Combining Static Permissions and Dynamic Packet Analysis to Improve Android Malware Detection

Yung-Ching Shyong[1], Tzung-Han Jeng[§ 1,2], Yi-Ming Chen[2]
Department of Information Management, National Central University, Taoyuan, ROC 1
Information and Communication Security Laboratory, Chunghwa Telecommunication Labs, Taoyuan, ROC2
e-mail: tzunghan@cht.com.tw

***Abstract*—Nowadays Android smart mobile devices have become the main target of malware developers, so detecting and preventing Android malware has become an important issue of information security. Therefore, this paper proposes an Android application classification system that combines static permissions and dynamic packet analysis. This system first obtains the static information of Android applications through static analysis, classifies the applications as benign or malicious through machine learning, and avoids excessive dynamic data collection time by filtering out benign applications. Then in the dynamic analysis stage, the malware's network traffic is used to extract multiple types of features, and then machine learning is used to achieve the malware family classification. The experimental results showed that the accuracy rate of the static model for malicious and benign classification was 98.86%. On the other hand, the accuracy of the dynamic model proposed in this paper for family classification of applications is 96%, which is better than 94.33% of DroidClassifier [1]. The final experiment confirmed that the system proposed in this paper can not only save 52.5% of dynamic data collection time but also improve the accuracy of Android application family classification.**

Keywords-Android malware detection; static analysis; dynamic analysis; application family classification

I. INTRODUCTION

According to the statistics and analysis of the IDC (international data corporation) website [2], Android has maintained a market share of more than 85% in the smartphone market in recent years. Moreover, Android's smartphone share is expected to increase throughout the forecast period (2019~2023). On the other hand, Android malware samples continue to grow in 2018, and Android malware variants grew 31% in a year and now number close to 20 million [3]. As described in Nokia's Threat Intelligence Report 2019 [3], among smartphones, Android devices are the most commonly targeted by malware. In the mobile network, the report found that 47% of their malware samples aimed Android, whereas only 35.82% aimed Windows. In view of the huge increase in the number of Android smartphone and Android malware, the topic of detecting and preventing malicious attacks needs to be taken seriously.

Analysis methods in the field of mobile security research are mainly divided into two types: (1) Static analysis: decompile APK files without executing applications, and analyze existing code in files, such as application permissions and opcode. (2) Dynamic analysis: execute the application and collect the behaviors it generates, such as: system call, dynamic code loading, network traffic, establish the behavior characteristics of the application and further analysis. Traditionally, the analysis of malware requires a lot of manpower and time. The analysis process usually requires the experience of information security experts for malware analysis. However, the number of malware variants has greatly exceeded the ability of human analysis, but the knowledge of information security experts cannot be passed on as quickly as malware. Therefore, it is important to construct a system that can automatically analyze Android applications and classify them into malware families.

Furthermore, one of the most commonly used technologies for mobile devices to communicate is the Internet, CISCO reports that mobile data traffic will grow at a CAGR of 46 percent from 2017 to 2022 [4]. As mobile network traffic grows, the feasibility of collecting network traffic as a dataset to detect mobile malware is also increasing. Unfortunately, [5] confirms that each application takes an average of five minutes to record the malicious network behavior of malware, which shows that data collection in dynamic network analysis is a time-consuming phase. In addition, many studies use only a single type of network protocol packet (TCP packet or HTTP packet) to detect malware [5-7], but do not use multiple network protocols to detect malware at the same time. Finally, the malware family has the same intentional behavior pattern, and this classification can help information security engineer to understand the behavior pattern of malware, take or establish similar defensive protection measures, or pre-suspend the execution of the malware in advance, so as to achieve the purpose of intrusion prevention. However, many studies only classify applications as benign or malicious [7-9], but do not further classify them into malware families.

With the above description, three problems in analyzing Android malware are summarized:

1) Dynamic analysis is very time consuming in the data collection phase.

2) Many malware network traffic analyses are limited to extracting TCP or HTTP network packet features. However, many protocols in the entire network traffic can observe malware features, but most researches do not use many network protocols to detect malicious programs.

978-1-7281-5800-6/20/$31.00 ©2020 IEEE

§ Corresponding author.

3) Many studies only classify applications as benign or malicious, but do not further classify them into malware families. This classification method cannot provide enough malicious behavior information to information security engineers, which makes it difficult to establish corresponding prevention strategies.

Based on the above motivations, this paper provides an Android malware detection system that can solve the above problems and achieve the following purposes:

1) Use static analysis to filter out malware with risk before dynamic analysis. This approach can greatly reduce the amount of dynamic data collection for applications in dynamic analysis to help speed up the entire dynamic analysis process.

2) This paper extracts multiple types of network packet features from the execution activities of Android applications for analysis, which improves the accuracy of detection.

3) Classify malware families. Because different malware families have different malicious behaviors, the classification results of this paper will help information security engineers to establish defense strategies.

The rest of the paper is organized as follows: Section II discusses the related work. Section III introduces our architecture and methodology, Section IV presents the experiments and results analysis. In the section V, we summarize the contribution of this paper and provide the direction for future development.

II. Background and Related Work

A. Related Work of Static Analysis

Currently, many of the Android application security detection methods can disassemble their APK files and extract important information from the code without waiting for the application to execute. Many of these studies use the application's permission information as a research goal for static analysis. PUMA [10] extracts the uses-permission and uses-feature information declared by the application for classification, and the classification accuracy is 86.41%. [11] used the same method to classify the Drebin dataset and found that the accuracy reached 96.05%. [12] uses the application permissions to create a two-stage application detection system. The first step is to extract the declared requested permissions information from the application's Manifest.xml file for analysis. The second stage is to use APKtools to decompile the APK file to get the smali code of the application. If information such as API calls related to declared permissions appear in the smali code, consider this declared permission is the actual used permissions of the application, and the used permissions is regarded as a feature of malware detection. The detection accuracy of the method proposed in [12] is 98.6%. In [12], it is also mentioned that the permission-based detection method is more suitable as a preliminary filtering mechanism for the malware classification system.

B. Related Work of Network Traffic Analysis

Researches on TCP packet analysis in network traffic include [8] and [13]. [8] collecting the application's network traffic and establishing a .pcap file to extract the application's total uplink traffic, total downlink traffic, and traffic interval as its detection features. [13] incorporates the source IP and destination IP of the TCP packet into the detection feature. In addition, there are also researches on malicious application detection using HTTP packets. [14] uses HTTP traffic generated by Android malware when communicating with remote malicious servers, and establishes malware family signature through coarse and fine grained clustering. This signature can be used for malware detection. This signature can be used to detect which cluster family the malware belongs to. Moreover, TextDroid splits the transmission content of the HTTP request packet by special characters and then creates an n-gram sequence and then inputs it into the SVM classifier for detection. However, the detection rate of this HTTP semantic feature is only 76.99% [9]. Also, DroidClassifier extracts five kinds of semantic information from the contents of the HTTP request packet and stores it in the malware database to calculate the weight of each HTTP request packet to set the detection threshold [1]. TrafficAV extracts traffic features of TCP and HTTP and puts them into C4.5 decision tree respectively for accuracy comparison. However, TrafficAV does not combine the features of TCP and HTTP network traffic for machine learning [7].

[5] found that more than 70% of the samples in the Drebin [6] malware dataset were able to collect malicious network traffic behavior within five minutes. By observing the network traffic data and characteristics of the malware family, it is found that samples under the same malware family will query similar URL domains by using DNS packets, and further use the URL analysis engine URL Void to detect that many URLs are malicious URLs.

CREDROID [15] proposed 22 privacy leakage features for Android malware detection. CREDROID defines 20 Personally Identifiable Information (PII) privacy keywords, and then through the string comparison method to compare the network packet with PII privacy keywords to produce 20 privacy leakage features. And use the IMEI code, user address book to obtain another 2 privacy leakage features. In addition, [16] use VirusTotal API to transmit each IP connected by Android application to VirusTotal for analysis, and then establish 6 IP reputation score features to detect Android malware. The following describes how to obtain 6 IP reputation score features. To begin with, get three materials, detected_url (Latest detected URLs), detected_communicating (Latest detected files that communicate with this IP address) and detected_download (Latest detected files that were downloaded from this IP address) from the return data of the VirusTotal API. Next, the IP analysis results greater than 0.1 in each material are summed separately to obtain three feature values named detected_url_01, detected_communicating_01, and detected_download_01. Finally, the IP analysis results greater than 0.5 in each material are summed separately to obtain three feature values named detected_url_05, detected_communicating_05, and detected_download_05.

For example, suppose an application sends packets with four different destination IPs, and the detected_url obtained from IP analysis results is 0.2, 0.6, 0.08 and 0.3 respectively. As a result of the returned data, the detected_url_01 feature value is 0.2 + 0.3 + 0.6 = 1.1, and the detected_url_05 feature value is 0.6.

C. Summary of Related Work

In the related work of static analysis, the related research on static analysis of Android malware is introduced. The static analysis method is fast, low-cost, and provides a complete analysis of the application execution path. Many static analysis studies use the application's permission information as their detection feature, and most studies use the Random Forest algorithm to obtain the best detection accuracy. Additionally, [12] mentioned that the permission-based detection method is more suitable as a preliminary filtering mechanism for the malware classification system. Therefore, this paper first used static permission analysis and Random Forest to filter out malicious applications from the Android application dataset, and then uses dynamic analysis methods to classify malware families.

In the related work of network traffic analysis, it can be known that there have been many researches on the use of network traffic for dynamic analysis to detect Android Malware in recent years. As shown in Table I, most researches only use a single network traffic protocol for malware detection, rather than combining the features of multiple types of protocols for detection. Therefore, this paper combines the features of multiple types of network traffic protocols extracted from network traffic as the basis for analyzing and detecting Android applications. The experimental results prove that the detection accuracy can be significantly improved.

TABLE I. SUMMARY OF THE RELATED WORK IN NETWORK TRAFFIC DYNAMIC ANALYSIS FOR ANDROID APPLICATION

Related Work	Network Packet Type				Years
	TCP	*HTTP*	*DNS*	*Other*	
[5]			V		2015
[14]		V			2015
[13]	V				2015
[7]	V	V			2016
[1]		V			2016
[15]				V	2016
[8]	V				2017
[9]		V			2017
[16]				V	2017

III. ARCHITECTURE

This section provides a detailed description of the system architecture of the Android Application Analysis System proposed in this paper. The overall system architecture and detailed system design are described in system overview. And the summary of system flow describes the complete data and operational flow.

A. System Overview

The architecture of Android Application Analysis System presented in this paper is shown in Fig. 1. This system has four main modules and a database for storing analysis results. The main four modules are risk detector module, data collector module, feature extractor module and classifier module. The function of each module is described in detail below.

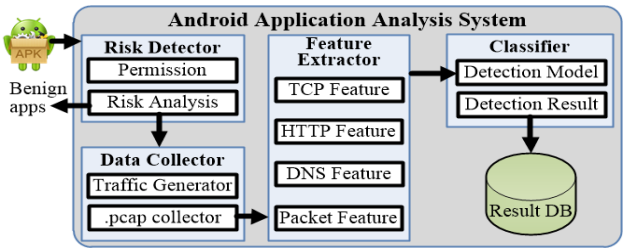

Figure 1. System overview.

1) Risk detector module

The risk detector module is used to identify whether an application is benign or malicious to filter out benign applications. The purpose is to reduce the number of samples collected in the dynamic analysis stage to reduce the data collection time required for the overall dynamic analysis. This module is divided into two phases: permission extraction and risk analysis.

The first phase requires collecting permission information declared by the application. This paper uses the aapt dump permissions command to extract the permission list of the application [17]. In addition to the system permissions [18] of the Android system, there are custom permissions [19] in the permissions declared by the application. Custom permissions are permissions that an application developer defines as needed to allow applications to protect different components, restrict other applications' access to specific components, and expose their functionality to other applications. After collecting the permission list, convert it to a permission vector.

After the application permission vector is obtained, the second phase classifies the risks through a pre-trained model. The pre-trained model's training materials have a total of 2024 training samples, including 1024 malware samples in the Drebin [6] dataset and 1000 benign application samples in the popular rankings of the Google Play store [20]. Record the permissions used in the 2024 training samples and extract their permission vectors. Then use Random Forest as the classification algorithm of this module, confirm the model accuracy with 10-fold cross-validation, and use the model with the best accuracy as the risk analysis model in the risk detector module.

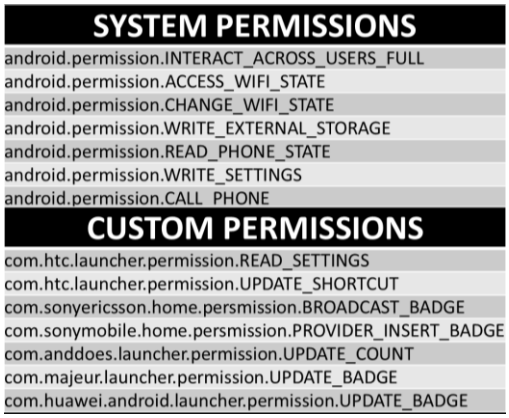

Figure 2. System permissions and custom permissions.

2) Data collector module

The dynamic data collection process of this paper is shown in Fig. 3. Each Android application sample entering the Data Collector Module collects network traffic data through a complete dynamic data collection process. The process includes important actions such as installing samples in the Android emulator, using tcpdump to obtain network packets, and generating events through the Monkey command-line tool to trigger malicious behavior. The initialization phase takes 15 minutes and the collection phase takes 5 minutes.

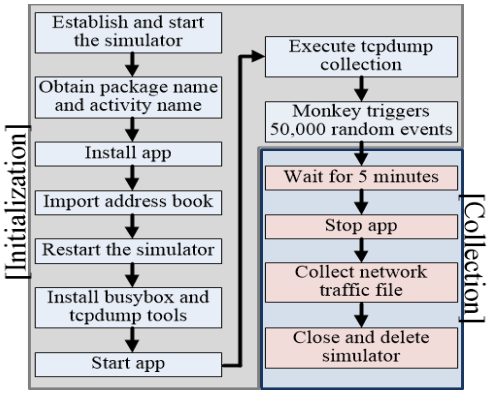

Figure 3. Network traffic collection flow.

3) Feature Extractor Module

The feature extractor module extracts four types of dynamic analysis features from the .pcap file in the network packet of the Android application. The four types are shown in Table II, which are DNS packets, TCP packets, HTTP packets, and other packet contents. For TCP packets, this paper extracts the uplink/downlink packet number, uplink/downlink traffic, average uplink/downlink traffic of packet features described by [7], and the connected IP feature mentioned by [13]. For HTTP packets, this paper uses the Host, Request Method, Request Uri and User Agent described by [7] to construct feature vectors. In addition, DNS packets refer to [5] for feature extraction. The feature vector is constructed by all domain names that appear in the DNS packet record. Lastly, other packet content features are extracted from the privacy leakage features and IP reputation score features in the way described in section II.

TABLE II. DYNAMIC NETWORK PACKET EXTRACTION FEATURES

Network Traffic Types	Features
TCP	● Uploading / downloading packet number ● Uploading / downloading bytes ● Average bytes of uploading / downloading ● Connected IP
HTTP	● HOST ● Request-Uri ● Request-Method ● User-Agent
DNS	● Domain name
Other	● Leakage of privacy information ● IP Reputation Score

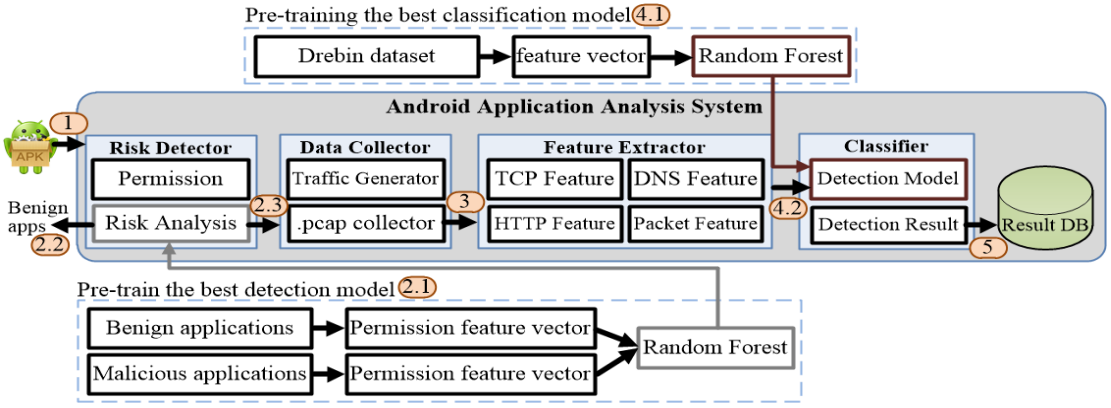

Figure 4. System flow.

4) Classifier module

The classifier module classifies the family of malware through pre-trained model. In order to avoid misjudge the benign application which the risk detector module failed to filter, the classifier module incorporates a benign application dataset in the training data. The pre-trained classification model's training materials have a total of 1410 training samples, including 1360 malware samples in the Drebin [6] dataset and 50 benign application samples in the popular rankings of the Google Play store [20]. Collect the network packet data of the 1410 training samples and extract the four feature vectors described in Feature Extractor Module. Then use the most suitable machine learning algorithm to construct the classification model. In this paper, various machine learning algorithms are tested to train the classifier, and 10-fold cross-validation is used to confirm the accuracy of the classification model. Finally, the algorithm with the best accuracy is adopted as the final classification model.

B. Summary of System Flow

This paper presents an analysis system that can detect and classify Android applications. Through this system, users can detect whether the Android application is benign or malicious, and find out the malware family to which the malicious application belongs. For a detailed description of each module in the analysis system, see System Overview. This section summarizes the flow of analysis system, as shown in Fig. 4. The following is a detailed description of each step of the system flow.

Step 1. Upload the application to be detected to the Android Application Analysis System. The risk detector module first collects the permission list of this application and converts it into the permission vector.

Step 2. For risk detection, there are three steps in this phase:
- Step 2.1. The best malware detection model is pre-trained and placed in this module for malware detection.
- Step 2.2. Detect Android applications through a pre-trained model. If the Android application is detected as a benign application, it is filtered out and informs the user that the application detection result is benign.
- Step 2.3. If the application is determined to be malicious, it is sent to the data collector module to collect network packet data.

Step 3. Extract the TCP packet features, HTTP packet features, DNS packet features, and other packet content features from the .pcap file of the collected network packets.

Step 4. Family classification of Android application, this phase has two steps:
- Step 4.1. Pre-train the best Android application family classification model and put it into the classifier module for application family classification.
- Step 4.2. The malware family classification is completed through the pre-trained classification model and the network packet features produced in step 3.

Step 5. Displays the result of the malware family classification and stores the result to Result DB.

IV. Experiments and Evaluation

This section provides three experiments to evaluate the functions and effectiveness of the Android Application Analysis System proposed in this paper. The experimental environment of this research is built on Windows 10, the CPU is i7-7700 3.6GHz, memory uses 16G DDR3, DB is MySQL, Malware dataset come from Drebin [6], and Benign come from Google Play store [20].

A. Accuracy of Risk Detector Module (Static Analysis)

The purpose of this experiment is to pre-train the best static risk detection model for the Android Application Analysis System. Also, verify the ability of a risk detector module based on static analysis to identify benign applications and malicious applications. The experimental sample contains 1000 benign applications and 1024 malicious applications. The experiment used 2024 samples to generate two feature datasets. Each sample of feature dataset 1 has 2582 features, which are generated by extracting system permissions and custom permissions for each sample. Feature dataset 2 uses only system permissions to generate 226 features for each sample. In addition, this experiment uses 10-fold cross-validation to compare the accuracy of various machine learning algorithms, and finally adopts the algorithm with the best accuracy as the detection model of the risk detector module.

The accuracy of various machine learning algorithms is shown in Table III, and the Random Forest algorithm has the best accuracy. Then, this paper adjusts the parameter (number of trees) of the Random Forest algorithm to obtain the best detection model. As shown in Fig. 5, when the parameter is 300 or 400, the Feature dataset 2 can achieve the highest accuracy of 98.86%. Therefore, this paper uses the random forest with parameter 300 and the feature dataset 2 to generate the model as the detection model of the risk detection module.

TABLE III. Accuracy of Different Algorithms Under Static Analysis

Machine learning algorithm	Feature dataset 1	Feature dataset 2
Naïve Bayes	97.58%	95.55%
Random Forest	**98.72%**	**98.86%**
J48	97.72%	96.39%
KNN	96.79%	97.72%
PART	97.13%	97.13%

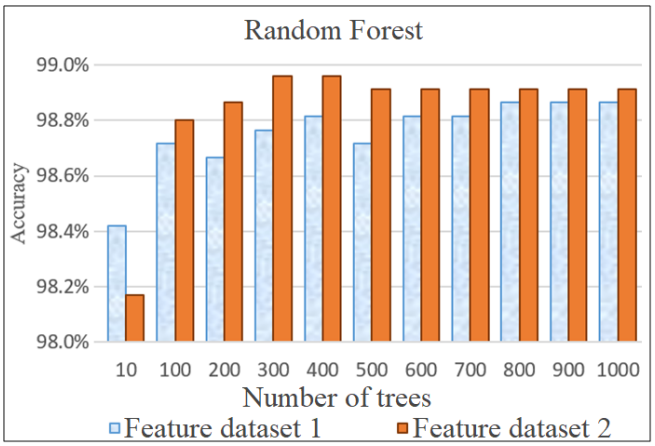

Figure 5. Accuracy of random forest models with different parameters.

B. *Accuracy of Classifier Module (Dynamic Analysis)*

1) *Comparison of four types of network packet features*

The purpose of this experiment is to compare the accuracy of application family classification for a single type of network packet feature and multiple types of network packet features. The sample was a collection of 1,360 malicious applications and 50 benign applications from ten malicious families. This experiment treats all benign applications as a family, so there are a total of 11 family categories. In this experiment, Random Forest is adopted as the machine learning algorithm, and the trained model will be used as the classification model of the classifier module. The experimental result is shown in Fig. 6. The classification accuracy of the four characteristics of TCP, DNS, other content, and HTTP falls within the range of about 78% to 91%, while the classification accuracy of combining the four features can reach 95.6%. This experiment can prove that the classification effect of combining multiple types of network packet features is better than using only a single network packet feature for classification.

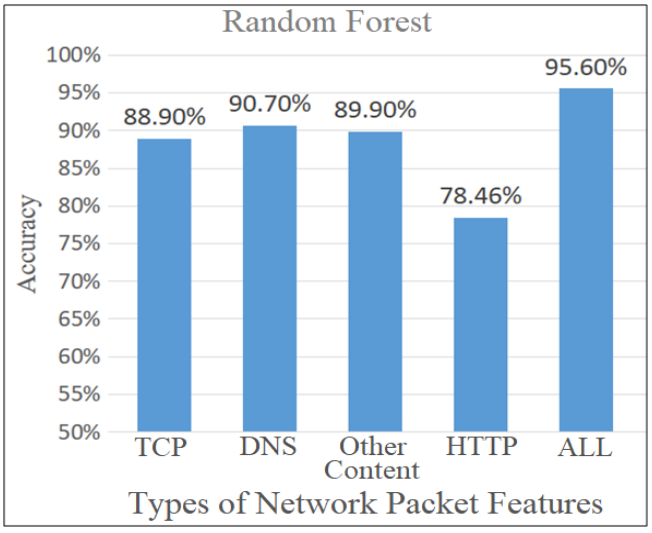

Figure 6. Comparison of the accuracy of network packet features.

2) *Compare with DroidClassifier*

One of the contributions of this paper is to propose a dynamic analysis model that combines multiple network packet features to classify application families. In this experiment, the dynamic analysis model is directly compared with DroidClassifier [1] for malicious application family classification accuracy. In the comparison method, 1,360 malicious programs of ten malicious families are taken as samples, and the accuracy of 10-fold cross-validation is compared between the dynamic analysis model proposed in this paper and DroidClassifier. As a result of the experiment, the dynamic analysis method proposed in this paper combined with Random Forest to construct the classification model can obtain 96% accuracy, which is 1.67% more than the 94.33% accuracy of DroidClassifier.

C. *Overall System Verification*

The purpose of this experiment is to verify the overall efficiency of Android Application Analysis System. The experimental sample consists of 100 malicious applications and 100 benign applications. It is worth noting that these 200 test samples are completely different from the applications used when training the detection and classification model. In order to complete the Android Application Analysis System, the previous experiment named "Accuracy of Risk Detector Module" provided its highest accuracy model, which was set as the detection model of the risk detector module. And the previous experiment named "Comparison of four types of network packet features" offered its highest accuracy model, which was set as the classification model of the classifier module.

One of the contributions of this paper is to propose a two-stage analysis method combining static and dynamic analysis. This section uses two procedures to perform two application classification experiments to compare the advantages and disadvantages of the proposed analysis method and the traditional single-stage analysis method. The procedure 1 uses the entire process of the Android Application Analysis System to classify 200 samples into Application families. The procedure 2 is the application family classification of 200 samples only through dynamic analysis of the classification model. Finally, the data collection time and classification accuracy of the two procedures are compared.

In procedure 1, the risk detector module judges 105 samples as benign applications and then filters them out, including 6 malicious applications that are misidentified as benign. The feature extractor module only needs to collect the dynamic analysis data of 95 samples, which takes 1900 minutes. After analysis by the classifier module, the family classification results of 5 applications are wrong. Therefore, a total of 11 applications are classified incorrectly. On the other hand, in the program 2, 200 samples need to collect dynamic analysis data, which takes 4,000 minutes. In the end, the family classification results for 14 applications are wrong. As shown in Table IV, this experiment confirmed that the Android Application Analysis System proposed in this paper can not only save 52.5% of dynamic data collection time but also improve the accuracy of Android application family classification.

TABLE IV. TWO-STAGE ANALYSIS VS. SINGLE-STAGE ANALYSIS

	# of misclassifications	Data Collection Time	Accuracy
Procedure 1	**11**	**1,900 Minutes**	**94.5%**
Procedure 2	14	4,000 Minutes	93%

V. CONCLUSION

In recent years, Android has maintained a fairly high market share, and thus has become the target of many malicious programs. This paper aims at the security topic of Android system, and proposes an analysis system that can detect Android malware and do application family classification. This paper establishes an Android Application Analysis System that combines static and dynamic analysis. The goal is to reduce the number of applications that need to collect dynamic data, which in turn reduces overall analysis time. The experiment confirmed that the Android Application Analysis System proposed in this paper can not only save 52.5% of dynamic data collection time but also improve the accuracy of Android application family classification. In addition, the classification model proposed in this paper combined with multiple network packet features has 96% accuracy higher than the 94.33% accuracy of DroidClassifier [1]. Finally, the analysis system proposed in this paper can classify Android application family, and this classification can help information security engineer to understand the behavior pattern of malware, take or establish similar defensive protection measures, or pre-suspend the execution of the malware in advance, so as to achieve the purpose of intrusion prevention.

Future work will study the data and features generated by the methods presented in this paper to match the deep learning to produce more effective detection and classification effects.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers' helpful comments. This research is partially supported by the Software Research Center of National Central University, Taiwan, and the Ministry of Science and Technology, Taiwan. The Grant number is MOST 105-2221-E-008-069-MY2.

REFERENCES

[1] Z. Li, L. Sun, Q. Yan, W. Srisa-An, and Z. Chen, "DroidClassifier: Efficient adaptive mining of application-layer header for classifying Android malware," International Conference on Security and Privacy in Communication Systems, pp. 597–616, October 2016.

[2] M. Chau, Smartphone Market Share. Available at: https://www.idc.com/promo/smartphone-market-share/os. (accessed 08.01.2019).

[3] NOKIA, Nokia Threat Intelligence Report, 2019, Available at: https://pages.nokia.com/T003B6-Threat-Intelligence-Report-2019.html (accessed 08.01.2019).

[4] Cisco, Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper, 2019, Available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html (accessed 08.01.2019).

[5] Z. Chen, H. Han, Q. Yan, B. Yang, L. Peng, L. Zhang, and J. Li, "A first look at android malware traffic in first few minutes," 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 206–213, August 2015.

[6] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of Android malware in your pocket," The Network and Distributed System Security Symposium, vol. 14, pp. 23–26, February 2014.

[7] S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, and Z. Jia, "TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic," 24th IEEE/ACM Int. Symp. Qual. Service, pp. 1–6, June 2016.

[8] Y. Pang, Z. Chen, X. Li, S. Wang, C. Zhao, L. Wang, K. Ji, and Z. Li, "Finding Android Malware Trace from Highly Imbalanced Network Traffic," IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), pp. 588–595, July 2017.

[9] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "TextDroid: Semantics-based detection of mobile malware using network flows," IEEE Conference on Computer Communications Workshops, pp. 18–23, May 2017.

[10] B. Sanz, I. Santos, C. Laorden, X. U.-P. P. Bringas, and G. Alvarez, "PUMA: Permission Usage to detect Malware in Android," International Joint Conference CISIS12-ICEUTE12-SOCO12 Special Sessions, in Advances in Intelligent Systems and Computing, vol. 189, pp. 289–298, 2013.

[11] J. G. de la Puerta, B. Sanz, I. S. Grueiro, and P. G. Bringas, "The Evolution of Permission as Feature for Android Malware Detection," International Joint Conference Springer International Publishing, pp. 389–400, 2015.

[12] X. Liu, and J. Liu, "A two-layered permission-based Android Malware detection scheme," 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp. 142–148, April 2014.

[13] F. Narudin, A. Feizollah, N. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," Soft Computing, vol. 20, pp. 343–357, January 2016.

[14] M. Aresu, D. Ariu, M. Ahmadi, D. Maiorca, and G. Giacinto, "Clustering Android malware families by http traffic,," 10th International Conference on Malicious and Unwanted Software (MALWARE), pp. 128–135, October 2015.

[15] J. Malik, and R. Kaushal, "CREDROID: Android malware detection by network traffic analysis," 1st ACM Workshop on Privacy-Aware Mobile Computing, pp. 28–36, July 2016.

[16] C. Hu, "Dynamic Android Malware Analysis with de-identification of personal identifiable information," MA thesis, National Central University, Taiwan, 2017.

[17] Little418, Check APK Permissions with aapt, 2014, Available at: https://little418.com/2014/07/check-apk-permissions-with-aapt.html (accessed 08.01.2019).

[18] Android Developers, Permissions overview, 2018, Available at: https://developer.android.com/guide/topics/permissions/overview (accessed 08.01.2019).

[19] Android Developers, Define a Custom App Permission, 2018, Available at: https://developer.android.com/guide/topics/permissions/defining (accessed 08.01.2019).

[20] Google, Google Play Apps, 2019, Available at: https://play.google.com/store/apps/top (accessed 08.01.2019).