate data flow features, which are called the intermediate data transfer features, will be

missing. It will affect the integrity of the data flow and bring a higher false positive rate for malware detection.

Based on this, we analyze the Android data flow transmission. The existing malicious code detection method based on taint analysis in Android platform is difficult to analyze the intermediate process of data transmission. To solve the problem, we propose to extract and analyze the data transmission between the components of the Android application and between the applications, which makes the data flow features more complete and more accurate. Compared with the existing methods, the accuracy of our model is significantly improved.

To summarize, we make the following contributions:

(1) We propose the completion of the data flow features of Android applications, and use more convincing analysis and demonstration to illustrate that combining the results of taint analysis with the features of data transfer between Android applications and components can greatly improve the detection effect;

(2) We develop a set of extraction tools for Android application data flow features, which mainly include the extraction of data flow features and data flow transfer features;

(3) Based on the data flow features mentioned above, we propose an Android malicious application detection method—SUIP, whose detection effect is significantly improved compared with the previous detection method.

The rest of this paper is organized as follows:

In chapter 2, we introduce the current static analysis methods in Android malicious code detection, as well as the evolution and status quo of detection methods based on data flow features. In chapter 3, we analyze the Android data flow features and propose a complementary method towards data flow characteristics. In chapter 4, we introduce the detection model design based on the data flow features of the completion. In Chapter 5, we analyze the experiments and results of our detection model. In Chapter 6, we summarize the work and discuss the shortcomings and follow-up research directions.

2. Related work

In the research of Android malicious application detection, researchers continue to design new detection methods and models. In recent years, with the rise of machine learning technology, detection based on machine learning has become an important research direction. The general idea of machine learning malicious code detection technology is: collect a large number of Android malicious and non-malicious application samples, extract features from the samples, filter them to form feature vectors, and then input the feature vectors into the machine learning algorithm to obtain the machine learning model, and finally use the machine learning model to detect Android applications. It can be seen that in the detection algorithms based on machine learning, the most important thing is the extraction of features, and according to whether the program needs to be executed during feature extraction, it can be divided into static detection technology and dynamic detection technology.

In the research of machine learning detection technology based on static feature analysis, researchers extract various features of Android applications and use combinations of various features to train classification models for detection. In 2014, DREBIN proposed by Arp, D. et al. [2] collects static features of Android applications such as permissions, application components, suspicious API calls, and network addresses, and forms a joint feature vector based on these features, which is then put into the machine learning model for training. After training, a lightweight model which can automatically recognize some typical malicious applications is completed. In 2015, Avdiienko et al. [3] found that the sensitive data flow of malicious applications and security applications are quite different. Based on this, Avdiienko uses Flowdroid for taint analysis, extracts the "normal data flow set" in the Android benign program, and calculates the geometric difference between the sensitive data flow and the "normal data flow" of the application during program detection. The difference is used as a feature to train the detection model to form a MUDFlow tool. On the basis of MUDFlow, Pengbin Feng et al. [4] compared the key data flows in benign and malicious applications and found that there are some common data flows in these two applications. These data flows affect the model accuracy during

model training, at the same time, there is a relatively large difference between the frequency of some key data flow in benign and malicious applications within the two programs. If these data flow are used as features for model training, the accuracy rate of model detection can be greatly improved. Pengbin Feng thus designed a new feature selection algorithm CflowSel, and developed the detection tool SCDFlow. In 2017, F. Idrees et al. [5] used integrated learning algorithms in Android malicious program detection, developed the detection tool PinDroid, it used permissions and Intent component information as classification features, and used integrated learning algorithms to optimize detection results. In 2018, Liu Qiyuan and others proposed an implicit data flow analysis method for the problem that common tools such as Flowdroid cannot analyze [6]. Wang Lei and others proposed a Flowdroid's multi-source stain analysis tool [7] based on the high false positive rate of static taint analysis.

Flowdroid [8] is a commonly used static stain analysis tool, but it also has some shortcomings. Flowdroid cannot obtain the exact execution sequence between Android components. Therefore, if the components are executed in random sequence, this will lead to redundant analysis and false positives in the Flowdroid results [9], and Flowdroid can only perform in-app analysis but doesn't analyze messages passing between applications. Based on the above situation, we analyze the data flow between Android applications and components, and strengthen the data flow and control flow analysis by adding analysis of action and URI features, and complement the intermediate data flow features to make up for the shortcomings of Flowdroid.

3. Android data flow analysis and feature completion

Although Flowdroid performs well in Android application data flow analysis, there are still some drawbacks in using it for malicious program detection. First of all, Flowdroid does not support the analysis of multi-threaded programs; secondly, Flowdroid cannot understand the true execution sequences of components, and can only assume that threads are executed in any linear sequence, and different components are executed in any sequence; finally, Flowdroid's output results focus on describing the origin and destination of tainted data, it lacks a description of the middle transmission process of the data flow. However, the different dissemination of tainted data between components or applications can result in different behaviours of malicious programs. In the application of machine learning, the lack of analysis of these features will lead to the accuracy of the model not reaching the expected effect.

The data flow characteristics extracted by Flowdroid focus on the origin and destination of the tainted data, but it cannot fully describe the changes and behaviors of the tainted data in the transmission path. For example, the application may send tainted data to any component or pass it to other applications. The target component, target application, as well as the impact and behavior on the target of these tainted data are unknown. It is difficult to describe the Android data flow characteristics completely only using the Source-Sink pair extracted by Flowdroid. In order to make the data flow features extracted by taint analysis more accurate, the intermediate transmission process of the data flow needs to be complemented. By analyzing the Android system mechanism, we found that the missing data flow intermediate features can be complemented and supplemented by the Android system communication mechanism. Among them, the Intent and Content-Provider in the communication mechanism can respectively describe data transfer between Android application components and between applications.

3.1. Intent

Intent is an important way for the components in the Android program to interact. It can not only indicate the actions to be performed by the components, but also transfer data between components [10]. In Android applications, Intent is the communication hub of the four major components. In the process of taint analysis on Flowdroid, Intent is only seen as a way to transfer data. But in fact it also contains an action parameter, which affects the program behavior indicated by the Intent. Intent can be used to start Activity, it can also be used to start Service and send broadcast. Therefore, we choose the

action parameter of Intent to describe the data transfer intention between Android application components.

3.2. Content-provider

Content-Provider is a data sharing mechanism provided by the Android system. It provides a complete mechanism that allows one program to access the data of another program. It is a standard way for the Android system to share data across programs. The ContentResovler class of this component provides an interface for applications to access shared data. ContentResovler uses a universal identifier (URI) as a parameter to add, delete, modify, and check shared data. In the process of taint analysis on Flowdroid, ContentResovler is also only seen as a way to transfer data, but different URIs represent different application databases and different program behaviors. Therefore, we use URI to describe the data transfer intent between applications.

Traditional Android malicious code detection methods based on data flow characteristics often only use the origin and origin of the taint as features. The analysis results are more inclined to reflect whether this data flow transmission behavior is sensitive, but in actual scenarios, some benign applications, such as System applications will also do this, which leads to a higher false alarm rate simply using the Source-Sink pair in the Flowdroid result as a feature. Intent and Content-Provider are both important components of the Android system's communication mechanism. Intent's action parameter can describe the data transfer intention between application components, and the URI can describe the data transfer intention between applications. While using the data flow extracted by Flowdroid as a feature to detect Android malicious code, if we can use the action parameter and URI to supplement the features of the data flow, we can construct a more complete data flow feature, which will help improve the accuracy of detection.

In addition, in the analysis of the communication mechanism of the Android system, we found that malicious applications need to use sensitive permissions to control program behavior, so we also use permissions as a type of feature when building a machine learning model.

In previous work, researchers often used various Android application features to mix, or used taint analysis data flow alone as features for malicious code identification, which resulted in poor detection results and weak interpretability. We fully analyzed Flowdroid's data flow feature defects and the Android system data flow transmission mechanism, and selected Source-Sink, Intent action parameters and URI as feature combinations. This is also what researchers rarely pay attention to when using taint analysis for Android malicious application detection research.

4. Design of inspection model

Based on the above analysis, we use the above method to supplement the existing problems in Flowdroid taint analysis, and propose SUIP, an Android malicious application detection method based on data flow characteristics.

Figure 1 shows the workflow of the detection method. It is mainly divided into four modules: data collection and preprocessing, feature extraction, feature processing and classification detection. In the classification and detection module, we conducted experimental tests on a variety of machine learning models and found that the LightGBM algorithm is the best. Therefore, we choose this algorithm as the classification algorithm. The specific experimental data will be introduced in the experimental part.

4.1. Data collection and preprocessing

We collect samples of Android apps from Google Play and VirusShare. Malicious Android apps come from the VirusShare website, and benign apps are obtained from Google Play using crawlers. In order to reduce the interference of individual benign applications with suspicious behavior on the model, we preprocessed the benign application data set directly obtained from the app store-to exclude suspicious applications through the online suspicious file analysis service provided by the VirusTotal platform. The benign data sets we use are all with VirusTotal detection results of zero threat, which avoids malicious applications in the benign APK collection.

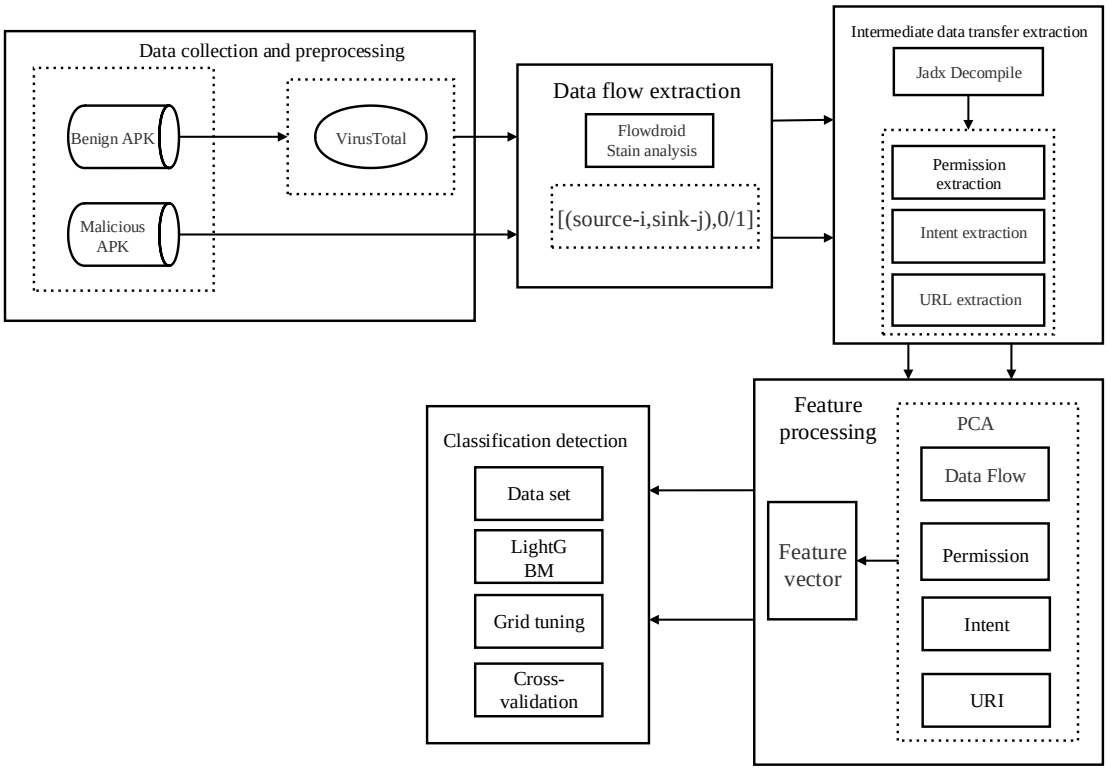

Figure 1. Workflow of SUIP.

4.2. Feature extraction

The feature extraction module is divided into two parts. In the first part, we use FlowDroid to extract the data flow characteristics of the Android application. In the second part, we perform batch decompilation of APKs, and extract the permissions and Intent and ContentResolver features of Android applications. In this work, we implemented an integrated Android feature extraction tool [https://github.com/gyjstu/Apk-Detector]. This tool can effectively extract data flow characteristics, intermediate data transfer characteristics, and permission characteristics of Android applications.

4.2.1. Data flow. In this module, we use Flowdroid to analyze the data flow of the application, and use the APK-sensitive Sourse-Sink pair (176 sensitive sources and 227 sensitive sinks) officially provided by Flowdroid as the basis for taint analysis. Through taint analysis, the APK Behaviors are extracted in the form of Sourse-Sink pairs to represent their risk flow information.

*4.2.2. Intermediate data transfer and Permission.*We introduced the action constants in the Intent for data transfer between components and the URI parameter in the Content-Resolver for data transfer between applications to make up for the shortcomings of insufficient data flow characteristics, and introduced permission features to make up for the problem of data transfer permission dependence. When extracting these features, we first use Jadx to parse and decompile APK files, and then filter out URI constants, Action constants, and permission constants through regular extraction.

4.3. Feature processing

The features we obtain can be divided into two parts, one is the data flow feature Source-Sink pair, the other is the data transfer feature including Action constants, data transfer feature URI constants between applications, and permission constants. For data flow characteristics, Flowdroid officially provides a complete Source-Sink document, which lists all possible sensitive Source functions and Sink functions. These functions include 176 Source functions and 227 Sink functions, for a total of

39,952 possible Source-Sink pairs. For the three parts of the data transfer feature, the official Android documentation lists all possible values, including 215 URI constants, 215 Action constants, and 167 Permission constants, totaling 597 features. First, we need to use the above features to construct a feature vector, and then reduce the dimensionality of the data.

4.3.1. Construct feature vector. This part converts the features extracted in the previous part into feature vectors. The values of the features that exist in the APK file are marked as 1, and the unacquired features are marked as 0. Since the number of features in these two parts is relatively large, the resulting matrix will be relatively sparse, so we need to reduce the data dimension of them.

4.3.2. Data dimensionality reduction. This module uses the principal component analysis algorithm PCA to reduce the dimension of the features. We tried two dimensionality reduction methods for the above two features respectively: 1. First merge and then perform PCA dimensionality reduction; 2. First perform PCA dimensionality reduction separately and then merge the results. Through the comparison of experimental effects, we found that the effect of the latter is significantly better than the former. Next, we continued to try through the double-layer loop, and found that when the data transfer features are reduced from 597 to 70, and the data flow features are reduced from 39,952 to 225, the accuracy rate obtained under the same machine learning model is the highest.

4.4. Classification detection

After feature processing, the next step is model training. We first divide all processed data sets and feature vectors, and use random sampling to select the training set and the test set at a ratio of 3:1. Then we use the LightGBM algorithm for training, continuously adjust the parameters of the model through grid tuning, and perform 5-fold cross-validation on the training set to find the hyperparameters that make the highest accuracy. Finally, we use the test set to test and get the final model accuracy.

5. Experimental results and analysis

The experimental test data consists of 8310 Android applications, including 5100 normal samples and 3210 malicious samples. During training, we divide 75% of the samples into the training set and 25% into the test set to train the machine learning model.

5.1. Comparison of machine learning models

When choosing a machine learning algorithm, we compared a variety of machine learning models and selected the best algorithm for SUIP. The results are shown in Table 1. In order to facilitate intuitive comparison, only the accuracy of the models are compared here to represent the pros and cons of the models.

Table 1. Comparison of machine learning model accuracy.

Machine learning model	Accuracy
LightGBM	98.50%
SVM	95.02%
GradientBoost	90.60%
AdaBoost	90.06%
ExtraTreesClassifier	86.18%
Random forest	86.19%
KNN	86.74%

Logistic regression	92.81%
Neural Networks	92.83%

5.2. The detection effect of SUIP

After comparing the machine learning models, we found that the performance of the LightGBM algorithm is the best. Therefore, we choose the LightGBM algorithm to train the malicious code detection model, and the final detection results are shown in Table 2.

Table 2. SUIP final model results.

Accuracy	Precision	False alarm rate	Recall	F1
98.50%	98.67%	0.78%	95.92%	94.32%

5.3. Compared with other static detection methods

In order to verify the detection effect of the detection system in this paper, we use the same training set and test set to compare with other methods. The comparison results are shown in Table 3.

Table 3. Compared with other static detection methods.

Method	Features	Accuracy	Precision	False alarm rate
AndroDialysis [11]	Authority, Intent	95.50%	91.00%	5.28%
DroidDet [12]	Permissions,Sensitive APIs,System events	88.39%	88.40%	3.16%
SigPID [10]	Permissions	93.62%	91.40%	8.64%
FlowDect [14]	Sensitive data flow	82.20%	88.26%	2.11%
SUIP	Sensitive data flow, Data transfer features	98.50%	98.67%	0.78%

It can be seen from the experimental results that SUIP has a better detection effect than the existing static detection technology for Android applications. This also shows that the use of inter-component data transfer features and inter-application data transfer features to supplement data flow features is reasonable and effective. The detection model proposed in this paper has higher identification capabilities and accuracy for Android malicious applications.

6. Summary and discussion

In this article, we extract data transfer characteristics between Android applications and components, and enhance the existing Android malicious code detection method based on taint analysis by making the data flow characteristics more complete and accurate. The SUIP we implemented has an accuracy rate of 98.50% for Android malicious code detection, which is better than existing static Android malicious code detection methods. This result can prove that the use of Android data transfer features can complement the data flow features and improve the accuracy of the data flow features in detecting Android malicious code. This can also inspire subsequent research on Android application detection methods based on data flow characteristics.

In addition, we found in our experiments that protection mechanisms such as confusion and shelling can lead to incomplete feature extraction and affect detection accuracy. Therefore, in order to enable the detection model to support Android malicious code that uses a complex obfuscation protection mechanism, we will consider the detection of Android application protection measures and

targeted feature extraction in subsequent research to improve the effectiveness of the detection model. On the other hand, we believe that how to automatically unpack the APK before decompilation will be an in-depth direction in the field of Android malicious application detection in the future, and it will also be a very meaningful work.

Acknowledgments
Thanks to my mentor and senior. In the thesis work, Professor Wen Weiping and seniors Li Jingwei and Liu Hongyi provided a lot of help and guidance. At the same time, I am very grateful to Teacher Reez who helped us in writing.

References
[1] Sihan Q 2016 Advances in Android safety research *Journal of Software*. 27(1)45−71
[2] Arp D, et al. 2014 Drebin: Effective and explainable detection of android malware in your pocket. *Ndss.* 14
[3] Avdiienko V, et al. 2015 Mining apps for abnormal usage of sensitive data *IEEE/ACM 37th IEEE International Conference on Software Engineering.* 1 pp 426-436
[4] Qiyuan L, Jian J and Hongsheng C 2018 Ontology Model of Implicit Information Flow Detection *Computer application.* 38(01) pp61-66
[5] Lei W, Qing Z and Dongjie H 2019 A Multi-source Smudge Analysis Technique for Android Application Privacy Disclosure Detection *Journal of Software.* 30(02) pp 21-40
[6] Jiwei Y, Mingsu L and Qiong L 2017 Analysis and comparison of privacy leak detection tools based on Android platform *Computer science.* 44(010) pp127-133
[7] Feng P, Ma J and Sun C 2017 Selecting critical data flows in Android applications for abnormal behavior detection *Mobile Information Systems.* 2017
[8] Arzt S, Rasthofer S, Fritz C, et al. 2014 Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps *Acm Sigplan Notices*. 49(6) pp 259-269
[9] Idrees F, Rajarajan M, Conti M, et al. 2017 PIndroid: A novel Android malware detection system using ensemble learning methods *Computers & Security*. 68 pp 36-46
[10] Reps T, Horwitz S and Sagiv M 1995 Precise interprocedural dataflow analysis via graph reachability *Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages.* pp 49-61
[11] Feizollah A, Anuar N B, Salleh R, et al 2017 Androdialysis: Analysis of android intent effectiveness in malware detection *Computers & Security*. 65 pp 121-134
[12] Hui-Juan Z, et al. 2018 DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model *Neurocomputing.* 272 pp 638-646.
[13] Jin L, et al. 2018 Significant permission identification for machine-learning-based android malware detection *IEEE Transactions on Industrial Informatics*. 14.7 pp 3216-3225
[14] Dali Z, et al. 2019 Detection of Android Malicious Application Based on Data Flow Depth Learning Algorithm *Information Security Journal.* 4(02) pp 57-72