An Improved Malicious Application Detection in Social Networks (MADSN)

Nagmden Miled Nasser Adel Ashour Abosdel

Faculty of Engineering, Azzaytuna University

الملخص:

الأندرويد هو نظام تشغيل الأجهزة المحمولة الأكثر استخدامًا (OS) ،وظهر عدد كبير من أسواق تطبيقات الأندرويد التابعة لجهات خارجية، و دفع غياب تنظيم السوق الخاص بطرف ثالث للمؤسسات البحثية إلى اقتراح تقنيات مختلفة للكشف عن البرامج الضارة ومع ذلك ، نظرًا لتحسينات البرامج الضارة نفسها ونظام الأندرويد ، من الصعب تصميم طريقة اكتشاف يمكنها بكفاءة وفعالية اكتشاف التطبيقات الضارة لفترة طويلة، وفي الوقت نفسه سيؤدي اعتماد المزيد من الميزات إلى زيادة تعقيد النموذج والتكلفة الحسابية للنظام. تؤدي الأذونات دورًا حيويًا في أمان تطبيقات الأندرويد. في هذا البحث تمّ اقتراح نموذج للكشف عن التطبيقات الضارة استنادًا إلى عدم الثقة في السمات، وحيث يستخدم (MADSN) وظيفة الانحدار اللوجستي لوصف علاقة الإدخال (الأذونات) والإخراج (الملصقات) علاوة على ذلك ، فإنه يستخدم خوارزمية سلسلة ماركوف مونت كارلو (MCMC) لحل الميزات عدم الثقة بعد تجربة 2037 عينة ، للكشف عن البرامج الضارة، و أظهرت نتائج التجربة أن استخدام الأذونات الخطرة فقط، أو أن عدد الأذونات المستخدمة لا يمكن أن يميز بدقة ما إذا كان التطبيق ضارًا أم حميدًا. بالنسبة لاكتشاف البرامج الضارة و يحقق النهج المقترح دقة تصل إلى 95.5٪ ومعدل إيجابي كاذب (FPR) يبلغ 1.2٪ ، وبالنسبة لاكتشاف عائلات البرامج الضارة ، تبلغ الدقة 95.6٪. تشير النتائج إلى أن طريقة دقة الكشف عن العينة غير المعروفة الجديدة تبلغ 92.71٪. بالمقارنة مع الأساليب الحديثة الأخرى ، فإن النهج المقترح أكثر فعالية من خلال اكتشاف عائلات البرامج الضارة والبرامج الضارة.

Abstract:

Android is the most widely used mobile operating system (OS). A large number of third-party Android application (app) markets have emerged. The absence of third-party market regulation has prompted research institutions to propose different malware detection techniques. However, due to improvements of malware itself and Android system, it is difficult to design a detection method that can efficiently and effectively detect malicious apps for a long time. Meanwhile, adopting more features will increase the complexity of the model and the computational cost of the system. Permissions play a vital role in the security of the Android apps. In this paper, a malicious application detection model based on features uncertainty is proposed MADSN uses logistic regression function to describe the input (permissions) and output (labels) relationship Moreover, it uses the Markov chain Monte Carlo (MCMC) algorithm to solve features' uncertainty. After experimenting with 2037 samples, for malware detection ,The experiment results show that only use dangerous permissions or the number of used permissions can't accurately distinguish whether an app is malicious or benign. For malware detection, the proposed approach achieve up to 95.5% accuracy and the false positive rate (FPR) is 1.2%.For malware families detection, he accuracy is 95.6%. The results indicate that the method for unknown/new sample's detection accuracy is 92.71%. Compared against other state of-the-art approaches, the proposed approach is more effective by detecting malware and malware families.

Keywords: APK,MCMC, machine learning, permission , malicious.

Introduction

Android is currently the most used smart-mobile device platform in the world, occupying 82.8% of market share (Andreas, et al.,2010,p14). As of now, there are nearly 2 million apps available for downloading from Google Play, and more than 50 billion downloads to date. Unfortunately, the popularity of Android also creates interests from cyber-criminals who create malicious apps that can steal sensitive information and compromise systems. Unlike other competing smart-mobile device platforms, such as iOS, Android allows users to install applications from unverified sources such as third party stores. In order to remove malicious apps and low-quality apps from the application market, a large number of malicious app detection technologies have been applied, such as static detection and dynamic detection(Faruki, et al.,2017) . Static detection does not

need to run the app. It analyzes the files in the APK package to determine whether the app is benign or malicious (Tuncay , Demetriou , Ganju ,& Gunter ,2018). Static detection method is based on decompilation technology and doesn't need run the apps. It analyses the code, rule matching and other operations (such as permissions, data flow, control flow, etc.) (Faruki, et al.,2017). MaMaDroid (Onwuzurike et al.,2019)used Markov chains to build API sequence model. The method learn and test through the feature obtained by API sequence model. The F-measure of MaMaDroid can reached 99%. Droid Sieve (Zhao, Zhang, , Su, & Li,2015) proposed high-quality features for malware detection and malware family detection. These features include Intents, permissions, mate-information, etc. MUDFLOW (Avdiienko , et al.,2015) used sensitive sources (include the Intents, Sinks, API, etc.) to detect new malware and its accuracy can reach 86.4%.In contrast to static detection, dynamic detection detects apps' behavior at runtime. It captures and analyzes sensitive behavior in real time. Dynamic detection needs to be run in a specially built environment **(**Yang, Huang , & Gu ,2018) . DroidCat (Cai , Meng , Ryder , & Yao ,2019) used dynamic features to detect resource obfuscation, system-call obfuscation and other obfuscation. The F1 of DroidCat can reached 97%. DroidScribe (Dash, et al.,2016)analyzed the running behavior of apps by dynamic detection method and divided malware into different families. In this paper as follows :A brief introduction to the Facebook architecture , the characteristics of Android permissions and MCMC. In next Section the introduces the methodology of this paper. In next Section is an evaluation and a discussion. The last section concludes this paper.

Background

Facebook Architecture

Facebook, one of the leading social networks, offers a framework for software developers to create lightweight applications. These application are able to run inside the social network and interact with its resources (users and users' data). When a user accesses a Canvas page, several steps occur in the Facebook REST server and the hosting server in order to render application's contents to the user's browser. These steps are depicted in Figure 1. Initially, the user's browser requests the Canvas page URL from the Facebook server(Tuncay, et al,2018). Following, the Facebook server sends an HTTP POST request to the application hosting server for the Callback URL, asking for the FBML of the

Canvas page. If the application needs to retrieve any social data then the hosting server sends an HTTP GET/POST request to the Facebook REST server for the needed data. After executing all API method calls, the hosting server returns the resulted FBML to the Facebook server. The Facebook server transforms that FBML into HTML and sends it back to the user's browser (Andreas, et al.,2010,p14).

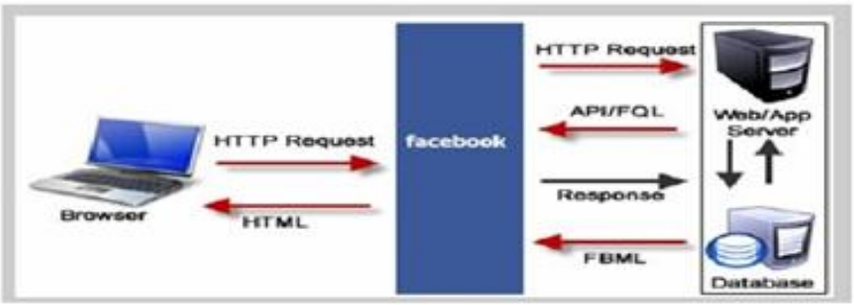


Figure 1: How an FBML Canvas page is rendered

Android Permission

The Android OS is an open source OS, mainly based on Linux, and considered one of the most popular mobile phone operating systems. Android was developed by Google (Du , Wang , &Li ,2017) and a group of developers known as the Open Handset Alliance (Tuncay, et al,2018).The distribution of applications for this OS is basically done through application stores such as the Google Play Store or the Samsung Galaxy Store (LuL, Li , Wu , Lee, & Jiang ,2012), etc. In contrast to IOS, Android allows independent installation of apps hence leaving itself open to potentially harmful ones. This makes the need for Android malware detection systems even greater. The apps in these markets are stored in an Android PacKage (APK) format. Among them. An APK file is a compressed file that packs the apps Dalvik bytecode (Classes.dex files), compiled and plain resources, assets, and the XML manifest file.The AndroidManifest.xml (or Manifest file) is designed for the meta-data such as requests for permissions, components defined in the app such as Activities, Services, etc.

AndroidManifest.xml file: AndroidManifest.xml file is the entry of an application, including the following authentication information: version, package name, components, permissions and other basic information (Du et al.

,2017).Our research focuses on the permissions under tag. The 24 dangerous permissions declared by Android in Google are extracted, The extracted permission table is shown in Table 1.

Table 1:The extracted permission in the Android Manifest.xml file

Number	Permission Name(prefix is omitted)	Number	Permission Name(prefix is omitted)
1	READ_CLALENDAR	2	WRITE_CLALENDAR
3	CAMERA	4	READ_CONTACTS
5	WRITE_CONTACTS	6	GET_ACCOUNTS
7	ACCESS_FINE_LOCATION	8	ACCESS_COARSE_LOCATION
9	RECORE _AUDIO	10	READ_PHONE_STATE
11	ANSWER_PHONE_CALLS	12	READ_CALL_LOG
13	WRITE_CALL_LOG	14	ADD_VOICEMAIL
15	USE_SIP	16	PROCESS_OUTGOING_CALLS
17	BODY_SENSORS	18	SEND_SMS
19	RECEIVE_SMS	20	READ_SMS
21	RECEIVE_WAP_PUSH	22	RECEIVE_MMS
23	READ_EXTERNAL_STORAGE	24	WRITE_EXTERNAL_STORAGE

Markov Chain Monte Carlo (MCMC)

The Markov chain Monte Carlo method (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain. The more steps are included, the more closely the distribution of the sample matches the actual desired distribution. Various algorithms exist for constructing chains, including the Metropolis–Hastings .(MCMC) introduces the Markov process into Monte Carlo simulation (Suarez-Tangil, et al.,2017). It implements the dynamic simulation of random samplings (Hock, &Earle,2016). It makes up for the defect that the traditional Monte Carlo method can only simulate statically (Li , SunL,Yan, Li, Srisaan , &Ye,2018).

MADSN Model

This section mainly introduces the MADSN model proposed in this paper. The MADSN model consists of three parts: Support Based Permission Ranking (SPR), Data pre-Processing and machine learning and Detection System. As shown in Figure 2.

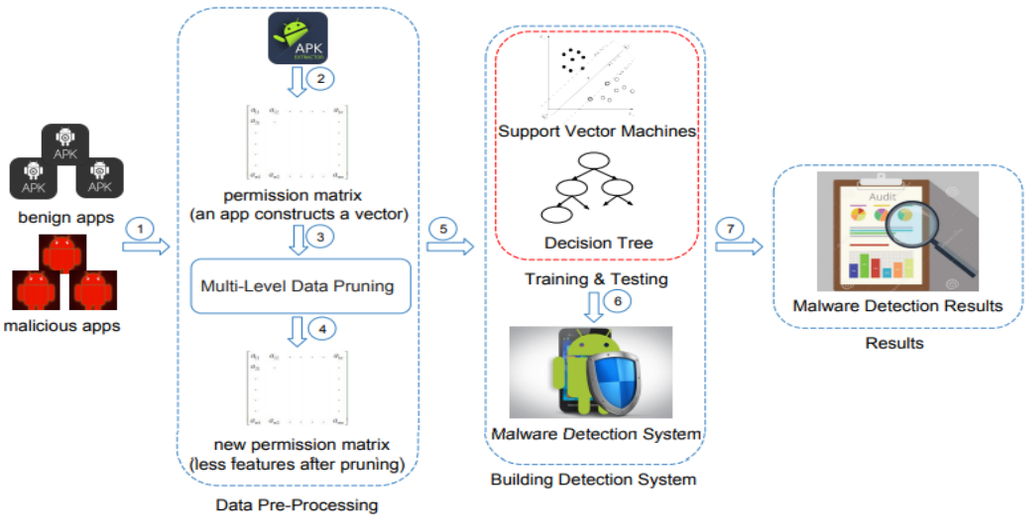


Figure 2: Overview of a malicious application detection model MADSN

Support Based Permission Ranking (SPR)

To further reduce the number of permissions, we turn our focus to the support of each permission. Typically, if the support of a permission is too low, it does not have much impact on malware detection. Then, the requested permission list is built by extracting permission requests from each app listed in Android Manifest file. The permission information is translated into a binary format dataset where '1' indicates that the app requests the permission, and '0' indicates the opposite. The permission lists extracted from malicious apps and benign apps are combined to form a holistic dataset for data analysis. The extracted permission matrix is shown in Figure 3.

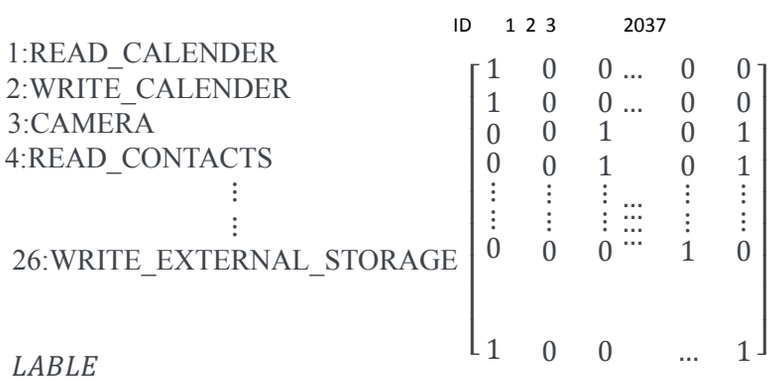


Figure 3: permission matrix

Function model selection

In the MCMC calculation, a suitable function needs to be selected for the sampled posterior distribution model. Suppose …..,$x_{i-3}, x_{i-2}, x_{i-1}, x_i$, … …. are the states in a Markov chain. The transition probability can be described as by:

$$P(x_i|\ldots\ldots, x_{i-3}, x_{i-2}, x_{i-1}) = P(x_i|x_{i-1}) \quad (1)$$

where P is the probability of the occurrence of the event(benign or malicious); $P(x_i|x_{i-1})$ is the probability of transitioning to the state x_i under the condition of x_{i-1}.

$$; P(x_i|x_{i-1}) = \frac{1}{1+e^{-(\beta x_i+\alpha)}} \quad (2)$$

Where β is the weight of the parameters in the model, α is the measurement noise, Moreover , α and β simulate their values though MCMC . Logistic regression assumes that the dependent variable P follows the Bernoulli distribution is shown as Figure 3

$$S_i \sim Ber\big(P(x_i)\big), i = 1,2, \ldots\ldots, N \quad (3)$$

$$; P \sim Ber\left(\frac{1}{1+e^{-(\beta x_i+\alpha)}}\right) \quad (4)$$

The goal of MCMC was to find the optimal values of parameters β and α based on the data from the assumption of normal prior distribution.

$$\alpha \sim N(\mu_i, \delta_i^2) \quad (5)$$

$$\beta \sim N(\mu_i, \delta_i^2) \quad (6)$$

$$\mu_i = \alpha + \beta x_i \quad (7)$$

where α is the intercept, and β is the coefficient for covariate Xi, while δ_i represents the observation or measurement error.In summary, the parameter relationship in the logistic regression model is shown as Figure 4.

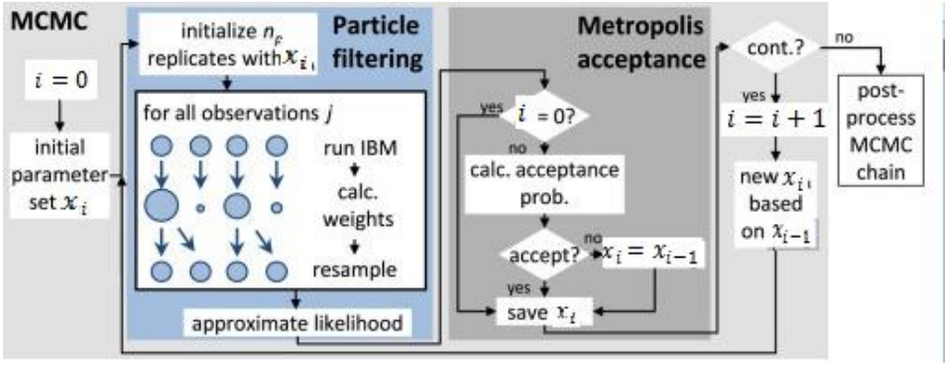


Figure 4:Simplified-flowcharts-of-the-PMCMC-algorithm-combining-MCMC-left-for-posterior-sampling

Using the Metropolis–Hastings algorithm to sample the posterior distribution of α and β the algorithm is shown as Figure 5.

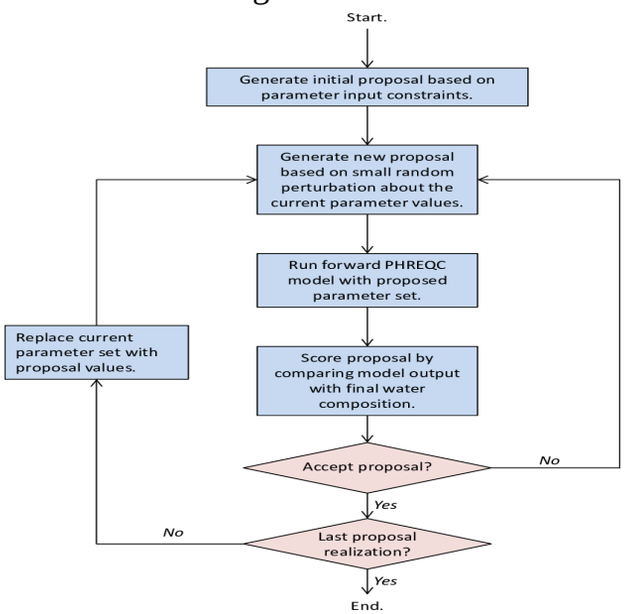


Figure 5:The Metropolis-Hasting Monte Carlo (MHMC) algorithm

Through MCMC sampling, the 95% confidence interval of the highest probability density interval of beta can be calculated. The larger the interval, the greater the uncertainty value of the permission, and the less suitable it is for malicious detection. On the contrary, it is more suitable for malicious detection.

Machine learning and Detection System

Machine learning (ML) classifiers have played a part in the development of intelligent systems for several domains over the years. ML approaches are gaining traction in identification and detection of malware on both mobile and PC platforms. Our work is based on supervised machine learning whereby the features described in the previous section are acquired from a labeled dataset and used to build and train a model(Ian , Witten, Eibe ,& Mark , 2011). The ML algorithms considered in our investigation include: naive Bayes (NB) (probabilistic), Bayesian network (BN) (rule-based), J48 (function-based), random tree (RT) and random forest (RF) machine learning classification algorithms. Figure 6 illustrates the building blocks of the detection approach. The rule based classifiers produce the most easily interpretable output whilst the probabilistic classifier is most easily amenable to post-training sensitivity tuning.

We evaluate our model with five metrics: false positive rate (FPR), accuracy, F-measure, ROC and AUC .

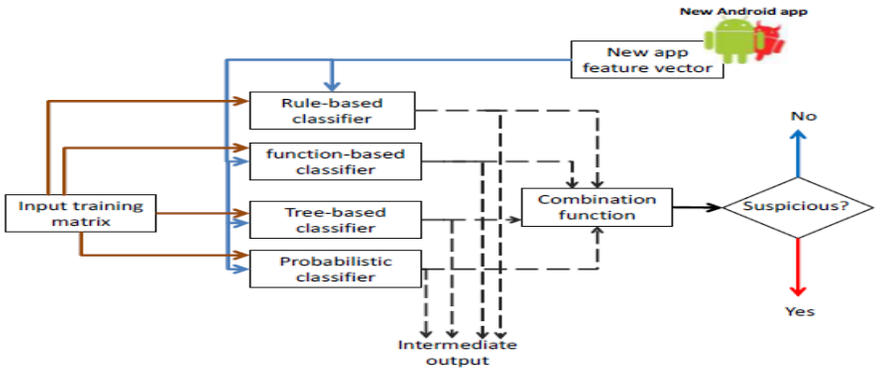


Figure 6:Android malware detection with the composite parallel classifier

Evaluation and discussion

This section mainly introduces the experimental results and discusses the results.

Datasets

The information of the datasets is shown in Table 2, We divided the dataset2 datasets for different experiments. Dataset2 combined with Dataset1 are used to verify that the proposed approach has good detection accuracy for different datasets sizes.

Table 2: Information of datasets

Name	Source of Ben./Mal.	Number of Ben./Mal.	Total number
Dataset1	YingYongBao/Denbin-5	589/556	1145
Dataset2	WanDouJia/Virus Share	469/423	892
Samples Total			**2037**

Experimental Methods

In order to analyze dangerous permissions, MADSN uses the Python language and the data science package (Python Software Foundation,2010), to implement the Metropolis algorithm. We run our Metropolis algorithm on an Intel Core i5 fourth-generation processor with 6 cores clocked at 2.5 GHz, and with 16 GB of on-board memory. For the MCMC run, MADSN selected 5000 samples for analysis, which ensures that the model converges before sampling. The traceplot and autocorrplot of the alpha (α) and beta (α) parameters for READ_PHONE_STATE are shown in Figure 7. When using MCMC, the

initially generated values are often inaccurate. After the Markov chain converges, the generated parameters are used to model the values. We used 10,000 samples to calculate. The previous 50% sample was abandoned.

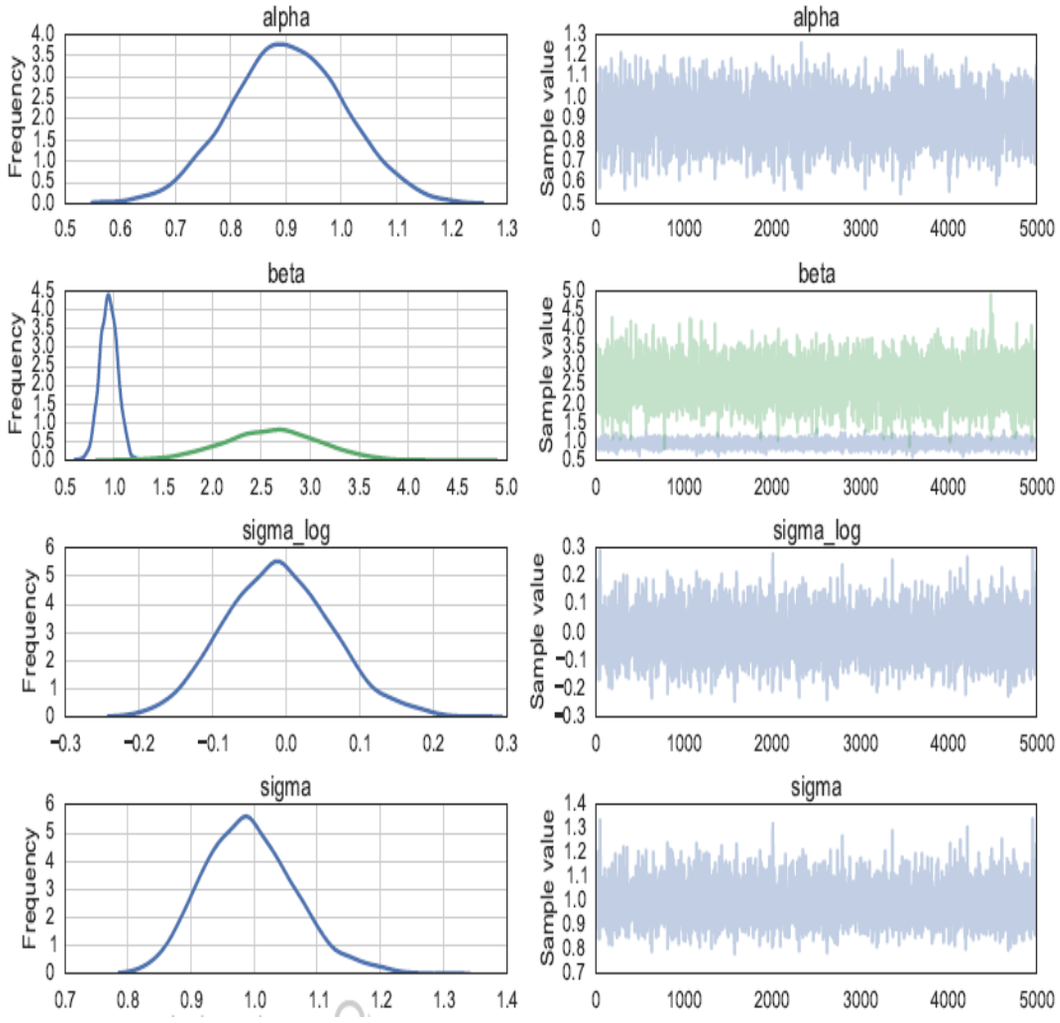


Figure 7:Sampled values of alpha (α) and beta (β) for EAD_PHONE_STATE (left is Autocorrplot,right is Traceplot).

The Uncertainty of Permissions

After MCMC runs, and the posterior probability of all parameters is calculated, the density function set of deferent parameters in MADSN model are obtained, since it is impossible to display a detailed view of the posterior probabilities of all parameters of the model. Thus, the forest plot is used to show the uncertainty of (β) for 24 dangerous permissions, as shown in Figure 9. An interesting result can be seen from Figure 8, permissions which are frequently used in malware (or benign apps) and rarely used in benign apps (or malware) are more important when distinguishing malware from benign apps. So, these

permissions are used to detect malicious apps, We removed 6 permissions (WRITE – CONTACTS ,ADD_VOICEMAIL ,USE_SIP,BODY_SENSORS, ECEIVE_SMS and READ_SMS) that did not contribute significantly to malicious detection. Our method uses the remaining 18 permissions to classify malicious apps.

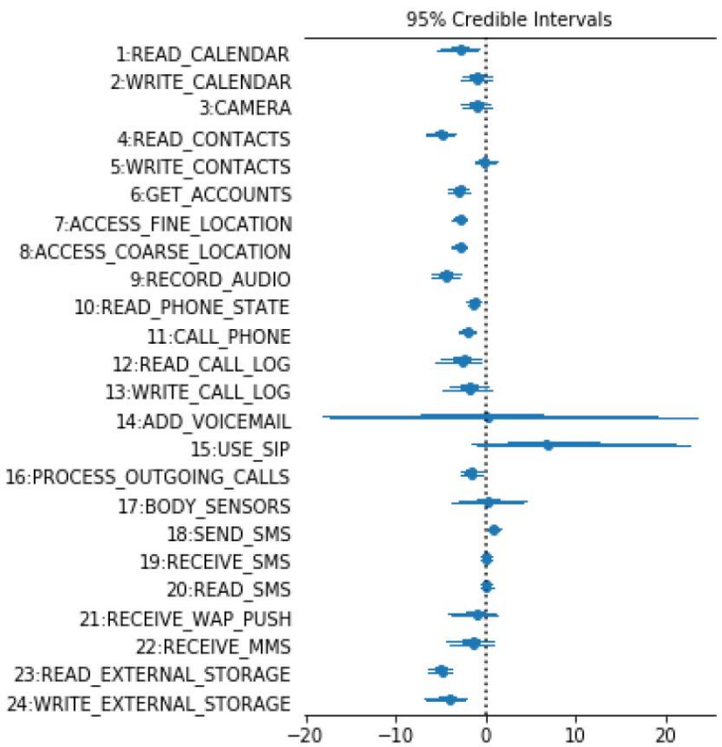
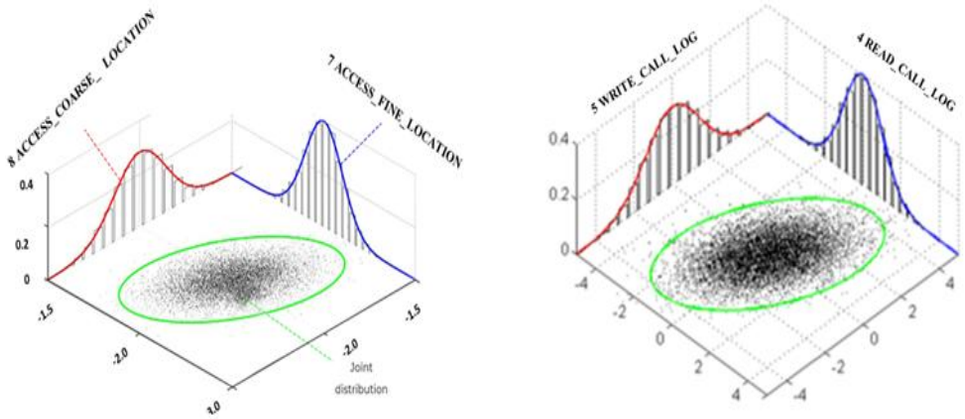


Figure 8:Forest Plot of Interesting (β) Values and their Associated Uncertainties

Joint Probabilities Analysis

Joint probability is a statistical measure that calculates the likelihood of two events occurring together and at the same point in time. After analyzing the probability distribution of different model parameters in detail, we study their joint probabilities, in order to discover more interesting patterns. Scatters are used to represent different permissions relationship, for some highly relevant permission pairs such as ACCESS-FINE-LOCATION and ACCESS-COARSE-LOCATION (Figure 9 a). Most malicious apps use both permissions at the same time. However, most benign apps use one alone. We found similar relationships among several other pairs, such as: READ-CALL-LOG and WRITE=CALL-LOG. On the contrary, READ-CONTACTS and WRITE- CONTACTS (Figure

9 b) belong to the same group of dangerous permissions. However, malicious apps prefer to use one of them. Most benign apps use both.



(a) (b)

(a) β Values for ACCESS_FINE_LOCATION and ACCESS_COARSE_ LOCATION,
(b) β Values for READ_CONTACTS and WRITE_ CONTACTS

Figure 9 : Joint Probability Distributions of β Values for Different Combinations of Permissions.

Model Evaluation

Evaluation metrics such as false positive rate (FPR), accuracy (A) and Area Under ROC curve (AUC) are used as classifier performance indicators (refer Eqs. (8) and (9)). Here, False positive (FP) indicates the misclassification of a benign app as malware. Truly classified benign files are indicated as true negative (TN). Malware samples correctly classified as malware are referred as true positive (TP). Malware instances classified as benign are known as false negative (FN). If a graph is plotted considering FPR values in abscissa and proportion of correctly classified malware instances in the ordinate the resulting curve is known as receiver operating characteristics (ROC) curve. The area in ROC curve corresponds to AUC. The AUC is between 0 and 1, the closer AUC value towards 1, the better the performance of classification model. So models with higher AUCs are preferred over those with lower AUCs. Thus, the detector is considered to be effective if FPR is minimum however large the values of A and AUC may be.

$$FPR = \frac{FP}{FP+TN} \qquad (8)$$

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)} \quad (9)$$

F-Measure represents the harmonic mean of Precision = TP/(TP + FP) and Recall = TP/(TP + FN).
F-Measure is defined as follows:

$$F - Measure = \frac{(2\times Precision\times Recall)}{(Precision+Recall)} \quad (10)$$

Area Under Curve (AUC) is defined as the Area Under ROC Curve. The ROC curve does not clearly indicate which classifiers perform better. But AUC can better evaluate the classifier. The greater the AUC, the better the classifier.In fact, *Recall = TPR*, which is currently assigned to the positive sample category, the true positive sample as a percentage of all positive samples, also called the recall rate (how many positive sample ratios are recalled). Accuracy is the percentage of all samples that are correctly predicted for the correct sample, and represents the differentiating ability of a classifier (where the differentiating ability is not biased to positive or negative examples). Precision-recall is actually two evaluation indicators, but they are generally used simultaneously. Ideally, both are high, but generally high accuracy and low recall, or low recall and high accuracy. In cases where both requirements are high, it can be measured in terms of *F-Measure*.

Performance of Detection

In this section, we evaluate the performance of our method with five machine learning classifiers on different datasets. The experimental results of Percentage Split is shown in Table 3.

Table 3: Classification results of different classifiers.

Classifier	FPR	Accuracy	F-Measure	AUC
NB	8.3%	91.5%	88.3%	83.0%
BN	8.8%	91.1%	90.3%	90.1%
J48	**1.2%**	**95.5%**	**94.7%**	**94.4%**
RT	8.1%	91.8%	91.4%	89.0%
RF	47.0%	69.5%	44.5%	45.1%

It can be seen from Table 3 that comparing the experimental results of each machine learning classifier, the performance of J48 in the case of 10-fold cross validation (TPR, FPR, F-Measure(F-M), Accuracy) are better than other

machine learning classifiers. Among them, FPR reached 1.2%, F-M reached 94.7%, and Accuracy was 95.5%. The AUC of the J48 reaches 94.4% ,In terms of speed, the algorithm is also efficient to train the model is shown in Figure 10.

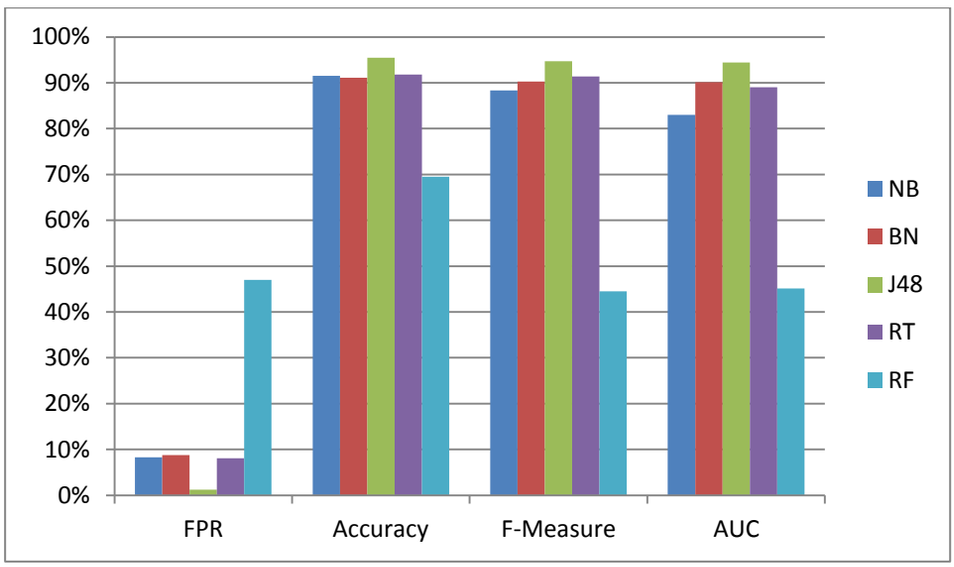


Figure 10: Classification results of different classifiers.

The ROC curve of the J48 classifier is shown in Figure 11. From Figure 11, the curve is close to the upper left, and the area AUC under the curve is 94.4%. According to AUC and ROC curves, the J48 classifier has better classification effect.

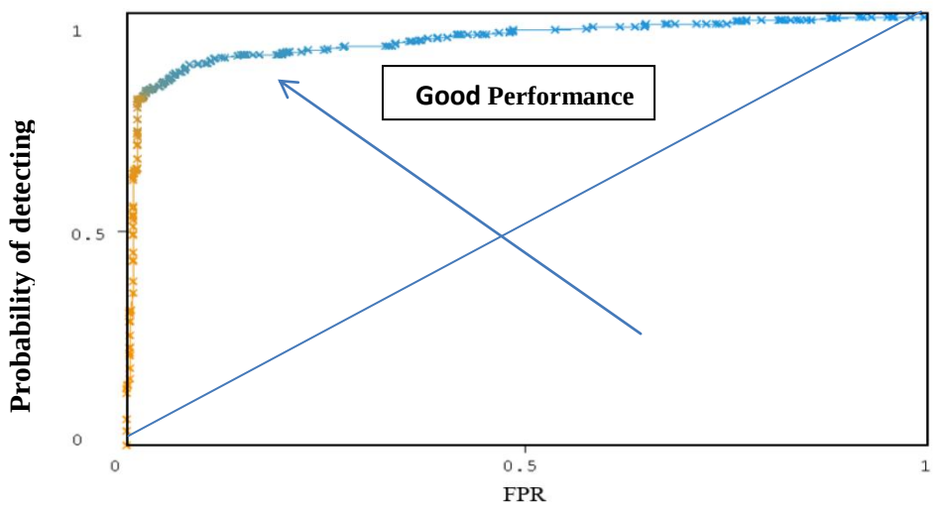


Figure 11: The ROC curves of J48

Table 4: Experimental results for different dataset.

Dataset	Number of Samples	Method	FPR	Acc.	F-M
Dataset1	1145	24 Dangerous Permissions	5.7%	88.7%	87.5%
Dataset1	**1145**	**Our method**	**1.2%**	**95.5%**	**94.7%**
Dataset2	**892**	**Our method**	**5.6%**	**92.7%**	**91.3%**

From Table 4, we can see that after 1145 samples are classified with 18 selected dangerous permissions, Our method's accuracy can reach 95.5%, F-measure can reach 94.7%, and FPR can be 1.2%. Our method's classify performance is higher than using 24 dangerous permissions, because the proposed approach uses fewer features in learning and classification. See in Figure 12.

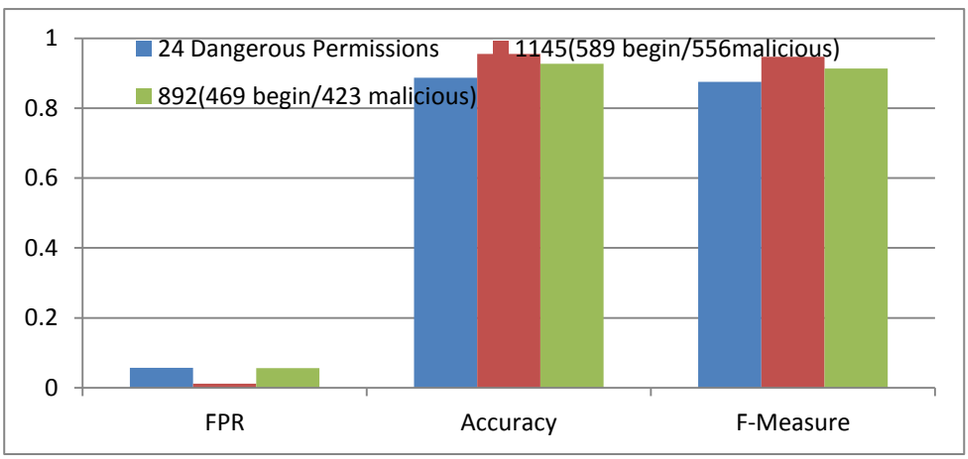


Figure 12: Experimental results for different dataset.

Comparison with other approaches

The proposed approach also is compared with other state-of-the-art malicious detection methods that only use permissions features. SIGPID (Li et al.,2018) is an approach that applies permission ranking. We reimplemented their approach for comparison. Because the dataset used is different, the results are different from theirs. The comparison results are shown in Table 5. SIGPID using only 22 significant permissions to classify different families of Apks. Compared with SIGPID, the F-M of our method is 91.6%, and the SIGPID is 98.7%. SIGPID takes 14 times as long to learn and test data as our method. Our method has higher F-M and less training and learning time. Meanwhile, if we only use chi-Square(Wang et al.,2019) of Google stated for detection. Its detection accuracy rate is 83.1%, far lower than the proposed method.

Table 5: comparison with other state-of-the-art detection approaches

Dataset	Features	Classifier	Samples (Mal./Ben.)	ACC	F-M	Time(s)
SIGPID	Permission	J48	Dataset1	94.6%	91.6%	4.5
Chi-Square	Permission	J48	Dataset1	93.1%	91.2%	3.1
Our method	**Permission**	**J48**	**Dataset1**	**95.5%**	**94.7%**	**3**

It can be seen from Figure 13 that our method is superior to chi-square in accuracy and F-measure. The accuracies of FEST (Suarez-Tangil et al.,2017) and FgDetector(Avdiienko et al.,2015) were 98% and 98.15%, respectively. These are better than our mouthed , at 95.5%. However, FEST is detected by 5 types (permission, API, action, IP and URL) of 398 typical features. FgDetector used the hardware components, requested permissions, app components, filtered intents, API calls and used permissions etc. for detection. Our method only uses 18 dangerous permissions for analysis. So, Our method has small feature dimension and high efficiency in learning and classification.

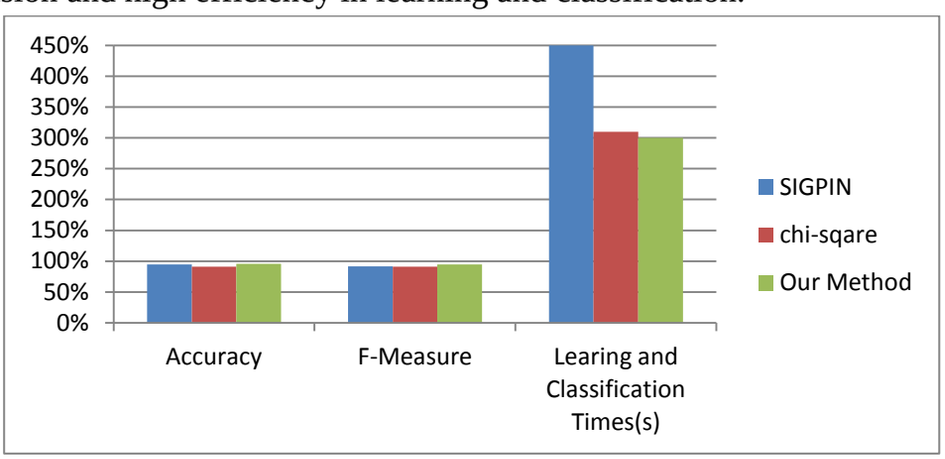


Figure 14: comparison with other state-of-the-art detection approaches

Conclusions

In this paper, we discuss the importance of android system permission in android app' security. Only use dangerous permissions or the number of used permissions can't accurately distinguish whether it is a malicious app or a benign app. Through the MCMC algorithm, MADSN calculates the uncertain value of the permission feature in the machine learning classification process. After the uncertainty analysis, 18 permission features are retained for machine learning classification. Compared with the method of directly using 24 dangerous permissions for classification, It is found that the accuracy of the proposed

model is higher under the J48 classifier. the ACC reaches 95.5%. So the proposed method has a high accuracy. According to experiments, the proposed method is also applicable to different sizes of datasets. The detection accuracies of different sizes of datasets are all higher than 88%. The proposed method is also suitable for large-scale malwares detection. For 20 common malware families, the detection accuracy of the proposed method are 95.5%. The malware detection accuracy is better than some state-of-the-art malicious detection methods. Meanwhile, the method is effective for the unknown and new apps' detection, and the accuracy of detection reaches 92.71%. Therefore, the proposed method is simple, feasible and efficient for android apps security detection.

References

1-Andreas M., Elias A., Spiros A., Demetres A., Sotiris I., & Evangelos P.(2010).Understanding the Behavior of Malicious Applications in Social Networks , IEEE Network 24(5):14 – 19

2-Avdiienko V, Kuznetsov K, Gorla A, Zeller A, Arzt S, & Rasthofer S, et al.(2015). Mining Apps for Abnormal Usage of Sensitive Data. In: Proceedings of the 37th International Conference on Software Engineering -Volume 1. ICSE '15. 426–436.

3-Cai H, Meng N, Ryder BG, & Yao D..(2019). DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling. IEEE Transactions on Information Forensics and Security.14(6):1–1.

4- Dash SK, Suarez-Tangil G, Khan S, Tam K, Ahmadi M, & Kinder J, et al.(2016). DroidScribe: Classifying Android Malware Based on Runtime Behavior. In: 2016 IEEE Security and Privacy Workshops (SPW. 252–261.

5-Du Y, Wang J, &Li Q.(2017) An Android Malware Detection Approach Using Community Structures of Weighted Function Call Graphs. IEEE Access,17478–17486.

6-Faruki P, Bharmal A, Laxmi V, Ganmoor V, Gaur MS,& Conti M, et al. (2017).Android Security: A Survey of Issues, Malware Penetration, and Defenses, IEEE Communications Surveys & Tutorials, 998–1022.

7-Hock, K., & Earle K.,(2016), Markov chain Monte Carlo used in parameter inference of magnetic resonance spectra. Entropy, 18, 57.

8-Ian H. Witten, Eibe Frank ,& Mark A.(2011), Hall Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, Burlington, MA

01803, USA.
9-Li J, Sun L, Yan Q, Li Z, Srisaan W, &Ye H..(2018). Significant Permission Identification for Machine-Learning-Based Android Malware Detection. IEEE Transactions on Industrial Informatics. .3216–3225.
10-Lu L, Li Z, Wu Z, Lee W, & Jiang G.,(2012). CHEX: statically vetting Android apps for component hijacking vulnerabilities. In: ACM Conference on Computer and Communications Security; 229–240.
11-Onwuzurike L, Mariconti E, Andriotis P, De Cristofaro E, Ross G, & Stringhini G. (2019).Ma-MaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version). ACM Transaction on Information and System Security.14.1–14.34.
12-Suarez-Tangil G, Dash SK, Ahmadi M, Kinder J, Giacinto G, & Cavallaro L..(2017). DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware. In: Proceedings of the Seventh ACM on Conference on Data and Application Security. 309–320.
13-Tuncay GS, Demetriou S, Ganju K, &Gunter CA(2018) . Resolving the Predicament of Android Custom Permissions.In: Network and Distributed System Security Symposium. 1–15.
14-Wang W, Zhao M, &Wang J..(2018) Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence & Humanized Computing.3035–3043.
15-Wang W., Zhao, M., Gao, Z., Xu, G., Xian, H., Li, Y., & Zhang, X.(2019). Constructing features for detecting android malicious applications: Issues, taxonomy and directions. IEEE Access 2019, 67602–67631.
16-Yang G, Huang J, & Gu G..(2018) Automated Generation of Event-Oriented Exploits in Android Hybrid Apps. In: Network and Distributed System Security Symposium. 1–15.
17-Zhao, K., Zhang, D.; Su, X., & Li,W.(2015). Fest: A feature extraction and selection tool for Android malware detection .IEEE Symp. Comput. Commun. 714–720.