Research Article

Risk Measurement Method of Collusion Privilege Escalation Attacks for Android Apps Based on Feature Weight and Behavior Determination

Hui Li,^{1,2} Limin Shen ,¹ Yuying Wang,¹ Jiayin Feng,^{1,3} Honglei Tan,⁴ and Zhiming Li¹

¹School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China

²School of Business Administration, Hebei Normal University of Science & Technology, Qinhuangdao 066004, China

³School of Mathematics and Information Science & Technology, Hebei Normal University of Science & Technology, Qinhuangdao 066004, China

⁴School of Continuing Education, Yanshan University, Qinhuangdao 066004, China

Correspondence should be addressed to Limin Shen; shenllmm@ysu.edu.cn

Received 10 September 2020; Revised 4 December 2020; Accepted 16 December 2020; Published 6 January 2021

Academic Editor: Salvatore D'Antonio

Copyright © 2021 Hui Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To solve the issue of measuring the risk of the application-layer collusion privilege escalation attacks in Android apps, this paper proposed a risk measurement method based on the feature weight and behavior determination. Analytic hierarchy process (AHP) is used to calculate the weight of feature in the feature set extracted from the app. App behavior and attack behavior are modeled by process algebra. The weak equivalent and nonequivalent are introduced to determine the behavior of apps, whereas the measurement function is constructed to calculate the app risk measurement value. In an experiment with three known apps, the measurement values are 0.629, 1, and 0.976. These results are consistent with reality, and the effectiveness and feasibility of the proposed method are verified. Through the benchmark and test set experiments, it can be seen that the measurement value of apps that has weak equivalent to attack behavior is distributed between 0.0468 and 1, and the measurement value distribution is reasonable, which verifies the accuracy and rationality of the method.

1. Introduction

With the rapid development of mobile Internet, the Android platform is widely used in all aspects of people's work, study, and life. Therefore, the Android platform has become a research hotspot [1]. At the same time, more and more personal privacy information is saved in smart devices, the value of devices and information is consequently increasing, and attacks against the Android system and Android apps are frequently occurring [2]. Reference [3] indicated that Android devices accounted for 47.15% of the monthly infection rate of mobile networks in 2018. In the third quarter of 2019, 365,000 new malicious programs were added to the Android platform, of which privacy theft accounted for 17.9%, and the app's unauthorized actions are increasing [4, 5]. This paper discusses the privilege escalation attacks

between multiapplications used for theft of users' privacy information through gradual refinement of permissions. Such attacks are more covert, difficult to detect through a single application, and more threatening [6].

At present, researchers use static, dynamic, and hyper methods to detect privilege escalation attacks. A method of protection against privilege escalation attacks is the improvement of the Android permission mechanism, access control mechanism, and security framework. However, for the detection and protection of collusion in privilege escalation attacks that constitute multiple applications, there is a lack of analysis and detection of communication between multiple applications, in particular, the lack of effective detection and protection of a single application. Worse yet, most existing security solutions become costly or rigid in recent Android dynamic permission environment [7].

Therefore, the risk measurement of privilege escalation attacks becomes a new solution. Although researchers also put forward many classic solutions to the risk measurement of software and apps, there are the following problems: ① risk assessment is conducted from the overall ability or third-party evaluation of the app, but the detailed features of the app are not considered and ② risk assessment of app behavior is only based on one feature, such as permission or API call, but not multiple features.

Therefore, it is especially important to measure the risk of a single application that can constitute multiapplication collusion privilege escalation attacks. Because of limitations of the current situation about the study on the risk measurement of collusion privilege escalation attacks, this paper proposes a risk measurement method for application-layer collusion privilege attacks using feature weight and behavior determination:

- (1) Construct the feature set of application-layer collusion privilege escalation attacks: based on analyzing the model of privilege escalation attack, a feature set is constructed, which is composed of six features: application's dangerous permission, component's dangerous permission, sensitive API call, sensitive data acquisition, component Intent communication, and sensitive data transition.
- (2) Feature weight calculation: AHP is used to calculate the weight of each feature in feature set, which provides basic data for app risk measurement.
- (3) App behavior modeling and equivalence relationship determination: process algebra is introduced to model the behavior and attack behavior of the app. The concept of weak equivalence and nonequivalence is used to judge the behavior of the app.
- (4) Calculate risk value of the app: based on the construction of the risk measurement function, the risk measurement is carried out for the app which is weak equivalence to the attack behavior.

The proposed method can effectively measure the degree of risk of weak equivalence apps with the attack model. The main contributions are as follows:

- (1) The attack behavior feature set is constructed, and the weight of each feature is given. Static extraction methods were used to extract six features of the app (application's dangerous permission, component's dangerous permission, sensitive API call, sensitive data acquisition, component Intent communication, and sensitive data transition), and the weights of six features are calculated by AHP. The weights were 0.033, 0.313, 0.107, 0.089, 0.150, and 0.308, respectively. It makes up for the limitation of not considering the detailed features or only one feature when evaluating the overall behavior of the app.
- (2) Behavior modeling and determination: process algebra is used to model the application behavior and

attack behavior, and weak equivalence concept is used to decide the equivalence relationship. Our method only measures the risk of the app which is weak equivalent to the attack behavior, which can save the cost of measurement.

- (3) Feature weight and behavior determination are combined to measure the risk of collusion privilege escalation attacks in the first time, and the risk measurement function is constructed to test the case, benchmark test, and actual APK test set. Through experiments and comparison, the accuracy and validity of the method are verified.

2. Related Work

Android malware detection and protection has always been a hot research topic. Because the early signature-based method for detecting malicious code [8] cannot detect the unknown malicious behavior, at present, it is mainly based on the app behavior detection method [9–13]. These methods mainly focus on the detection of malware on system calls, control flow, data flow, permission information, sensitive API, and other behavior features, and the method of multidimensional feature behavior portrait is used to detect malware. The authors of [14–17] proposed machine learning technology, deep learning technology, and unstructured user input technology for detecting the malicious Android app. Because privilege escalation attacks are characterized by strong concealment and camouflage, the detection of malicious behavior of the app cannot detect the privilege escalation attacks very well. Therefore, scholars have also proposed many theories and methods for better detection and protection. For the kernel-level privilege escalation attacks, scholars have enhanced the access control framework and monitored the permission information and other methods to prevent attacks [18, 19]. For the application-layer collusion attack discussed in this paper, the authors of [7, 20–23] proposed extended forced access strategy, monitoring system calls, and restricting dangerous interapplication communication to prevent the attacks. Although researchers have achieved good research results in the detection and protection of malware and privilege escalation attacks, collusion privilege escalation attacks involve collusion attacks between multiapplications, and the malicious detection of a single application cannot detect collusion privilege escalation attacks very well. Therefore, risk measurement becomes a new way to solve the collusion privilege escalation attacks.

Meanwhile, scholars have achieved many research results in the measurement of software security and credibility. From software requirement [24], behavior [25, 26], capability [27], and software development process [28], software credibility analysis, modeling, and measurement are carried out. The method of multiattribute decision making is widely used in engineering, technology, and economy and is also used in software credibility analysis [29, 30]. Using the

research results of software credibility measurement for reference, the research on risk measurement and credibility of Android app has made remarkable progress. The study in [31] proposed a risk assessment method based on permission and functional relationship analysis and standardizing Android developer permission applications. It can recommend reasonable permission configurations for users in real time, reducing the risk of permission abuse. However, it only considers the risk of using permissions in function implementation. And, reference [32] defined the criticality concepts of Android permissions according to the abuse of permissions by malware and put forward the standards for measuring application security risks. Although this standard has advantages, it is too simple to measure the risk of collusion privilege escalation attacks. Because this standard is primarily concerned with permission risk. Xu et al. [33] proposed an Android application security credit index measurement method based on AHP combined with the certification strength of Android software and violation records in the third-party application market. This method considers the violation records of the third party, so credibility evaluation is more objective. Different from the measurement of certification strength and violation records in reference [33], Li et al. [34] analyzed the behavior feature of the app itself and proposed a sandbox-based risk assessment method based on the app's behavior on Android to dynamically monitor and record activity. This reference, however, only focuses on the API calls of the app. The study in [35] proposed a framework layer integrity measurement method for the representation and usage of the framework layer of the Android system. This method can effectively guarantee the integrity of the framework layer code and the integrity of the runtime of the Android system. However, for the apps that can constitute collusion privilege escalation attack, because these apps are all security when measured alone, the method is not effective for collusion privilege escalation attacks.

The risk measurement of software and the app only focuses on at the macro violations or a feature, so the effect of risk measurement is not good for the application-layer collusion privilege escalation attacks. The main problems are as follows:

- (1) The multifeatures of the attack are not considered fully. The existing app risk measurement methods mainly focus on the overall behavior or a feature of the app (such as permission, API call, and so on). The attack mode of collusion attack we discussed is covert, so we need to consider many features when we evaluate the risk of collusion privilege escalation attack.
- (2) There are few measurement methods for collusion privilege escalation attacks. Because every app that constitutes collusion privilege escalation attacks is normal and safe, the existing measurement methods must fail to measure those attacks, so it is necessary

to build a special measurement method to effectively measure collusion privilege escalation attacks.

Therefore, it is particularly important to measure the risk of a single application constituting a collusion privilege escalation attacks. The semantics of process algebra is used to describe and determine the behavior of the app [24], and AHP is adopted as a decision-making method which combines qualitative and quantitative analysis [36], and then a risk measurement method of collusion privilege escalation attacks based on feature weight and behavior determination is constructed.

3. Attack Model Analysis and Feature Extraction

3.1. Attack Model Analysis. Utilizing the drawbacks of the Android system architecture in its permission mechanism and interapplication communication, multiple applications can conspire to attack. This problem of the Android platform has been described by the model of multiapplication collusion in the application-layer [37].

In the multiapplication collusion privilege escalation attack, an application component with less permission can access the components that have more permissions than it by using interapplication communication and permission refinement. In this way, the application can obtain permissions that it does not have in order to steal private data. In order to study privilege escalation attacks, three normally independent apps have been designed to construct the case of privilege escalation attacks [38]. The key codes for the three applications are presented in Table 1.

The diagram of privilege escalation attacks case can be obtained from Table 1.

As can be seen from Table 1 and Figure 1, a multi-application collusion privilege escalation attack is composed of six typical features: dangerous permission of the app, dangerous permissions of components, sensitive API calls, sensitive data acquisition, component Intent communication, and sensitive data transition.

3.2. Construction and Extraction of Feature Set. The attack behavior feature set AF is composed of six typical features of a privilege escalation attack.

$AF = \{AF1, AF2, AF3, AF4, AF5, AF6\}$ where AF1 denotes a dangerous permission of the app; AF2 denotes a dangerous permission of the component; AF3 denotes a sensitive API call; AF4 denotes a sensitive data flow acquisition; AF5 denotes a component Intent communication; and AF6 denotes a sensitive data transition.

Android APKs were decompiled by Apktool to smali files. Smali files can be analyzed to obtain information, such as class, inheritance, interface, and function call. Meanwhile, the feature set can be extracted with the information description file AndroidManifest.xml of the app.

TABLE 1: Key code of privilege escalation attack case.

	Com1 of App1	Com2 of App2	Com3 of App3
Application permission <i>AppP</i>	NULL	<uses-permission android: name = "android.permission.SEND_SMS"/>	NULL
Component permission <i>Comp</i>	NULL	app's has SEND_SMS permission	android.permission = "android.permission.SEND_SMS"
Component < intent- filter>	NULL	<action android:name = "sendInfo"/>	<action android:name = "sendInformationSMS"/>
Sensitive API calls	NULL	void enforceReceiveAndSend(java.lang.String <source, sink>	void enforceReceiveAndSend(java.lang.String <source, sink>
Sensitive data flow pairs Component Intent communication	Intent intentA = new Intent("sendInfo"); Bundle bundleA = new Bundle(); bundleA.putString("userTel", edfTel.getText().toString()); intentA.putExtras(bundleA); startActivity(intentA);	Bundle bundle = this.getIntent().getExtras(); keyStr = bundle.getString("userTel"); Intent intentB = new Intent("sendInformationSMS"); Bundle bundleB = new Bundle(); bundleB.putString("keyInfo", keyStr); intentB.putExtras(bundle2); startActivity(intentB);	Bundle bundleC = this.getIntent().getExtras(); sendStr = bundleC.getString("keyInfo"); String phoneNo = "XXXXXXXXXXXX"; String message = sendStr; SmsManagersmsManager = SmsManager.getDefault(); smsManager.sendMessage(phoneNo, null, message, null, null);
Number of sensitive data transitions	2	1	1

FIGURE 1: Diagram of a constituted privilege escalation attack.

3.2.1. Dangerous Permission of the Application. Extract the permission according to the tag `<uses-permission>` in the `AndroidManifest.xml`. Then, summarize the dangerous permissions of the app according to the dangerous permission list provided by Google.

3.2.2. Dangerous Permission of the Component. Extract the registration component information and the component's `<intent-filter>` information according to the `<activity>` etc. tag in the `AndroidManifest.xml`. Then, extract the permissions applied by the component according to "android: permission = XXXX," and finally, summarize the dangerous permissions applied by the component according to the dangerous permission list.

3.2.3. Sensitive API Call. Extract system call sequence with strace of Android SDK. Then, combine it with conclusions from reference [39] to obtain sensitive API calls.

3.2.4. Sensitive Data Flow Acquisition. Obtain sensitive information source `<source>` by using FlowDroid [40], app resource files, and smali files.

3.2.5. Component Intent Communication. Extract the action name of components according to `<intent-filter>` in 3.2.2, and obtain component Intent communication information according to the function calls (e.g., `getExtras` and `putExtras`) in smali file.

3.2.6. Sensitive Data Transition. Extract the flow path from source to sink by FlowDroid [40], and obtain sensitive data transition. The source acquires sensitive information while the sink sends sensitive information.

4. Feature Weight Calculation Based on AHP

AHP [41] can analyze the weight of the factors related to decision-making and can effectively distinguish the influence degree of different features in the process of constructing the pairwise comparison matrix A . AHP can be well applied to the analysis of the features of collusion privilege escalation attacks with complex attack features and strong concealment. At the same time, reference [33] shows that AHP can effectively analyze the influencing factors.

4.1. AHP Model Introduction. Two important steps in this method are the construction of the pairwise comparison matrix and consistency test.

4.1.1. Construction of Pairwise Comparison Matrix

- (1) There are n factors at a certain level, $X = \{x_1, x_2, \dots, x_n\}$; in order to compare the influence of X on a certain objective, the degrees of influence of n factors on a certain objective are ranked. The quantified relative weight a_{ij} is introduced to describe the comparison result of factors i and j for the objective. Suppose that there are n factors in the comparison. Then, the pairwise comparison matrix $A = (a_{ij})_{n \times n}$ can be described as follows:

$$A = (a_{ij})_{n \times n} = \begin{Bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{Bmatrix}, \quad (1)$$

where a_{ij} ranges from 1 to 9 and the corresponding reciprocal. ① $a_{ij} = 1$ means that factor i is equal to factor j in influence; ② $a_{ij} = 3$ means that factor i is slightly stronger than factor j in influence; ③ $a_{ij} = 5$ means that factor i is stronger than factor j in influence; ④ $a_{ij} = 7$ means that factor i is obviously stronger than factor j in influence; ⑤ $a_{ij} = 9$ means that factor i is absolutely stronger than factor j in influence; and ⑥ $a_{ij} = 2n$, $n = 1, 2, 3, 4$, means the influence of factor i relative to factor j is between $a_{ij} = 2n - 1$ and $a_{ij} = 2n + 1$.

- (2) The maximum eigenvalues and eigenvectors can be obtained by the column normalizing the pairwise comparison matrix. The weight of the factors can be determined after the consistency test.

4.1.2. Consistency Test. Consistency index, random consistency index, etc. were used to calculate the consistency ratio so as to test the matrix A

- (1) Consistency index (CI)

$$CI = \frac{\lambda - n}{n - 1}, \quad (2)$$

where n is the sum of the diagonal elements of the matrix A and λ is the maximum eigenvalue.

(2) Random consistency index (RI)

The values of RI are presented in Table 2.

(3) Consistency ratio (CR)

$$CR = \frac{CI}{RI}. \quad (3)$$

If $CR \leq 0.1$, the degree of inconsistency of the pairwise comparison matrix A is within the acceptable range, and its normalized eigenvectors represent the weight vector. If $CR > 0.1$, the pairwise comparison matrix A needs to be reconstructed.

4.2. Feature Weight Calculation

4.2.1. Construction of Pairwise Comparison Matrix. According to the method introduced in Section 4.1, the scale comparisons of features in AF are presented in Table 3.

According to Table 3 and formula (1), the pairwise comparison matrix can be created as follows:

$$A = \begin{pmatrix} 1 & \frac{1}{7} & \frac{1}{5} & \frac{1}{3} & \frac{1}{5} & \frac{1}{7} \\ 7 & 1 & 5 & 3 & 2 & 1 \\ 5 & \frac{1}{5} & 1 & 1 & 1 & \frac{1}{3} \\ 3 & \frac{1}{3} & 1 & 1 & \frac{1}{3} & \frac{1}{3} \\ 5 & \frac{1}{2} & 1 & 3 & 1 & \frac{1}{3} \\ 7 & 1 & 3 & 3 & 3 & 1 \end{pmatrix}. \quad (4)$$

4.2.2. Consistency Verification and Weight Calculation. By normalizing the column vectors of formula (4), the following results were obtained.

(1) Maximum eigenvalue

$$\lambda_{\max} = 6.228. \quad (5)$$

(2) Maximum eigenvector

$$\omega = \begin{pmatrix} 0.033 \\ 0.313 \\ 0.107 \\ 0.089 \\ 0.150 \\ 0.308 \end{pmatrix}. \quad (6)$$

(3) Consistency verification

According to formulas (2) and (5) with $n=6$, CI is calculated as follows:

$$CI = \frac{6.228 - 6}{6 - 1} = \frac{0.228}{5} = 0.046. \quad (7)$$

Then, from Table 2, it can be seen that $RI = 1.24$. According to formulas (3) and (7), CR is calculated as follows:

$$CR = \frac{0.046}{1.24} = 0.037 < 0.1. \quad (8)$$

According to formula (8), it is verified that A is consistent.

Therefore, weight vector ω is obtained according to formula (6), and the weight of each feature in the feature set is presented in Table 4.

5. Behavior Modeling Based on Process Algebra

Process algebra [42] can effectively describe the interaction between two systems and determine the equivalence between them. In process algebra, the behavior of the system can be represented by actions and events. Therefore, it can effectively describe the permissions and message communication of the Android app. The Android app is composed of components; the behavior of components constitutes the behavior of the app. Therefore, according to the features of components, the behavior model of components and the behavior of the app can be built. At the same time, references [24, 25] show that process algebra can effectively determine software behavior.

5.1. Application Behavior Model. The Android app is composed of a series of components that can help users implement functions and services, such as interface building, data storage, and background running. The Android system consists of four major components, namely, Activity, Service, BroadcastReceiver, and ContentProvider. Activities are used to present functions and interface presentations. Services provide background running services. BroadcastReceiver is used to receive broadcasts, and ContentProvider supports data storage and reading. Therefore, the behavior of the application consists of the behavior of all components above. The syntax and semantic specifications of process algebra based on the behavior features of components are defined as follows:

TABLE 2: RI values.

n	1	...	5	6	7	...	11
RI	0	...	1.12	1.24	1.32	...	1.51

TABLE 3: Feature scale comparison table.

Feature i /feature j ($1 \leq i, j \leq 6$)	Scale comparison
Dangerous permission of app/dangerous permission of component	1/7
Dangerous permission of app/sensitive API calls	1/5
Dangerous permission of app/sensitive data flow acquisition	1/3
Dangerous permission of app/component Intent communication	1/5
Dangerous permission of app/sensitive data transition	1/7
Dangerous permission of component/sensitive API calls	5/1
Dangerous permission of component/sensitive data flow acquisition	3/1
Dangerous permission of component/component Intent communication	2/1
Dangerous permission of component/sensitive data transition	1/1
Sensitive API calls/sensitive data flow acquisition	1/1
Sensitive API calls/component Intent communication	1/1
Sensitive API calls/sensitive data transition	1/3
Sensitive data flow acquisition/component Intent communication	1/3
Sensitive data flow acquisition/sensitive data transition	1/3
Component intent communication/sensitive data transition	1/3

TABLE 4: Feature weights.

	AF1	AF2	AF3	AF4	AF5	AF6
Weights	0.033	0.313	0.107	0.089	0.150	0.308

$$\text{comBehavior} ::= \text{com}(\text{id}) | \sum_{\substack{i \in v \\ j \in w}} P_i \cdot \text{Feature}_j | \text{Feature}_1 | \text{Feature}_2 | \dots | \text{Feature}_w$$

$$| (\overline{X} < \text{data}_1, \text{data}_2, \dots, \text{data}_n > | X(\text{data}_1, \text{data}_2, \dots, \text{data}_n)) | (\text{Feature})P | !\text{Feature}_j. \quad (9)$$

In the formula,

- (1) $\text{com}(\text{id})$ means that each component has a unique process identifier id that can be omitted according to actual condition.
- (2) $\sum_{\substack{i \in v \\ j \in w}} P_i \cdot \text{Feature}_j$ is a summation, where w represented the number of features in AF, v represented the number of permissions, and P represented the permission for the app. $P_i \cdot \text{Feature}_j$ indicating that Feature_j is protected by P_i , and Feature_j must occur before P_i can start activity.
- (3) $\text{Feature}_1 | \text{Feature}_2 | \dots | \text{Feature}_w$ means component has w features.
- (4) $(\overline{X} < \text{data}_1, \text{data}_2, \dots, \text{data}_n > | X(\text{data}_1, \text{data}_2, \dots, \text{data}_n))$, where $\overline{X} < \text{data}_1, \text{data}_2, \dots, \text{data}_n >$ means the action of message sending and $X(\text{data}_1, \text{data}_2, \dots,$

$\text{data}_n)$ means the action of message receiving. $\text{data}_1, \text{data}_2, \dots, \text{data}_n$ represent n sensitive data.

- (5) $(\text{Feature})P$ means that the application behavior represented by Feature is under P protection.
- (6) $!\text{Feature}_j$ means the Feature_j can be reused.

5.2. Privilege Escalation Attack Model. This paper focuses on the multiapplication collusion privilege escalation attack. Dangerous behavior occurs when dangerous components apply for dangerous permission. Therefore, the multi-application collusion privilege escalation attack must be based on a certain dangerous permission and be completed by a component in the application. According to formula (9), the component attack behavior under permission P_1 can be defined as follows:

$$\text{attackBehavior} ::= \sum_{i \in 6} P_1. \text{AFi} | \text{AF1} | \text{AF2} | \dots | \text{AF6} | (\overline{X} < \text{data}_1, \text{data}_2, \dots, \text{data}_w > | X(\text{data}_1, \text{data}_2, \dots, \text{data}_w)) | (\text{AF})P_1. \quad (10)$$

Here,

- (1) $\sum_{i \in 6} P_1. \text{AFi}$ denotes the attack behavior set under P_1 and w represented the number of features in AF, $0 \leq w \leq 6$.
- (2) $\text{AF1} | \text{AF2} | \dots | \text{AF6}$ means that the component has multiple features.
- (3) $(\overline{X} < \text{data}_1, \text{data}_2, \dots, \text{data}_w > | X(\text{data}_1, \text{data}_2, \dots, \text{data}_w))$, where $\overline{X} < \text{data}_1, \text{data}_2, \dots, \text{data}_w >$ denotes the action of message sending of the component and $X(\text{data}_1, \text{data}_2, \dots, \text{data}_w)$ denotes the action of message reception of the component. $\text{data}_1, \text{data}_2, \dots, \text{data}_w$ represent w sensitive data.
- (4) $(\text{AF})P_1$ means that the behavior represented by AF in the application component is protected by permission P_1 .

6. Risk Measurement Method

According to the equivalence relationship of states in process algebra, two behavioral equivalence relationships are defined: behavior weak equivalence relationship and behavior nonequivalence relationship.

6.1. Behavior Transition. A set $\text{Act} = \{a_1, a_2, \dots, a_t, a'_1, a'_2, \dots, a'_t\}$ of actions, a state set $Q = \{(a_1, P), (a_2, P), \dots, (a_t, P), (a'_1, P), (a'_2, P), \dots, (a'_t, P)\}$ of states, and there exists a subset T of $Q \times \text{Act} \times Q$ called the transitions. That is, $(a_i, P), (a'_i, P) \in Q, a \in \text{Act}$ (or $a' \in \text{Act}$), then $(a_i, P) \times a \times (a'_i, P)$ (or $(a_i, P) \times a' \times (a'_i, P)$), called $(a_i, P) \xrightarrow{a} (a'_i, P)$ transition (a'_i, P) , witten as $(a_i, P) \xrightarrow{a} (a'_i, P)$ (or $(a_i, P) \xrightarrow{a'} (a'_i, P)$).

6.2. Behavior Weak Simulation. Suppose $S = \{((a_1, P), (a'_1, P), (a_2, P), (a'_2, P)), \dots, ((a_t, P), (a'_t, P))\}$ is a binary relation in the state space $\mathcal{P}$ of a component, $P, P', Q, Q' \in \mathcal{P}$, $\xrightarrow{\lambda}$ represents any action by a feature of a component. S is a weak simulation if and only if, whenever PSQ , if $P \xrightarrow{\lambda} P'$, then there exists Q' such that $Q \xRightarrow{\lambda} Q'$ and $P'SQ'$; if $P \xrightarrow{\lambda} P'$, then there exists Q' such that $Q \xRightarrow{\lambda} Q'$ and $P'SQ'$. This means that state P has passed a finite number of state transitions (even no transitions) to state Q .

Definition 1. (Risk behaviors' weak equivalence). The binary relation S over process space $\mathcal{P}$ is weakly equivalence only if S and its converse are both weak simulation. It is said that P and Q are weakly equivalent or weakly bisimulate if a weak bisimulation S exists such that PSQ , written as $P \approx Q$, where $P \in \text{comBehavior}$ and $Q \in \text{attackBehavior}$. This means that there is certain equivalence between the component behavior in the application and attack behavior. Therefore, it is necessary to establish a measurement function for risk assessment.

Definition 2. (Risk behaviors' nonequivalence). Whether to select any transition P , the matching transition Q cannot be found, that is, there is no strong (weak) simulation R . Then, P and Q are said to be not equivalent and written as $P \neq Q$. This means that there is no equivalence between the component behavior of the application and attack behavior. Therefore, its risk measurement value is 0.

Definition 3 (Risk behaviors' measurement function).

$$f(V, W) = \begin{cases} \log_2 \sum_{i,j=1}^6 (1 + V_i \cdot W_j), & P \approx Q, 0, P \neq Q. \end{cases} \quad (11)$$

Here,

- (1) V_i indicates whether the feature exists in the feature set. The corresponding feature of V_i is AFi , $0 \leq i \leq 6$. $\text{AFi} = T$ indicates that the i -th feature exists, and $\text{AFi} = F$ indicates that the i -th feature does not exist.

$$V_i = \begin{cases} 1, & \text{AFi} = T, \\ 0, & \text{AFi} = F. \end{cases} \quad (12)$$

- (2) W_j denotes the weight of the feature in the feature set, $W_1 + W_2 + W_3 + W_4 + W_5 + W_6 = 1$, and $0 \leq j \leq 6$.

Algorithm 1: app behavior determination and risk measurement. AF_i denotes the feature set of the i -th app, $1 \leq i \leq n$, n denotes the number of apps in the test set, and P_1 denotes a dangerous permission.

7. Experiment for Case

7.1. Feature Set Extraction. According to the attack case given in Section 3.1, AF of App1, App2, and App3 can be obtained by using the feature extraction method from Section 3.2, as presented in Table 5. T means AFi exists, F means AFi does not exist, and $0 \leq i \leq 6$.

7.2. Risk Behavior Modeling and Risk Measurement

7.2.1. Behavior Modeling. According to formula (9), the behavior models of components Com1, Com2, and Com3 can be obtained.

- (1) The behavior model of the component Com1 of App1 is presented in formula (13), where data_1 denotes sensitive data:

$$\text{App1Com1} ::= \text{Intent} | (X(\text{data}_1) | \overline{X} < \text{data}_1 >). \quad (13)$$


```

(1) Input:  $AF_i, P_1, comBehaviorModel_i, comAttackModel_i, f(V, W)$ 
(2) Output:  $measurmentValue$ 
(3) Assumption: use BM point to  $comBehaviorModel$ , use AM point to  $comAttackModel$ 
(4) Initialization:  $BM \rightarrow comBehaviorModel_1, AM \rightarrow comAttackModel_1$ 
(5) For each component has  $AF_i$ 
(6) Construction  $comBehaviorModel_i, comAttackModel_i$ 
(7) If  $BM \approx AM$  Then
(8) call  $f(V, W)$ 
(9) print  $measurmentValue = f(V, W)$ 
(10)  $BM \rightarrow Next, AM \rightarrow Next$ 
(11) ElseIf  $BM \neq AM$  Then
(12) print  $measurmentValue = 0$ 
(13)  $BM \rightarrow Next, AM \rightarrow Next$ 
(14) EndIF

```

ALGORITHM 1: App behavior determination and risk measurement.

- (2) The behavior model of the component Com2 of App2 is presented in formula (14), where data1 denotes sensitive data and P_1 denotes that the application has dangerous permissions:

$$App2Com2 ::= P_1.API + P_1.Flow + P_1.Intent + P_1.transition | (X(data1) | \bar{X} < data1 >). P_1 | AppP_1 | ComP_1. \quad (14)$$

- (3) The behavior model of the component Com3 of App3 is presented in formula (15), where data1 denotes sensitive data and P_1 denotes that the application has dangerous permissions. $ComP_1$ denotes that the component has dangerous permissions:

$$App3Com3 ::= P_1.API + P_1.Flow + P_1.Intent + P_1.transition | (X(data1) | \bar{X} < data1 >). P_1 | ComP_1. \quad (15)$$

7.2.2. *Attack Modeling.* The attack behavior models of components Com1, Com2, and Com3 can be obtained using formula (10).

- (1) The attack behavior model of the component Com1 of App1 is presented in the following formula:

$$App1Com1Attack ::= AF4 | (\bar{X} < y_1 > | X(y_1)). \quad (16)$$

- (2) The attack behavior model of the component Com2 of App2 is presented in the following formula:

$$App2Com2Attack ::= \sum_{j=2}^6 P_1.AF_j | (\bar{X} < y_1 > | X(y_1)) | (v\chi)P_1 \\ ::= P_1.AF2 + P_1.AF3 + P_1.AF4 + P_1.AF5 + P_1.AF6 | (x(y_1) | \bar{x} < y_1 >). P_1 | P_1. \quad (17)$$

- (3) The attack behavior model of the component Com3 of App3 is presented in the following formula:

$$App3Com3Attack ::= \sum_{j=2}^6 P_1.AF_j | (\bar{X} < y_1 > | X(y_1)) | (v\chi)P_1 \\ ::= P_1.AF2 + P_1.AF3 + P_1.AF4 + P_1.AF5 + P_1.AF6 | (x(y_1) | \bar{x} < y_1 >). P_1. \quad (18)$$

7.2.3. *Equivalence Relationship Verification.* The behavior of the app can be composed of its feature actions, feature states, and state transition. For example, the app has the behavior of message sending. The feature is message sending permission of the app. Action of the feature is waiting to message sending and sending message. Status is do not send SMS and

send SMS. Transition is confirmed to send. Based on the definition of behavior transition, it can be seen that ① P_{sms} represents sending message permission, ② a_1 waits for sending message under P_{sms} , ③ a_1 sends message under P_{sms} , ④ (a_1, P_{sms}) represents the action of waiting for sending message, ⑤ (a'_1, P_{sms}) represents the action of

TABLE 5: App1, App2, and App3 feature set.

	AF1	AF2	AF3	AF4	AF5	AF6
App1	F	F	F	T	T	T
App2	T	T	T	T	T	T
App3	F	T	T	T	T	T

sending message, and $\odot \lambda$ represents status transition. Then,

$$(a_1, P_{\text{sms}}) \xrightarrow{\lambda} (a'_1, P_{\text{sms}}). \quad (19)$$

According to the definition of weak simulation, the state set $\mathcal{P}$ of the component is composed of the state sets similar to the left and right sides of formula (19). Let P and P' are the behavior state set, Q and Q' are the attack behavior state set, and the attack behavior state set of components must be included in the behavior state set. Based on the above theory, the weak equivalence between formulas (14) and (17) is verified by derivation.

For the convenience of description, Tables 6 and 7 are used to simplify the states represented by related features.

Construct the state transition diagram of formulas (14) and (17), as shown in Figures 2(a) and 2(b), where a, b, c , and d represent migrations.

Suppose that $S = \{(q_0, p_0), (q_1, p_1), (q_1, p'_1), (q_2, p_2), (q_3, p_3), (q_4, p_4)\}$. Then,

- (1) p_0 weak simulation q_0 verification. $\forall (q, p) \in S$, each migration of the first element q , $q_i \xrightarrow{a} q_j$, $0 \leq i, j \leq 4$ can be migrated $p_i \xrightarrow{a} p_j$, $0 \leq i, j \leq 4$ by a second element p matching (or a series of migration or even no migration). For example, for (q_1, p'_1) , q_1 has $q_1 \xrightarrow{c} q_3$, so $\exists (q_3, p_3) \in S$, using $p'_1 \xrightarrow{c} p_3$ matching. Therefore, s is a weak simulation, that is, p_0 weak simulation q_0 .
- (2) In the same manner, the weak simulation relationship of each state in two state graphs can be verified.
- (3) According to the concept of risk behaviors' weak equivalence Definition 1, S^{-1} can be verified as a weak simulation in the way of (1).
- (4) Therefore, the weak equivalence between the states in Figures 2(a) and 2(b) can be verified. Then, the weak equivalence between formulas (14) and (17) is verified.

To make our verification more automated, the Mobility Workbench (MWB) was chosen as the verification tool. After standardization according to MWB grammar [43], the above application model was transformed into MWB language. The behavior model and the attack model of App3Com3 (representation in MWB with AC) were proved to have a weak equivalence relationship. Figure 3 shows the verification process of weak equivalence between the behavior and attack behavior of App3Com3 implemented by MWB. Similarly, the following two weak equivalence relationships were also verified: (1) App1Com1 and App1Com1Attack and (2) App2Com2 and App2Com2Attack.

7.2.4. Extended Attack Modeling. In order to describe the presented method, it is assumed that there is no sensitive data flow for Com1 of App1.

- (1) The behavior model of the feature changed in App1Com1 is presented in the following formula:

$$\text{App1ComChange: } ::= \text{Intent}[\overline{X} < \text{data1} > . \quad (20)$$

- (2) The nonequivalence relationship is verified. According to the concept of risk behaviors' nonequivalence Definition 2, Figure 4 shows the verification process of nonequivalence between the change behavior (represented by ACC) and attack behavior (represented by ACA) of App1Com1 implemented by MWB.

Therefore, according to Definition 3, the App1 measurement value of the changed feature is 0.

7.3. Risk Measurement for Case. It can be seen from the result in Section 7.2 that App1, App2, and App3 are weakly equivalent to the attack model. Therefore, according to the measurement method, formulas (11) and (12), and the feature set of Table 5, it is possible to obtain the feature existence matrix $B_{3 \times 6}$ which that the existence of features in the feature set of Com1, Com2, and Com3.

$$B_{3 \times 6} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \quad (21)$$

The result of multiplying formula (21) by weight vector ω is as follows:

$$B \times \omega = \begin{pmatrix} 0.547 \\ 1 \\ 0.967 \end{pmatrix}. \quad (22)$$

Based on formulas (11) and (22), the measured results of App1, App2, and App3 are shown in Table 8.

Therefore, we have the following:

- ① The measured results of App1, App2, and App3 are 0.629, 1, and 0.976.
- ② App2 and App3 are more dangerous than App1. The measurement value is consistent with the actual risk of the app.
- ③ App1 is used to collect sensitive information in a collusion privilege escalation attack. For a single app, its risk is indeed weaker than App2 and App3.

8. Method Evaluation and Validity Analysis

8.1. Method Evaluation. This proposed method can be divided into three steps: construction, extraction of behavior feature set and feature weight calculation, and app dangerous behavior modeling and decision and risk measurement. Among them, feature extraction is the basic of the method. The key steps are sensitive data flow and migration detection. The time complexity of the dangerous behavior

TABLE 6: Symbol simplification of formula (14).

	Formula content	Simplified symbols
Dangerous permissions of the application	App P_1 (P_1)	p_0
Dangerous permissions of the components	Com P_1	p_1
Sensitive API calls	API	p'_1
Component Intent communication	Intent	p_2
Sensitive data flow acquisition	Flow	p_3
Sensitive data transition	Transition	p_4

TABLE 7: Symbol simplification of formula (17).

	Formula content	Simplified symbols
Dangerous permissions of the components	AF $_2$ (P_1)	q_0
Sensitive API calls	AF $_3$	q_1
Component Intent communication	AF $_5$	q_2
Sensitive data flow acquisition	AF $_4$	q_3
Sensitive data transition	AF $_6$	q_4

FIGURE 2: (a) State transition diagram of formula (14) and (b) formula (17).

```

MWB:agent AC10(x,y)=(data1).y(data1).AC10(x,y)
MWB:agent ACA10(x,y)=(x(y1).y(y1).ACA10(x,y)
MWB:weq AC10 ACA10
The two agents are equal.
Bisimulation relation size = 2.
MWB:agent AC11(x,y)=(p1) (AC10(x,p1) | AC10(p1,y))
MWB:agent ACA11(x,y)=(p1) (ACA10(x,p1) | ACA10(p1,y))
MWB:weq AC11 ACA11
The two agents are equal.
Bisimulation relation size = 13.
MWB:agent AC12(x,y)=(api) (AC10(x,api) | AC10(api,y))
MWB:agent ACA12(x,y)=(af2) (ACA10(x,af2) | ACA10(af2,y))
MWB:weq AC12 ACA12
The two agents are equal.
Bisimulation relation size = 13.
MWB:agent AC13(x,y)=(flow) (AC10(x,flow) | AC10(flow,y))
MWB:agent ACA13(x,y)=(af3) (ACA10(x,af3) | ACA10(af3,y))
MWB:weq AC13 ACA13
The two agents are equal.
Bisimulation relation size = 13.
MWB:agent AC14(x,y)=(intent) (AC10(x,intent) | AC10(intent,y))
MWB:agent ACA14(x,y)=(af4) (ACA10(x,af4) | ACA10(af4,y))
MWB:weq AC14 ACA14
The two agents are equal.
Bisimulation relation size = 13.
MWB:agent AC15(x,y)=(transition) (AC10(x,transition) | AC10(transition,y))
MWB:agent ACA15(x,y)=(af5) (ACA10(x,af5) | ACA10(af5,y))
MWB:weq AC15 ACA15
The two agents are equal.
Bisimulation relation size = 13.

```

FIGURE 3: Validation of weak equivalence relationship.

determination algorithm is $O(n)$. The calculation of measurement function mainly focuses on multiplication, addition, and logarithm. For each app which is weakly equivalent to the attack behavior, the computational cost of risk measurement mainly comes from $B_{3 \times 6} \times \omega_{6 \times 1}$ matrix operation; there are (18 times of multiplication + 16 times of addition + 1 times of logarithm) $\times$ number of apps. Fifty-three Android APKs were tested, and the time and space cost analysis data are shown in Table 9. All experiments were implemented on a PC platform with 8G memory and Intel(R) core (TM) i5-2520M CPU.

8.2. Benchmark Test and Method Comparison. DroidBench [44] is a set of open source real-life Android applications to be used as a testing ground for static and dynamic security and measurement methods. FieldAndObjectSensitivity, InterAppCommunication, and InterComponentCommunication are test set in DroidBench. The FieldAndObjectSensitivity has the sensitive data leakage, the InterAppCommunication has the information leakage in the communication between apps, and the InterComponentCommunication has the information leakage between components. The features of collusion privilege escalation attacks that may exist in the three test sets are shown in Table 10, where T represents the existence of the feature and F represents no-existence.

Table 10 shows that test sets cover all attacks features in AF. Different test objects in the test sets have different attack features. For example, some of the test objects in InterComponentCommunication do not exist for AF4 and AF5 features. The method in Sections 3–6 and formula (11) are used to verify the behavior equivalence and measure the risk. Table 11 shows the AF sets of each test object obtained by this method, the equivalence relationship, $\sum_{i,j} V_i * W_j$, and the measurement results, where T represents the existence of the feature and F represents no-existence. At the same time, the method proposed in reference [45] is used to measure the test objects, and the measurement results are shown in Table 11.

The measurement values of 22 benchmark objects show that the following:

```

MWB>agent ACC10(x)=x(data1).ACC10(x)
MWB>agent ACA10(x,y)=x(y1),y<y1>.ACA10(x,y)
MWB>agent ACA11(x,y)=(af4)(ACA10(x,af4)|ACA10(af4,y))
MWB>eq ACC10 ACA11
The two agents are NOT equal.
MWB>weq ACC10 ACA11
The two agents are NOT equal.
MWB>

```

FIGURE 4: Validation of nonequivalence relationship.

TABLE 8: Measures of App1, App2, and App3.

App's name	Measurement value
App1	$\log_2(1 + 0.547) = 0.629$
App2	$\log_2(1 + 1) = 1$
App3	$\log_2(1 + 0.967) = 0.976$

TABLE 9: The time and space cost.

	Maximum	Minimum	Average
Time cost (S)	2.182	242.399	28.45181132
Space cost (S)	916.314584	36.084672	633.800664

TABLE 10: Attack features in test set.

Test set	AF1	AF2	AF3	AF4	AF5	AF6
FieldAndObjectSensitivity	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>
InterAppCommunication	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
InterComponentCommunication	<i>T</i>	<i>T</i>	<i>T</i>	<i>T/F</i>	<i>T/F</i>	<i>T</i>

- (1) The measurement value is between 0.8875 and 1, the distribution of measurement value is reasonable, and it matches the actual risk degree of the app.
- (2) The three test sets have information leakage problems of sensitive data, communication between apps, and communication between components. From the measurement values, it can be seen that apps with information leakage risk pose a greater risk of privilege escalation attacks. It can be verified that it is necessary to calculate the weight of different features by the relationship of $\sum_{i,j} V_i * W_j$ and measurement value.

By comparing with the measurement results in reference [45], the following can be seen:

- (1) The distribution of measurement values in our method is more reasonable, which can better reflect the actual risk degree of apps. The measurement results of reference [45] are concentrated between 0.9975 and 0.9999, and the distribution is relatively concentrated, which cannot better distinguish the dangerous degree of these high-risk apps.
- (2) The measurement results of our method are consistent with the attack behavior features. In this method, sensitive data transition is added to

the attack behavior feature set as attack behavior feature, which ensures the consistency of risk measurement value of the app with the same attack behavior features. However, in reference [45], the sensitive data transition was not considered in attack behavior feature set, but sensitive data transition was taken as a parameter in the measurement function so that it has the same attack behavior features but different measurement results, for example, the measurement results of objectsensitivity 1 and objectsensitivity 2 in Table 11.

- (3) Both methods ensure the effective measurement of the high-risk app. Through the measurement value of 22 benchmark test objects, it can be seen that the two methods can effectively reflect the risk degree of the app when they measure the app with high risk.

8.3. Test Set Composition. In order to ensure the diversity and comprehensiveness of the test set, 50 apps from the Android app market and 3 apps developed by the research team were selected for testing the feasibility and effectiveness of the proposed method [46]. There are 21 categories of samples, and each category contains 1–3 typical applications. The sample composition is presented in Table 12.

TABLE 11: Benchmark test measurement results.

Test sets and test objects	AF								Equivalence relationship	$\sum_{i,j} V_i * W_j$	Measurement results	Measurement results using reference [45]
	AF1	AF2	AF3	AF4	AF5	AF6						
FieldAndObjectSensitivity	FieldSensitivity1	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	FieldSensitivity2	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	FieldSensitivity3	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	FieldSensitivity4	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	ObjectSensitivity1	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	ObjectSensitivity2	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.999088118
InterAppCommunication	SendSMS	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.99987659
	StartActivityForResult	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.9999546
InterComponentCommunication	ActivityCommunication1	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	ActivityCommunication2	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	ActivityCommunication3	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	ActivityCommunication4	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	ActivityCommunication5	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	ActivityCommunication6	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	ActivityCommunication7	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	ActivityCommunication8	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999664537
	BroadcastTaintAndLeak1	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999088118
	IntentSink1	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999088118
	IntentSink2	T	T	T	T	T	T	Weak equivalence	1.000	1.000000000	1.000000000	0.999088118
	IntentSource1	T	T	T	F	T	T	Weak equivalence	0.911	0.934327778	0.934327778	0.997521248
	ServiceCommunication1	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248
	SharedPreferences1	T	T	T	T	F	T	Weak equivalence	0.850	0.887525271	0.887525271	0.997521248

TABLE 12: Test set classification.

Category Number	Office 3	Lifestyle 3	Shopping 3	Smart home 2	Medical 3	Finance 3	Education 3
Category Number	Picture 3	Books 1	System utilities 3	News 3	Home care 3	Language learning 3	Sport and fitness 3
Category Number	Browser 1	Travel 2	Entertainment 2	Social networking 3	Theme wallpaper 1	Theme plugin 2	Research team 3

TABLE 13: Percentage of the same company or developer.

	Same company or developer	Different company or developer
Number	13	40
Percentage	24.5	75.5

Because the same company or developer has the same development methods and coding habits, it is easy to cause the same defects in the developed apps, and the convenient conditions constitute the attack behavior among the developed apps. Therefore, the test set contains 13 apps developed by the same company or developer, as shown in Table 13.

8.4. Validity Analysis. The method presented in Section 3.2 was applied to extract the feature set of the APK in the test set. The method of behavioral modeling presented in Section 5 was then used to obtain the equivalence relationship of the application. The partial APKs from test set feature set description and results are provided in Table 14.

According to the measure algorithm and formula (11), the measurement values of APKs in Table 14 can be obtained. They are listed in descending order in Table 15.

The equivalence relationship of the application in the test set can be obtained by employing the behavior modeling method presented in Section 5. Its distribution is presented in Table 16.

The results of the equivalence relationship are consistent with the conclusions of [20, 47].

The risk measurement is carried out for the APK with weak equivalence relationship in Table 16 using this proposed method. Figure 5(a) shows the curve figure between the measurement value and the number of features in AF, and Figure 5(b) shows the measurement value of 39 APKs.

The method proposed in reference [45] is used to measure those 39 APKs, and the measurement results are shown in Figure 6. The measurement values are mainly concentrated in the range of 0.8647–1. Compared with the proposed method, the distribution of measurement values is too centralized.

Among the 13 APKs involved in Tables 13, 10 are weakly equivalent to attack behavior. Among them, the measurement value of 3 APKs is 1 and that of 1 APK is 0.967. The detailed results are shown in Figure 7.

Figures 5–7 show the following:

- (1) The ranges of measurement values are reasonable, which conforms to the risk degree of APK. From the comparison between Figures 5(b) and 6, it can be

seen that the measurement value of weak equivalent APKs to attack behavior is in range from 0.0468 to 1 by using the proposed method. And, the measurement value is distributed between 0.8647 and 1 by using the method in reference [45]. Therefore, the measurement value distribution of the proposed method is more reasonable. It also shows that the weak equivalent APK has a strong or weak risk measurement, which is consistent with the actual of APK risk.

- (2) The consideration of feature weight is more important. It can be seen from Figure 5(a) that the higher the number of features in AF, the higher the risk value of APK. Meanwhile, the specificity in this figure also proves the importance of feature weight. That is to say, there are many features in this figure, but the measure value is not as high as the APK with fewer features. Therefore, it is more important to consider the weight of each feature. For the features in AF, not only the number but also the weight of each feature should be considered. However, reference [45] does not consider the feature weight and it only considers the number of features. From the measurement values in Figure 6, it can be seen that the differentiation of risk degree is not obvious.
- (3) Apps developed by the same company or developer are more threatening. According to the conclusion of Figure 7 and reference [45], it can be seen from Figure 7 that 76.9% (among the 13 APKs of the same company or developer, 10 of them are weakly equivalent, accounting for 76.9%) of APKs developed by the same company or developer have collusion privilege escalation attack risk, accounting for 25.6% (among the 39 weakly equivalent APKs, 10 of them belong to the same company or developer, accounting for 25.6%) of the risk APKs in the test set, and their measurement values are high. Therefore, for the same company or developer development, it is necessary to strengthen code management and security education to prevent the occurrence of collusion privilege escalation attack due to development loopholes and intentional acts.

TABLE 14: The feature set description of partial APK.

Package names	AF1	AF2	AF3	AF4	AF5	AF6	Equivalence relationship
ssxy.kaoshi.namespace	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	Weak equivalence
com.invariantlabs.winder	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	Weak equivalence
com.damaiwan.dmapp	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	Weak equivalence
jjszjsgsl.kaoshi.namespace	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	Weak equivalence
com.example.healthmonitorlh	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	Weak equivalence
com.example.pinganpuhui	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	Weak equivalence
com.tupo.mi.kaoyan.vocabulary	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	Nonequivalence
—	—	—	—	—	—	—	—

TABLE 15: The measurement value of partial APK.

Package names	AF1	AF2	AF3	AF4	AF5	AF6	Measurement value
jjszjsgsl.kaoshi.namespace	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	1.000000000
com.example.healthmonitorlh	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	0.975996958
com.example.pinganpuhui	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	0.754459974
com.damaiwan.dmapp	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	0.680774425
ssxy.kaoshi.namespace	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	0.516015147
com.invariantlabs.winder	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	0.242450074
com.tupo.mi.kaoyan.vocabulary	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	0.000000000
—	—	—	—	—	—	—	—

TABLE 16: Behavior equivalence relationship.

	Nonequivalence	Weak equivalence
Number	14	39
Percent	26.4	73.6

(a)

(b)

FIGURE 5: Measurement value of (a) weak equivalence relationship and (b) 39 APKs.

- (4) The method of combining feature weight and behavior determination is more accurate and reasonable. From Figures 5 and 7 and comparison with Figure 6, it can be

seen that the measurement value is reasonably distributed between 0 and 1, and the measurement value is higher for APK with more features and high weight.

FIGURE 6: The measurement value of 39 APKs by using the method in reference [45].

FIGURE 7: Measurement values of APKs from same company or developer.

9. Conclusion

In this paper, a risk measurement method based on feature weight and behavior determination is proposed. The attack case, benchmark test set, and the actual APK test set are verified by experiments. The main lessons learnt from the proposed method are as follows:

- (1) In this method, AHP is used to calculate the weight of the attack features. It effectively overcomes the defect that the weight of attack features is not considered, which leads to the concentration of measurement results.
- (2) This method uses process algebra modeling and behavior determination. It effectively completes the preprocessing work before measurement so that only need to measure the APK which is weakly equivalent to the attack behavior. This method makes up for the lack of research on preprocessing in the current measurement method.
- (3) The effectiveness of this method is verified by the experimental results of attack cases, benchmark test

sets, and the actual APK test set. Compared with reference [45], the distribution of measurement value of this method is reasonable, which can more clearly indicate the risk degree of APK.

The contribution of this paper can be divided into two aspects: on the one hand, it improves the attack behavior feature set and uses AHP to calculate the feature weight, which solves the problem of incomplete analysis and no weight of the features of collusion privilege escalation attacks; on the other hand, on the premise of using process algebra to decision the behavior of APK, APK which does not need to be measured is filtered out, and a measurement function which fully considers the feature weight is constructed. The method ensures the rationality distribution of measurement result and the consistency of risk measurement value and actual risk.

In the future research, an interesting direction is to apply the theories and methods of multiattribute decision making and fuzzy AHP to our research. In addition, we can continue to refine the features in AF and construct a more perfect measurement function so as to further improve the applicability of the method.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partly financially supported through grants from the National Natural Science Foundation of China (no. 61772450), University Science and Technology Research Project of Hebei Province (no. ZD2018219), Key R&D Projects of Hebei Province (no. 20375001D), Hebei University Humanities and Social Sciences Research Youth Top Talent Project (no. BJ2020064), and Marine Science Research Project of Hebei Normal University of Science and Technology (no. 2018HY013).

References

- [1] T. L. Lu, Y. H. Du, O. Y. Li et al., "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, p. 11, 2020.
- [2] Mobile Security Development White Paper, <https://zt.360.cn/1101061855.php?dtid=1101061451&did=610365044>.
- [3] Nokia Threat Intelligence Report-2019, <https://pages.nokia.com/T003B6-Threat-Intelligence-Report-2019.html>.
- [4] Report on China's Mobile Phone Security in the Third Quarter of 2019, <https://zt.360.cn/1101061855.php?dtid=1101061451&did=210979141>.
- [5] Report on China's Mobile Phone Security in the First Half of 2019, <https://zt.360.cn/1101061855.php?dtid=1101061451&did=210935462>.

- [6] P. Bhat and K. Dutta, "A survey on various threats and current state of security in android platform," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–35, 2019.
- [7] Y. Xu, G. Wang, J. Ren, and Y. Zhang, "An adaptive and configurable protection framework against android privilege escalation threats," *Future Generation Computer Systems*, vol. 92, pp. 210–224, 2019.
- [8] Androguard, <https://github.com/androguard/androguard>.
- [9] Z. Peter, Z. Dmitry, P. Evgeny et al., "Detecting android application malicious behaviors based on the analysis of control flows and data flows," in *Proceedings of the 10th International Conference on Security of Information and Networks (SIN'17)*, pp. 280–283, Jaipur, India, October 2017.
- [10] J. X. Xiao, Z. C. Lu, and Q. H. Xu, "A new android malicious application detection method using feature importance score," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence (CSAI'18)*, pp. 145–150, Shenzhen, China, 2018.
- [11] X. J. Liu, X. F. Dong, and Q. Lei, "Android malware detection based on multi-features," in *Proceedings of the 8th International Conference on Communication and Network Security (ICCNS)*, Qingdao, China, November 2018.
- [12] Z. Peng, X. Luo, F. Zhao, Q. Cheng, and F. Liu, "Hydra-bite: static taint immunity, split, and complot based information capture method for android device," *Wireless Communications and Mobile Computing*, 2018, vol. 2018, pp. 1–19, 2018.
- [13] X. Su, L. J. Xiao, W. J. Li et al., "DroidPortrait: android malware portrait construction based on multidimensional behavior analysis," *Applied Sciences*, vol. 10, no. 11, 2020.
- [14] D. V. Priya and P. Visalakshi, "Detecting android malware using an improved filter based technique in embedded software," *Microprocessors and Microsystems*, vol. 76, 2020.
- [15] P. Abdurrahman and A. Tankut, "Learning to detect android malware via opcode sequences," *Neurocomputing*, vol. 396, pp. 599–608, 2020.
- [16] I. Martín, J. A. Hernández, A. Muñoz et al., "Android malware characterization using metadata and machine learning techniques," *Security and Communication Networks*, vol. 2018, p. 11, 2018.
- [17] O. Olukoya, L. Mackenzie, and I. Omoronyia, "Towards using unstructured user input request for malware detection," *Computers & Security*, vol. 93, 2020.
- [18] W. M. Zhou, Y. Q. Zhang, and X. F. Liu, "POSTER: a new framework against privilege escalation attacks on android," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS'13)*, pp. 1411–1413, New York, NY, USA, November 2013.
- [19] T. Yamauchi, Y. Akao, R. Yoshitani et al., "Additional kernel observer to prevent privilege escalation attacks by focusing on system call privilege changes," in *Proceedings of the 2018 IEEE Conference on Dependable and Secure Computing (DSC)*, Kaohsiung, Taiwan, December 2019.
- [20] C. Wang, R. B. Zhang, and G. Li, "Technology of detection for privilege escalation attack on android," *Transducer and Microsystem Technologies*, vol. 36, pp. 146–148, 2017.
- [21] S. Heuser, M. Negro, P. K. Pendyala et al., "DroidAuditor: forensic analysis of application-layer privilege escalation attacks on android," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 260–268, Sliema, Malta, April 2017.
- [22] S. Bugiel, L. Davi, A. Dmitrienko et al., "Poster: the quest for security against privilege escalation attacks on android," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, pp. 741–744, Chicago, IL, USA, October 2011.
- [23] H.-T. Lee, D. Kim, M. Park, and S.-j. Cho, "Protecting data on android platform against privilege escalation attack," *International Journal of Computer Mathematics*, vol. 93, no. 2, pp. 401–414, 2016.
- [24] W. Z. Yang, C. H. Huang, and Z. Y. Zhang, "A software trustworthiness metric model based on process algebra," *Journal of WUT (Information & Management Engineering)*, vol. 33, no. 6, pp. 928–931, 2011.
- [25] Q. Zhao, H. Q. Wang, G. S. Feng et al., "Measuring method of software dependability based on Pi calculus," *Journal of Jilin University (Engineering and Technology Edition)*, vol. 41, no. 6, 2011.
- [26] J. F. Tian and Y. H. Guo, "Software trustworthiness evaluation model based on a behaviour trajectory matrix," *Information and Software Technology*, vol. 119, 2020.
- [27] X. Yang, G. Jabeen, P. Luo, X.-L. Zhu, and M.-H. Liu, "A unified measurement solution of software trustworthiness based on social-to-software framework," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 603–620, 2018.
- [28] D. X. Wang and Q. Wang, "Trustworthiness evidence supporting evaluation of software process trustworthiness," *RuanJianXueBao/Journal of Software*, vol. 29, no. 11, pp. 3412–3434, 2018.
- [29] Y. Li, J. Wang, D. Zhao, G. Li, and C. Chen, "A two-stage approach for combined heat and power economic emission dispatch: combining multi-objective optimization with integrated decision making," *Energy*, vol. 162, no. 162, pp. 237–254, 2018.
- [30] L. T. Sha, J. M. Fu, J. Chen et al., "A sensitivity measurement for sensitive information processing," *Journal of Computer Research and Development*, vol. 51, no. 5, pp. 1050–1060, 2014.
- [31] J. J. Han, *Risk Evaluation Based on Relationship between Function and Permission for Android App*, Tianjin University, Tianjin, China, 2016.
- [32] M. Deypir, "Entropy-based security risk measurement for android mobile applications," *Soft Computing*, vol. 23, no. 16, pp. 7303–7319, 2019.
- [33] J. F. Xu, J. J. Wang, K. L. Zhu et al., "Credit index measurement method for android application security based on AHP," *Journal of Tsinghua University (Science and Technology)*, vol. 58, no. 2, 2018.
- [34] Z. J. Li, C. M. Wu, and X. Wang, "Assessment of android applications risk behavior based on a Sand box system," *J Tsinghua Univ (Sci&Technol)*, vol. 56, no. 5, 2016.
- [35] X. J. Zhou, Y. Qin, Q. X. Wu et al., "Integrity measurement method for android framework," *Computer Systems Applications*, vol. 28, no. 8, pp. 1–9, 2019.
- [36] Y. Li, Y. H. Li, Z. J. Han et al., "Determination of optimal opening scheme for electromagnetic loop networks based on fuzzy analytic hierarchy process," *Mathematical Problems in Engineering*, vol. 2016, no. 11, 2016.
- [37] S. H. Qing, "Research progress on android security," *Ruan Jian XueBao/Journal of Software*, vol. 27, no. 1, pp. 45–71, 2016.
- [38] Attack Case, <https://pan.baidu.com/s/1haAdhXTDpHvJyHywibOjQ>.
- [39] K. W. Y. Au, Y. F. Zhou, Z. Huang et al., "PScout: Analyzing the Android Permission Specification," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pp. 217–228, Raleigh, NC, USA, October 2012.

- [40] S. Arzt, S. Rasthofer, C. Fritz et al., “FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps,” in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 259–269, New York, NY, USA, June 2014.
- [41] T. L. Saaty and L. G. Vargas, “Estimating technological coefficients by the analytic hierarchy process,” *Socio-Economic Planning Sciences*, vol. 13, no. 6, pp. 333–336, 1979.
- [42] R. Milner, *Communicating and Mobile Systems the Pi-Calculus*, University Press, Cambridge, UK, 1999.
- [43] The Mobility Workbench User’s Guide, <http://www.it.uu.se/profundis/mwb-dist/guide-3.122.pdf>.
- [44] DroidBench-Benchmarks, <https://blogs.uni-paderborn.de/sse/tools/droidbench/>.
- [45] L. Shen, H. Li, H. Wang, Y. Wang, J. Feng, and Y. Jian, “Risk measurement method for privilege escalation attacks on android apps based on process algebra,” *Information*, vol. 11, no. 6, p. 293, 2020.
- [46] Test Set, <https://pan.baidu.com/s/1dYr-WLJjo8Nwlr0p4cR0wQ>.
- [47] L. M. Shen, H. Li, H. Y. Wang et al., “Multifeature-based behavior of privilege escalation attack detection method for android applications,” *Mobile Information Systems*, vol. 2020, p. 16, Article ID 3407437, 2020.