Cluster Comput
DOI 10.1007/s10586-017-1308-3

# A novel method to find important apps base on the analysis of components relationship

**Qi Li**[1] · **Chengze Li**[1] · **Guangyu Gao**[2] · **Yanyi Huang**[1]

Received: 30 July 2017 / Revised: 22 October 2017 / Accepted: 28 October 2017
© Springer Science+Business Media, LLC 2017

**Abstract** With the mobile Internet rapidly developing and the number of mobile applications increasing sharply, the security of the mobile apps has been paid more and more attention in recent years. Many analysis methods for single app have been used in detecting the vulnerability and malicious code. Since mobile apps always related to each other by invoking components, some researchers began to focus on the analysis for multi-applications. But facing with millions of mobile applications, with limited resources, how to improve the ability of security analysis and protection is a difficult problem. For this purpose, we introduce a novel method to mine the correlation among a large number of applications, and finding the nodes that are in the critical position in the process of invoking components. In the proposed method, we first extract the important information from apps and build a database of components. Then, we try to analysis the potential relationship of apps based on the process of invoking components. Moreover, we proposed a novel metric of importance, which can help to find the apps which play important roles in the app-network. We did some experiments to evaluate the proposed method, the experiments show that, we can assess the influence of apps, and figure out the priority of targets during massive application analysis, whether for purpose of detection or protection.

**Keywords** Vulnerability · Mobile application · Allocate efficiently · Metrics of importance

✉ Qi Li
liqi2001@bupt.edu.cn

[1] Beijing University of Posts and Telecommunications, Beijing 100876, China

[2] Beijing Institute of Technology, Beijing 100876, China

## 1 Introduction

Since the year 2011, with the mobile Internet rapidly developing, the number of mobile applications ("apps", for short) has been increasing at a high speed. According to the recent reporting from the global market research firm TrendForce, global smartphone shipments for 2015 grew 10.3% year on year to 1.293 billion units. As the statistics company AppBrain says, the number of Android apps in Google's Play Store is up to nearly 2 million, more than that in the Apple's App Store, over 1.5 million. With the increasing number of apps, the security issues of mobiles have been increasing sharply. As UK SCMagazine reports, 97% of mobile malware targets Android, the platform we mainly focus on [1–3].

With the development of mobile Internet, social networks and mobile payments, the mutual communication between software is more and more frequent [4–6]. For example, as the number of advertisements in mobile Internet increases, more and more applications need to add advertising libraries. With the development of social network and mobile payment, the third-party authorization libraries and payment SDKs are accessed into mobile apps. Therefore, the propagation of calls and data communications is no longer limited in a single app, but constituting a huge complex network through inter-component communications (ICC) of massive mobile applications.

As NowSecure co-founder Hoog [7] said, the real threat to mobile platforms is the leakage and exploit in the mobile applications. According to their statistics, more than 50% of the threat has connection with privacy leakage and more than 48% of the threats is relevant to the vulnerability of applications. However, security testing and defect analysis for a single application cannot meet the needs of mobile internet security protection. In October 2015, a vulnerability called Wormhole was revealed by Wooyun [8] which is located in

Published online: 10 November 2017

Moplus, an SDK released by Baidu Company. The infected applications can install other apps, start particular apps, push phishing pages, edit contacts, send fake SMS, upload local files to remote servers, send specific broadcasts, or do other actions, without user's authorization. Due to the influence of Baidu Developer Platform in China, Moplus is widely called by more than 14,000 apps. Once the infected apps have been installed, malicious apps can obtain privacy data though Moplus with ICC. Under this circumstance, faced with the threats to mobile platforms, it's necessary to detect multiple apps(multi-apps) based on associated behavior analysis.

Some techniques of associated behavior analysis have been proposed for several years, however, these studies are limited to the analysis of small number of association applications. Only few researchers can dig out hidden security problems from such a complex application network [9,10]. Based on the above analysis, we suggest that when analyzing the security and vulnerability of a large number of related applications, we should first analyze the application-network and select the key nodes (mobile apps) in the network. For this reason, we proposed a novel metrics, IMPPR, to find the influential mobile applications based on the analysis of components relationship. By using IMPPR, we can assess the influence of apps, and figure out the priority of targets during massive application analysis, whether for purpose of detection or protection.

The rest of this paper is organized as follow: in Sect. 2, we introduce the related works. In Sect. 3, we describe the proposed method in detail. In Sect. 4, we discuss our method and algorithm through theoretical proof. In Sect. 5, we design some experiments to evaluate the proposed method. We conclude our work in Sect. 6.

## 2 Related works

### 2.1 Inter-component communication

Android applications, developed using Java language, have many differences from Java applications in running mechanism. On the one hand, Android applications run in intelligent terminals, such as mobiles and tablets, where the running environments are extremely complicated, bringing lots of potential security threats [11,12]. This situation increases the difficulty of detecting security problems of mobile apps greatly. On the other hand, Android apps do not contain the main function, but they may have multiple entry points. There are some components that can be accessed to explicitly or implicitly, including applications, activities, services, content providers, receivers, etc. So that traditional analysis technique cannot work effectively on tracking control flows and data flows [13–15].

Octeau et al. [16] presents a method of static analysis on account of inter-component information flows, both in single apps and across apps, based on which they develop a static analysis tool, called EPICC. During the analysis, it first extracts specific context from permission files and executable codes. Then, it calculates the association between components, and getting the control flow supergraphs across components combined with the inner control flows of the apps. Arzt et al. presents a novel and highly precise static taint analysis method for callback functions in Android apps, designs a static analyzing tool called FlowDroid [17]. The authors proposed a detecting model taking the life cycle functions of Android apps into consideration, and the main purpose of this article is to analyze the callback functions. Combining the ideas of Octeau and Arzt, Klieber et al. introduced a taint flow analysis method, which improved the ability of FlowDroid and EPICC [18]. Octeau et al. presents a novel way to solve problems of multi-valued composite constant propagation (MVC), inferring Android inter-component communication (ICC) values, which is required to understanding how the components of the mobile application are called to each other [19]. Unfortunately, Octeau found that some potential intercomponent links are false positives. Afterwards, Octeau et al. overlays a probabilistic model of ICC on top of static analysis results, trained with domain knowledge [20]. They introduce a formalism for inferring ICC links based on set constraints, and design an efficient algorithm for performing link resolution, in order to compute all potential links. This work provides a good research idea for other researchers engaged in related work.

### 2.2 Influence assessment

In this paper, we first analyse the component-invoking relationship between mobile applications, and build a complex network on this analysis. Then, how to evaluate the influence of each node in the complex network is another problem that needs to be solved. There are lots of mature theories and metrics about influence assessment in a complex network, such as metrics aiming at nodes, the shortest path, random suffer, etc [21]. The degree of a node reflects the activation of it, the betweenness centrality mainly reflects whether a node is in the important position of the network, the k-core algorithm focuses on the community influence of a node, and the PageRank is widely used in web importance ranking, in PageRank the importance of a node is felt by the node connected to it [22–25]. However, when we detect the security of apps, the importance of an application depends on whether it is widely concerned, whether it affects many applications, and whether it involves sensitive information and so on. And the methods mentioned above cannot be directly applied to mobile application security detection.

To tackle these problems, this paper proposes a method of influence assessment based on associated components across apps. Firstly, we analyse the data communication between components and the control propagation across apps. After that, we build a complex network which is composed of the components in massive apps. Then, we proposed an influence assessment method, IMPPR, to evaluate the important of an app. Finally, we discuss the validity and rationality of our method through theoretical proof and experiments.

The contribution of our work lies in three points: (1) we research on techniques of ICC extraction, propose a method to construct the association model of ICC for massive apps; (2) we proposed a novel influence assessment metrics, IMPPR, taking apps' categories, popularity, and other factors, into consideration; (3) we developed a tool to judge the importance of each application in security detection process.

## 3 The proposed method

### 3.1 Framework overview

As shown in Fig. 1, there are three processes in the proposed method: clustering analysis, ICC extraction, and association analysis. At the very beginning, we use some tools such as web crawlers to collect information of apps and downloading files into local database, called App Database. In the first process, the clustering analyzer keeps taking apps downloaded recently into consideration, and distinguishing the 3rd part libraries from decompiled intermediate code, including advertising and other functional libraries. In the next process, on the basis of the 3rd part libraries clustered, the association extractor gets appropriate information from apps and models for them. Then record the model of each app into Association Database, from which we can calculate all the association of apps in App Database on component level. These two processes keep working as long as apps are collected into App Database. In the process of association analysis, the association analyzer works out the relationship related to the target app, decides which scope of components need analyzing and how they are related to each other. In the last process, the association analyzer will record all the correlative information into the Association Database.

### 3.2 Feature extraction

#### *3.2.1 ICC information extraction*

Android apps consist of four basic types of components, including activity, service, content provider and broadcast receiver. Components can be accessed to though permission declaration by other apps and components. Activity is the component that interacts with users by screen, displaying the specified control by the API such as setContentView. It can monitor events and handle users' response. There are six life cycle functions, including onCreate, onStart, onResume, onPause, onStop, and onDestroy. The Intent is used to communication and transmit data between activities. The API startActivity and startActivityForResult are used to start authorized activities, according to constructed Intent as shown.

```
Intent intent = new Intent();
intent.setClass(this, "ForwardTarget");
startActivity(intent);
```

```
ComponentName componentName = new ComponentName(" TargetApp",
"TargetApp.Activity")
Intent intent = new Intent();
intent.setComponent(componentName);
startActivity(intent);
```

```
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY,"searchString");
startActivity(intent);
```

And each activity to be invoked must registered in the file AndroidManifest.xml as shown below.

```
<intent-filter>
    <action android:name="chroya.foo"/>
    <category android:name= "android.intent. category.DEFAULT"/>
</intent-filter>
```

Service plays an important role. It is mainly used to execute some time-consuming logic in background. If necessary, it keeps running in background under the condition of program exiting. There are four life cycle functions, including onCreate, onStartCommand, onDestroy, and onBind. The API startService and bindService are used to start authorized services, according to constructed Intent as shown.

```
bindService(new Intent("net.blogjava.mobile.
aidlservice.IMyService"), serviceConnection,Context.
BIND_AUTO_CREATE);
```

```
Intent intent=new Intent(MainActivity.this, ServiceDemo.class);
startService(intent);
```

With the two methods above the target service can be invoked. And the authorized service need to be registered in the AndroidManifest.xml, as shown below.

```
<service android:name=".MyTargetService" >
<intent-filter>
<action android:name="com.aidlservice. IMyTargetService" />
</intent-filter>
</service>
```

ContentProvider implements various data sharing between apps in many ways, such as Shared Preferences, external

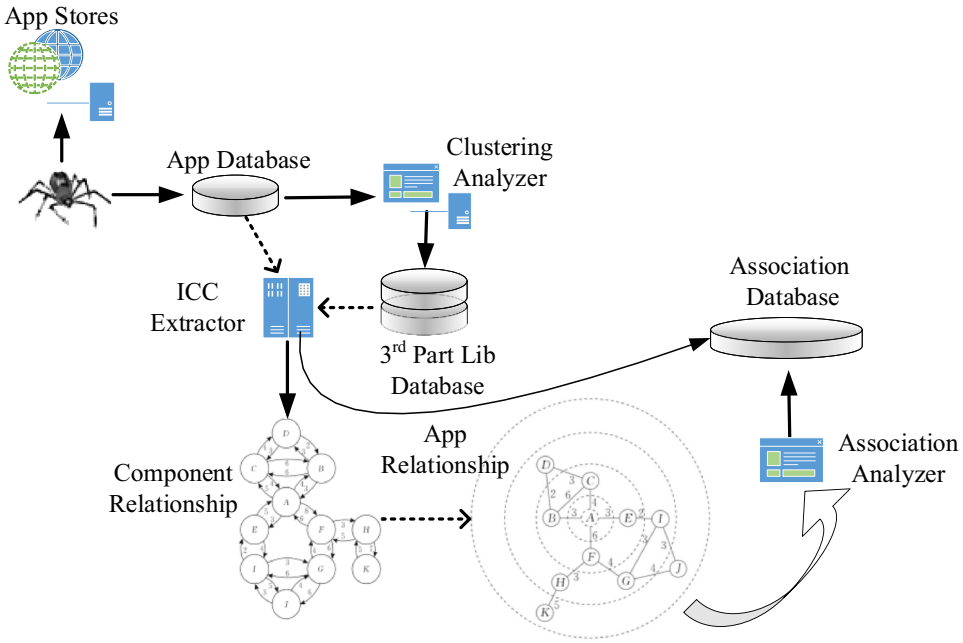


**Fig. 1** Architecture of the proposed method

and internal file storage, SQLite, and so on. The ContentProvider has access to several data sets with a particular uniform resource identifier (URI) as follow, which is supposed to be parsed by class *Uri* and API parse.

content : //com.example.tprovider/trains

The target Content Provider needs registering in the AndroidManifest.xml as well, as shown below.

```
<provider name=".TransmitProvider" authorities="com. example.tprovider" >
```

Broadcast Receiver is the component that focuses on receiving informed information and making corresponding. The broadcasts come from both system and apps, such as system boot accomplishment, low battery, time changing, or other user-defined messages. The API sendBroadcast is used to start authorized receivers, and apps can send broadcasts in the following two ways,

```
Intent intent = new Intent();
intent.setAction("com.android.xiang");
sendBroadcast(intent);
```

```
Intent intent = new Intent("net.blogjava.mobile. MYBROADCAST" );
sendBroadcast(intent);
```

Each receiver must be registered statically in the AndroidManifest.xml, or registered dynamically in source codes.

```
<receiver android:name="MyReceiver" >
    <intent-filter>
        <action android:name="net.blogjava.mobile.
         MYBROADCAST" />
    </intent-filter>
</receiver>
```

```
IntentFilter filter = new IntentFilter();
filter.addAction("com.android.shang");
registerReceiver(myReceiver, filter);
```

### 3.2.2 ICC association extraction

**Presentation of declaration.** The component must by declared in the AndroidManifest.xml as long as it is required to be accessed. According to presentations shown in Sect. 3.1, we construct the seven-dimension vector of declaration with the extracted features, as < targetComponent, targetType, targetPkg, targetClass, scheme, host, uri >. The declaration set is composed of these vectors.

**Presentation of calls**. We extract the calls related features and construct the vectors, as < sourceComponent, desType, desPkg, desClass, scheme, host, uri >. The call set is composed of these vectors.

**Presentation of inter-component association.** It is required to get the declaration set beforehand. For this purpose, during constructing the call set though scanning decompiled intermediate code, if the tuple < desType, desPkg, desClass > of element in the call set is identical to the tuple of that in the

**Table 1** Relationship for component association

| targetType | sourcePkg | sourceClass | desPkg | desClass | scheme | authority | uri |
|---|---|---|---|---|---|---|---|
| Activity | ◎ | ◎ | □ | ◎ | | | |
| | ◎ | ◎ | | | ◎ | □ | □ |
| Service | ◎ | ◎ | □ | ◎ | | | |
| Reciever | ◎ | ◎ | □ | ◎ | | | |
| Provider | ◎ | ◎ | | | | | ◎ |

◎Means the essential condition
□Means the optional condition

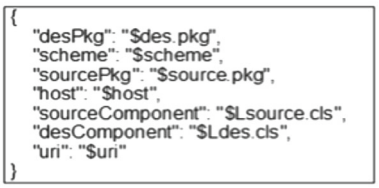


**Fig. 2** Presentation of action strings

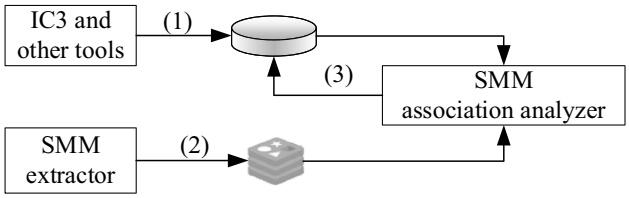


**Fig. 3** The main working process of SMM

declaration set, or other satisfied conditions, we figure out a record of association between components. We sum up all conditions, obtain the Table 1, and propose the presentation of ActionStrings with essential fields is shown in Fig. 2.

There are some permissions and components declared in the AndroidManifest.xml, and lots of information that can be extracted from the classes.dex file. Some of the information will be used in this phase, listed below:

- Libraries called by an application;
- Components which are declared in an application;
- Calls and Intent related to ICC;
- Permissions or user permissions which are declared in AndroidManifest.xml;
- Specific data.

In this paper, we use IC3 to extract some of the information mentioned above. Since the IC3 has some insufficient in the aspect of calculating the source of ICC, the IC3 can find out almost all the components of the apps, but it cannot find which components calls for some target components. In addition, the IC3 doesn't know which components are the target, and it doesn't know which scope of components the source belongs to as well. Aiming at the issues, we design the source mining module (SMM) for ICC. The SMM is used to calculate all ActionStrings of the apps, by which the declared components can be invoked.

The process of SMM is shown in Fig. 3. In step (1), we use the IC3 and other tools to extracted some information to generate the declaration set, and write these records into database. In step (2), we use our SMM extractor to get information of vital importance, mainly focusing on calculating the call set. Then in step (3), the SMM is used to calculate the association of components.

We use the SMM for data parsing and use the PostgreSQL database for information storage, building the entity association diagram based on the ICC, as shown in Fig. 4.

During the process, once an app is processed by the SMM, all action strings will be record into table ActionStrings of database. The list of items is one of the input parameters, from which we know whether the target of ICC is some component or not.

As a result, we figure out a large complex association network of apps and components.

### 3.3 Importance-evaluation method

PageRank is present by Page et al. in 1999 [26], the essence of the PageRank algorithm is that a problem of importance assessment is transformed into a public participation, which is solved by group democracy. The link between nodes is considered to be voting behavior. As shown in Eq. (1), suppose that the set $P$ is a collection of all nodes that are connected to the node $q$, the rank value of $q$ is defined as follows:

$$PR(q) = \sum_{p \in P} \frac{PR(p)}{|p|} \tag{1}$$

$PR(q)$ is the rank of node $q$, $|p|$ is the number of out-links of node $p$. Moreover, in PageRank algorithm, the coefficient $\alpha$ is added into random suffering to avoid instability caused by isolated nodes, as shown in Eq. (2),

$$PR(q) = (1-\alpha) \cdot \frac{1}{n} + \alpha \sum_{p \in P} \frac{PR(p)}{|p|} \tag{2}$$

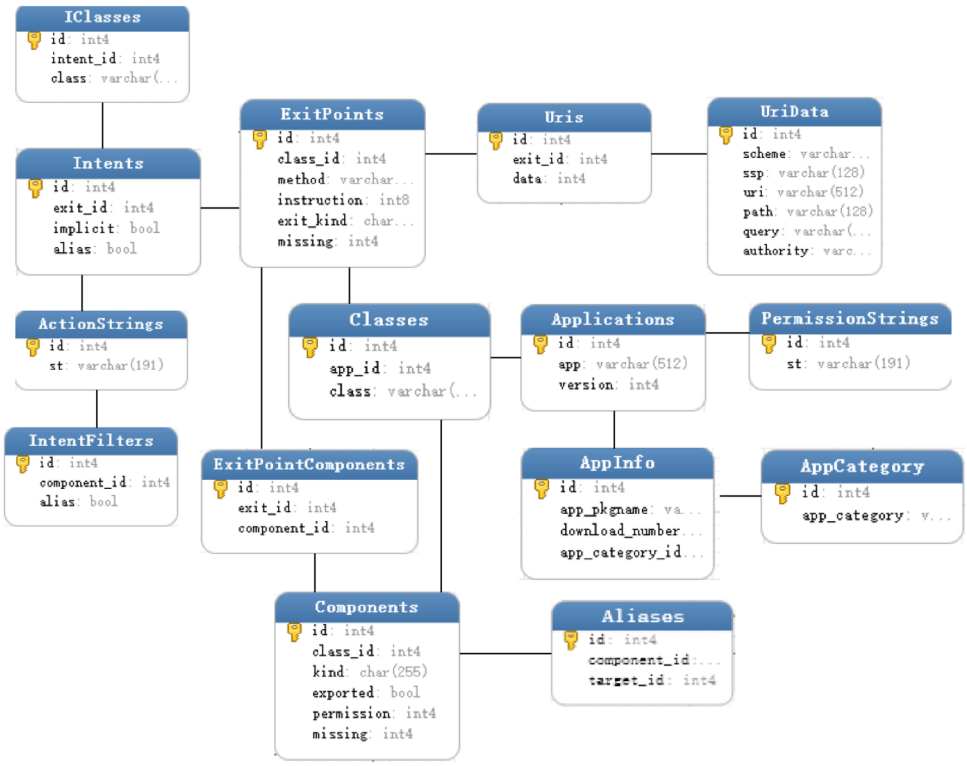


**Fig. 4** ER diagram associated with component

where $\alpha$ represents the probability of continuing suffering by the transition matrix after reaching a node, $n$ is the number of nodes in the complex network.

IMPPR is our influence assessment method present in this paper. Since PageRank does not take the theme of nodes into account, the rank value of nodes is evenly distributed to the following nodes in the iterative computation, resulting in unreasonable results in some certain. We put forward to IMPPR, calculating the extended activation forces of nodes, according to which the rank of nodes is distributed to the following nodes. The algorithm is shown below,

$$imppr\,(x) = (1-\alpha)\cdot\frac{1}{n} + \alpha\cdot\sum_{p\in I(x)} imppr\,(p)\cdot\frac{F\,(p,x)}{\sum_{q\in O(p)} F\,(p,q)} \tag{3}$$

In equation (3), *imppr(p)* is the influence assessment metrics of node $p$. We donate $I(x)$ as in-link set node $x$, donate $O(x)$ as the out-link set of node $x$, and donate $F(p,q)$ as the extended activation force from node $p$ to node $q$.

In our IMPMetrics, we define the importance of components according to the following principles:

- The important components mean that they have not only lots of in-links but also lots of out-links;
- The more in-links or out-links the component have, the higher value of metrics it gains;
- The categories and download numbers of apps affect the importance of the components;
- With the increasing of components' links, the unconnected graphs can be transformed into connected graphs, and the node will gain a nonzero value.

In Ref. [25], Guo et al. proposed activation force to weight the links of a complex network. In this paper, we improve the extended activation force taking the categories, popularity, and other factors of apps into consideration. Donate the extended activation force from node $i$ to $j$ as $f_{ij}$, and $f_{ij}$ is formulated as follows:

$$f_{ij} = e^{c_{ij}}\cdot e^{a_{ij}}\cdot e^{o_{ij}}$$

in which, $c_{ij}$ is the category weight coefficient, $a_{ij}$ is the metrics of activation force, and $o_{ij}$ is the other external weight factor. In this paper, $c_{ij}$ is determined artificially according to automated statistics by the WebCrawler.

The value of IMPMetrics reflects the situation of stability between in-links and out-links. The node with high value by IMPMetrics must have both a large scale of in-links and a comparable scale of out-links, which is supposed to be regarded as the vital node. For example, usually the SDKs don't allow apps access to the actual logic, supplying user interfaces for apps. And these interfaces don't access to the

actual logic directly, other than access through the level of middleware. With the proposed IMPMetrics, the level of middleware can gain a higher value of metrics, supposed to be regarded as the vital, especially in the area of vulnerability testing and components reinforcement.

## 4 Discussion about the convergence of IMPPR

In terms of Eq. (3), the algorithm IMPPR is an iterative process, and the results of each round of iteration are only related to the current state, based on the state transition matrix of nodes extended activation forces, which means that the whole working process is a Markov process. Assuming the state transition matrix without random suffering coefficient as M, and assuming that with random suffering coefficient as M′, then we have

$$M' = \alpha \cdot M + (1-\alpha) \cdot \frac{1}{n} \cdot I$$
$$impppr^{i+1} = M' \cdot impppr^{i}$$

The state transition matrix of nodes extended activation forces, denoted as $\boldsymbol{p}$, is stochastic due to the normalization during each round of iteration of IMPPR. The probability from one node to any others is positive, so $\boldsymbol{p}$ is irreducible. Apparently, $\boldsymbol{p}$ is aperiodic. So according to the theory of Markov chains, as $\boldsymbol{p}$ is stochastic, irreducible and aperiodic, then for any start vector, the power method applied to $\boldsymbol{p}$ converges to a unique positive stationary vector. In terms of Perron-Frobenius theorem, an irreducible and aperiodic Markov chain is guaranteed to coverage to a unique stationary distribution. So, the algorithm IMPPR converges to a stationary probability.

## 5 Experiments

In our experiment, we select app samples from released images, GooglePlay Store, the official app store of Android, and over 100 unofficial app stores in China. The WebCrawler module fetches information of apps from app stores and records them into our database, such as developers, description, download number, comments from users, etc. We extract apps from released images, and record developer and categories artificially. We have already handled more than 91 thousand apps by the time of this paper statistics, and extract over 1.68 million components from them. We construct a huge complex association network of components, and assess the influence of components and apps by IMPPR algorithm, Betweenness algorithm and PageRank algorithms.

In Fig. 5, we use the radius and color to represent the metrics values. Important nodes have larger radius and darker color than inconsequential nodes. By the proposed IMPMetrics, the nodes with high importance are mainly gathered in the area $C$ shown in Fig. 5a. The nodes with the highest metrics value of importance by betweenness centrality are located in the area $D$ in Fig. 5a, and those metrics by page-ranks, are located in the area $E$ in Fig. 5b.

After filtering and enlarging the nodes in these areas, we get Fig. 6. In Fig. 6, these diagrams are the enlarged from the corresponding position in Fig. 5. In the area $D1$, $D2$ and $D3$, there are some components gaining very high values of metrics by betweenness centrality, shown in Fig. 6c–e. But, from Fig. 5, we can see that the components in these areas are from the same app. In this situation, components in these areas gaining high values of metrics seems unreasonable.

In the area $E$, the component $e$ gaining the highest values of metrics by PageRank, shown in Fig. 6g. We can see that there are only lots of in-links from other components. After querying this record, this component is the service called *MediaScannerService*, an important component in the app*com.android.providers.media*, the system media player, called by lots of apps. According to our principles, for the security purpose, this component should not be regarding as the important one. In fact, our aim is to find important components for vulnerability analysis and reinforce, in which case the system components shouldn't be the targets. Usually, system components provide services for other apps with interfaces and they won't call components in other apps.

Figure 7a shows the metrics values of components. During the extraction process, we assign adjacent identifications(ID) to components of one app, and represent the app with the minimum ID of its components. Fig. 7b shows the metrics values of apps, the summation of metrics values of components. In Fig. 7a, b, there are some apps gaining large values with PageRank algorithm, while gaining small values with IMPPR, such as b, c, etc. Though, there're certain apps gaining approximate values with two algorithms, such as a, etc.

Figure 8 demonstrates the association of app $c$ and related apps, in which we represent app $c$ with the central area enclosed by a dashed circle. The surrounding nodes are the components of apps linked to c. There are lots of in-links from other apps to c, however there're no out-links from c to others. This means that even if $c$ is a malicious application, it will only affect its own operation, and cannot affect other applications. Therefore, we will pay more attention to applications that have important impact on other applications for the purpose of security protection. For that reason, in the proposed method it gains lower importance than the other two methods.

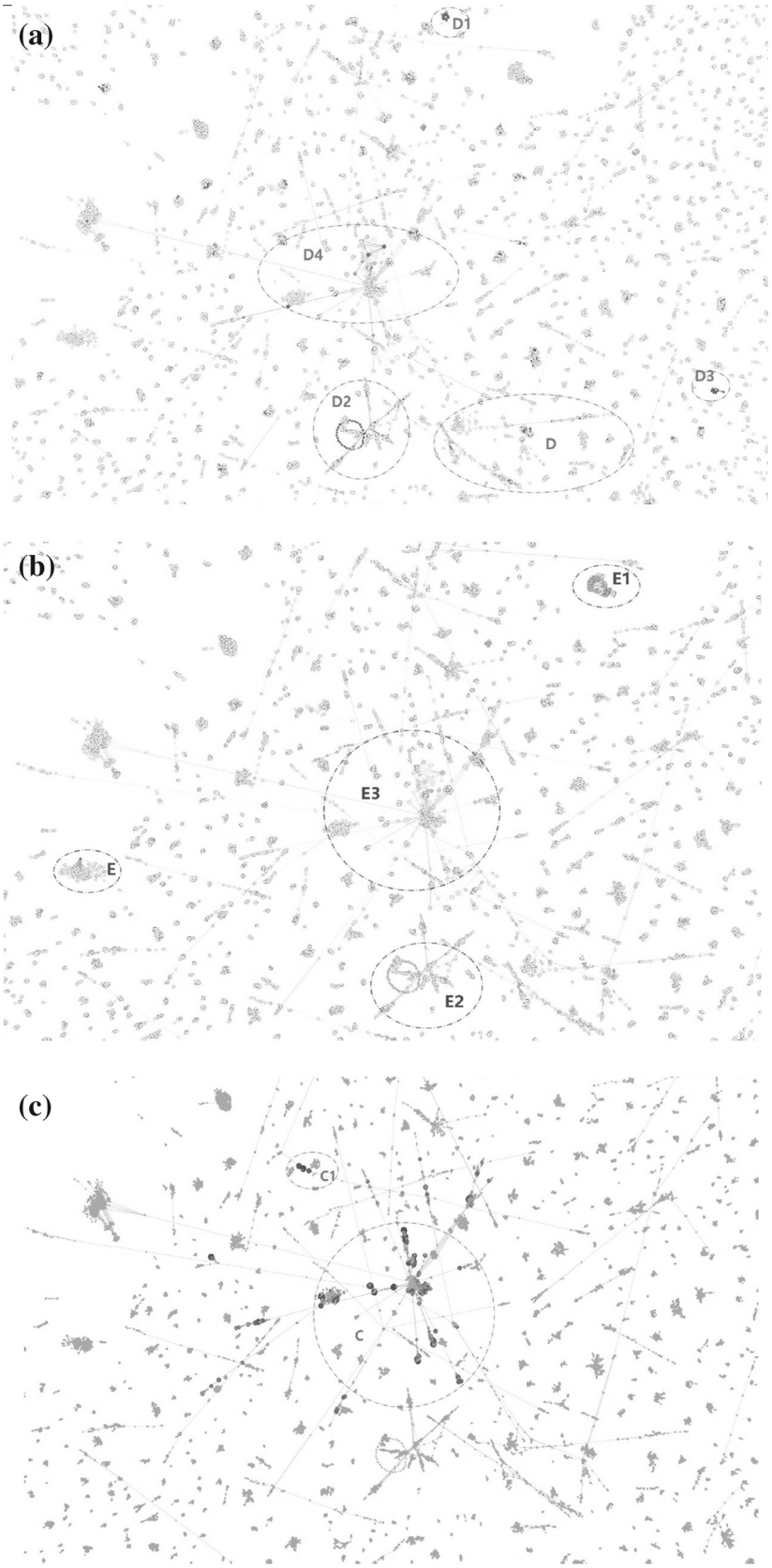


**Fig. 5** Comparison of the component relationship metrics. **a** The component relationship metrics by Betweenness centrality. **b** The component relationship metrics by Pagerank. **c** The component relationship metrics by IMPPR

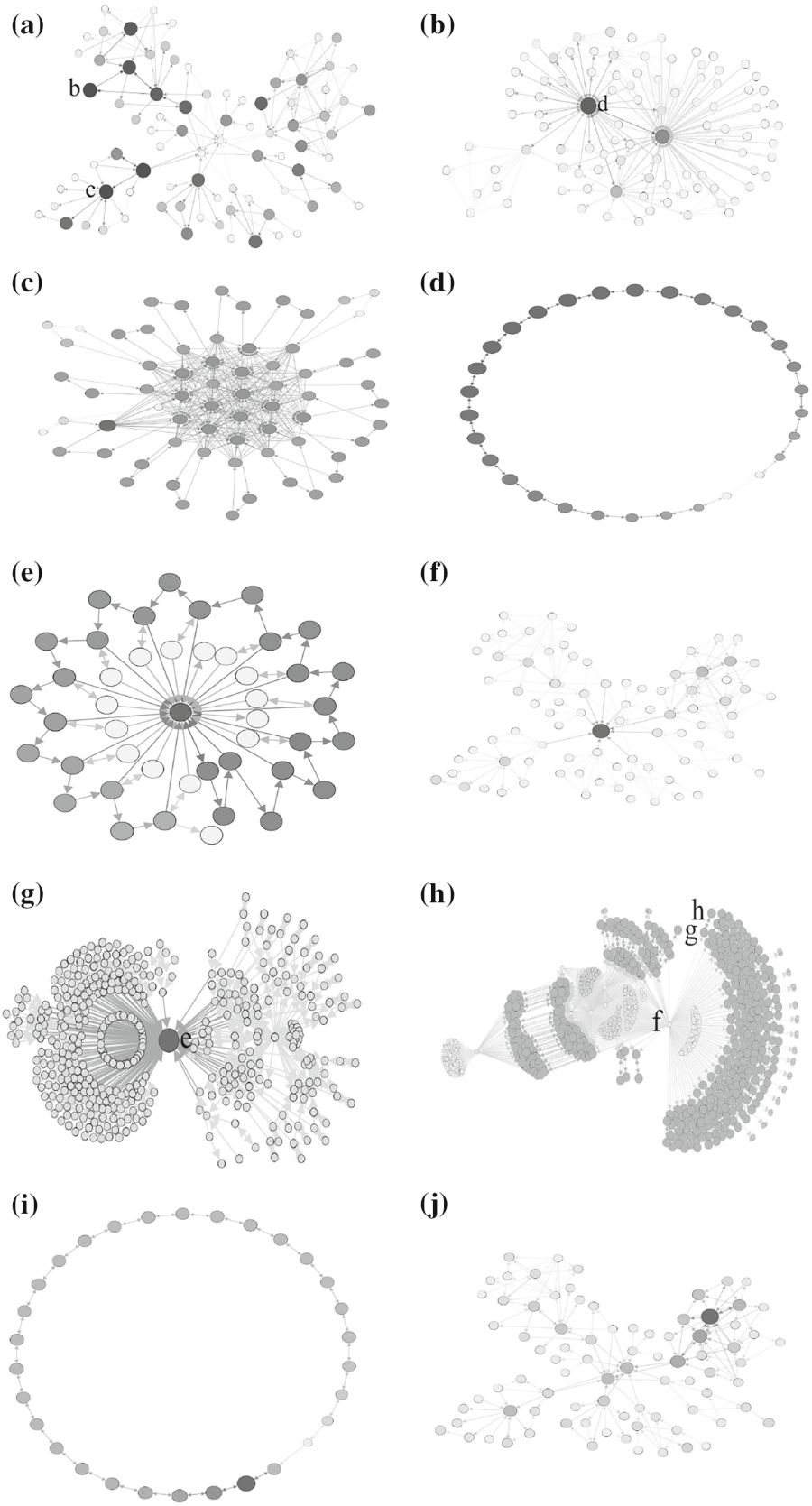


**Fig. 6** The enlarge of different area. **a** enlarge for area $C$. **b** enlarge for area $D$. **c** enlarge for area $D1$. **d** enlarge for area $D2$. **e** enlarge for area $D3$. **f** enlarge for area $D4$. **g** enlarge for area $E$. **h** enlarge for area $E1$. **i** enlarge for area $E2$. **j** enlarge for area $E3$

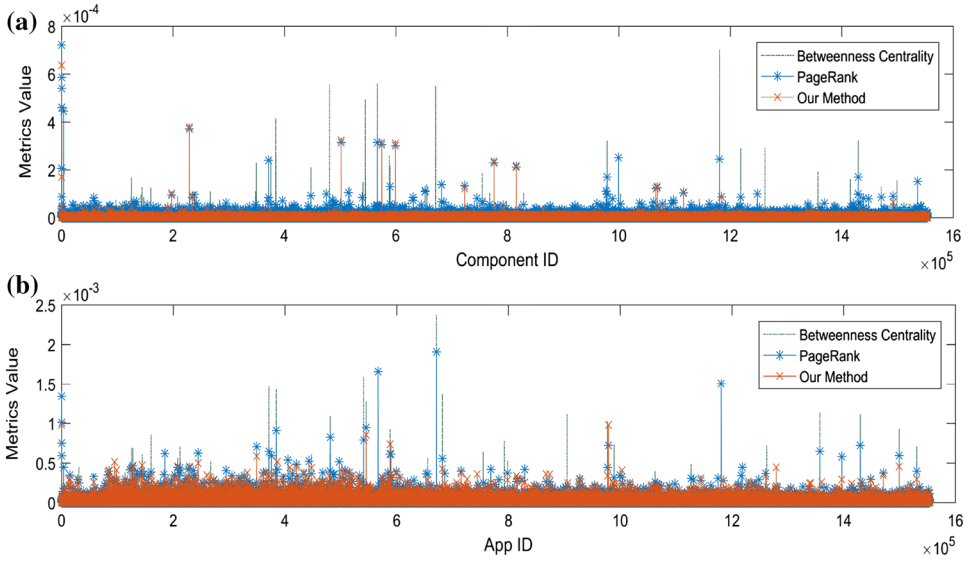


**Fig. 7** Influence assessment by IMPPR and PageRank

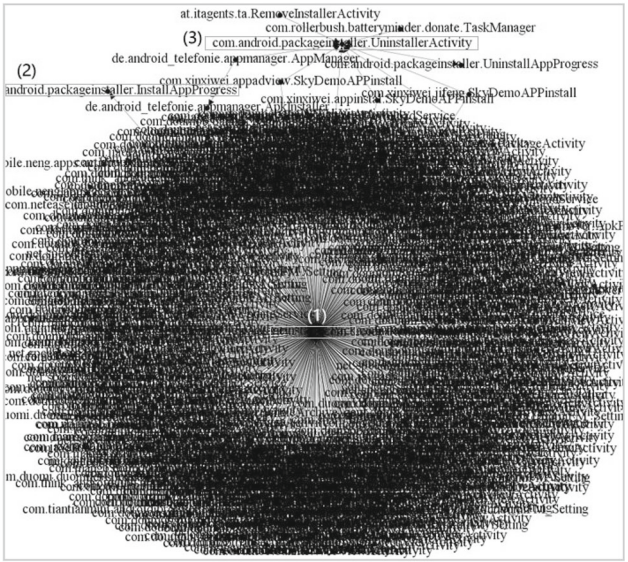


**Fig. 8** Association related to typical app c

## 6 Conclusion

In this paper, we present a method of influence assessment based on associated components in apps, and present a tool of influence assessment for massive Android applications. On one hand, we extract essential information from apps, analyze the control propagation and data communication between components across apps, and construct a huge complex association network of ICC for massive apps on component level. From our research, we can calculate potential connections with the target app, and find out all apps associated with the target. On the other hand, we put forward our influence assessment metrics, IMPPR, an improved algorithm based on PageRank, taking extended activation forces of apps into account, considering categories, popularity, and other factors. Using IMPPR, we can assess the influence of apps, and figure out the priority of targets during massive application analysis, whether for purpose of detection or protection.

Analyzing more than 91 thousand applications from Google's Play Store and other unofficial stores, we construct a huge complex association network with nearly 1.68 million components. According to our experiments, social apps and payment apps usually gain high influence because of their attributes for information propagation. Through analysis of experimental data, we can see that our IMPPR is more reasonable than PageRank algorithm in assessing influence of apps.

In this paper, we assess influence of apps from the aspect of intrinsic characteristic of apps, from which we find that social apps and payment apps play more important roles. In the future, we plan to assess influence from the aspect of users' data flow and business flow, which is worthy of our in-depth study.

**Acknowledgements** This work is supported by National Natural Science Foundation of China (CN) Project (U153610079, 61401038).

## References

1. AppBrain: Android statistics: number of android applications (2016)
2. Baidu: Bdsuite android market. https://www.baidu.com/ (2017)
3. Tencent: Myapp market. https://android.myapp.com/ (2017)
4. Malhotra, R.: an empirical framework for defect prediction using machine learning techniques with Android software. Appl. Soft Comput. **40**(10), 993–1006 (2016)
5. Li, L., Bartel, A., Bissyand'e, T. F., Klein, J., Le Traon, Y.: ApkCombiner: combining multiple android apps to support inter-app analysis. In: Proceedings of the 30th IFIP International

Conference on ICT Systems Security and Privacy Protection (SEC 2015) (2015)
6. Lu, L., Li, Z., Wu, Z., Lee, W., Jiang, G.: Chex: statically vetting android apps for component hijacking vulnerabilities. In: Proceedings of the 2012 ACM conference on Computer and communications security. ACM, pp. 229–240 (2012)
7. Hoog, A.: The incident response playbook for android and ios. In: RSA Conference 2016 (2016)
8. Wooyun.: Wormhole analysis report. Technical Report (2015)
9. Sbirlea, D., Burke, M.G., Guarnieri, S., Pistoia, M., Sarkar, V.: Automatic detection of inter-application permission leaks in android applications. IBM J. Res. Dev. **57**(6), 10-1 (2013)
10. Du, Y., Wang, X., Wang, J.: A static android malicious code detection method based on multi-source fusion. Secur. Commun. Netw. **8**(17), 3238–3246 (2015)
11. Zhao, Z., Wang, J., Wang, C.: An unknown malware detection scheme based on the features of graph. Secur. Commun. Netw. **6**(2), 239–246 (2013)
12. Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.-R., Shastry, B.: Towards taming privilege-escalation attacks on android. In: NDSS, vol. 17, p. 19 (2012)
13. Li, L.: Boosting static security analysis of android apps through code instrumentation. Ph.D. dissertation, University of Luxembourg, Luxembourg (2016)
14. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PloS ONE **9**(6), e98679 (2014)
15. Marforio, C., Francillon, A., Capkun, S., Capkun, S., Capkun, S.: Application collusion attack on the permission-based security model and its implications for modern smartphone systems. Department of Computer Science, ETH Zurich, Zurich (2011)
16. Octeau, D., McDaniel, P., Jha, S., Bartel, A., Bodden, E., Klein, J., Yves, L.: Effective inter-component communication mapping in android with EPICC: an essential step towards holistic security analysis. In: USENIX Security 2013 (2013)
17. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Yves, L., Octeau, D., McDaniel, P.: Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. vol. 49, no. 6, pp. 259–269 (2014)
18. Klieber, W., Flynn, L., Bhosale, A., Jia, L., Bauer, L.: Android taint flow analysis for app sets, pp. 1–6 (2014)
19. Octeau, D., Luchaup, D., Dering, M., Jha, S., McDaniel, P.: "Composite constant propagation: application to android inter-component communication analysis. In: Proceedings of the 37th International Conference on Software Engineering, IEEE Press, vol. 1, pp. 77–88 (2015)
20. Octeau, D., Jha, S., Dering, M., McDaniel, P., Bartel, A., Li, L., Klein, J., Yves, L.: Combining static analysis with probabilistic models to enable market-scale android inter-component analysis. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM, pp. 469–484 (2016)
21. Zhao, Y., Song, W.: Survey on social-aware data dissemination over mobile wireless networks. IEEE Access **5**, 6049–6059 (2017)
22. Zhou, J., Wang, Q., Tsai, S., Xue, Y., Dong, W.: How to evaluate the job satisfaction of development personnel. IEEE Trans. Syst. Man Cybern. **47**(11), 2809–2816 (2017)
23. Baldinelli, G., Bonafoni, S., Rotili, A.: Albedo retrieval from multispectral Landsat 8 observation in Urban environment: algorithm validation by in situ measurements. IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens. **10**(10), 4504–4511 (2017)
24. Bai, X., Lee, I., Ning, Z., Tolba, A., Xia, F.: The role of positive and negative citations in scientific evaluation. IEEE Access **5**, 17607–17617 (2017)
25. Guo, J., Guo, H.L., Wang, Z.Y.: An activation force based affinity measure for analyzing complex networks. Sci. Rep. **1**, 113 (2011)
26. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Tech. Rep. (1999)

**Qi Li** received the Ph.D. degree in computer science and technology from Beijing University of Posts and Telecommunications, China, in 2010. She is currently an associate professor in the Information Security Center, State Key Laboratory of Networking and Switching Technology, School of Computer Science, Beijing University of Posts and Telecommunications, China. Her current research focuses on information systems and software security.

**Chengze Li** received the B.E. degree in information security from Northeastern University, Shenyang, China, in 2011. He is currently pursuing the Ph.D. degree in information security at Beijing University of Posts and Telecommunication. His research interest lies at the intersection of program analysis, privacy and security, and mobile systems.

**Guangyu Gao** is is currently an associate professor in the Information Security Center, State Key Laboratory of Networking and Switching Technology, School of Computer Science

**Yanyi Huang** is a postgraduate of Beijing University of Posts and Telecommunications (BUPT) working on mobile malware analysis. She has been involved in mobile application security research since 2015.