DETECTING MALWARE IN ANDROID BASED DEVICES

Saka Sowmya[1], Sri.S.K.A.Manoj[2] , Dr. Lalitha Bhaskari[3]
P.G. Scholar, Department of CS & SE, College of Engineering (A), Andhra University, Visakhapatnam, India[1]
Research Scholar, Department of CS & SE, College of Engineering (A), Andhra University, Visakhapatnam, India[2]
Professor, Department of CS & SE, College of Engineering (A), Andhra University, Visakhapatnam, India[3]

ABSTRACT: Smartphones are growing rapidly in terms of device functionality and providing increased amounts of knowledge with more advanced computing capabilities and connectivity than ordinary mobile phones. Incorporation of lot of features has prompted the increased usage of smartphones. Application development is one of those features where users can download wide range of applications to perform different tasks. The open nature of markets are resulting in several privacy and security concerns as malware developers are attempting to restructure the permissions of applications and upload those compromised applications in various sites. These malwares are trying to invade the personal information of the user, read communications without permissions and exploiting data with malicious intent. Moreover, many kinds of android applications require too many permissions than which they need to provide user's services. Testing and in-depth analysis is required on the features of the application permissions. In this paper, a background application service is proposed that can detect whether an application is a malware or not by performing analysis of application permissions on installation, identifying malicious characteristics and alerting the user.

Keywords— Smartphones, Malwares, Permissions, Privacy Intrusion, Feature Extraction

I. INTRODUCTION

The term smartphone was first used in the mid of 1995 and the industry is continuously witnessing new entrants over the years which are trying to outcompete the incumbents for design strategies. Many advancements are taking place in enhancing the technical features and specifications to end-users. These changes in structuring the smartphones is resulting in increased number of users over the years and estimated to be around 2.5 billion in 2019. For every mobile device to run their services, an operating system is needed. The features and capabilities supported by the initial operating systems were limited. However the improved versions of mobile devices are demanding high speed central processing units with large storage space, high resolution screens including the operating system features of personal computer.

Application programs are placed on top of the operating system architecture which are responsible for various functions to perform. Its design has experienced a three-phase evolution in all these years: initially it was PC-based operating system, then embedded operating system and now came the mobile operating system. Throughout the process, the architecture has gone from complex to simple. The main changes are being performed in reducing the size of the hardware components, improving the software for better user interface and responsiveness and restructuring operating systems to be open in nature. The advancements are resulting in different competing mobile operating systems like Android, iOS, Symbian, BlackBerry OS etc., Starting from version 1.0, Android has been transforming in various dimensions. Many versions were released one after another, improvised than the previous versions, thus increasing the usage of these devices. Many applications are being developed and are made available in various sites providing different functionalities to end users. The developers are putting their efforts to produce applications on this platform but these applications are being altered with a malicious intent and inserted back into those sites as trusted applications. Consequently, an urgent need is arising to develop powerful solutions for application security. Solutions which are high in accuracy are not available in the market at present.

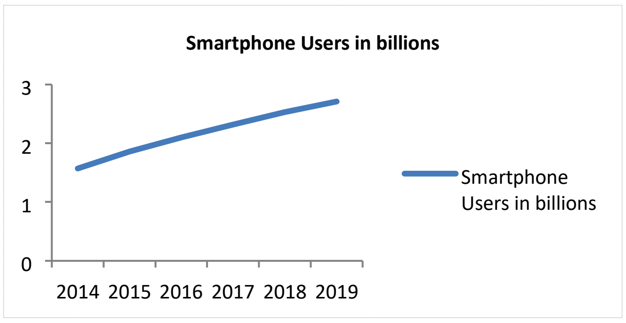


Figure 1: Smartphone Users

II. RELATED WORK

The application development and ease of uploading the applications is on increasing trend. The availability of applications on open source platforms is becoming an advantage for different malware developers as they are trying to find new ways to cross the security checks and change the functionality of the application code with their malicious code, thus leading to growth in malware activities. These altered applications are again uploaded to the websites. The operating system of an smartphone is designed in such a way that every application has to request the users who are downloading applications for a list of rights called permissions. The main aim of the permission model defined by Android is to ensure the security of the user's personal data. Android defined different levels of protection. They are *normal*, *signature* and *dangerous* permissions and depending on the permissions users grant the application, the privacy of the user is affected. *Normal Permissions* are used to access resources which are not present in the sandbox of the application and are granted automatically and cannot be revoked by the user. *Signature Permissions* are requests that has same signed certificate as the application that attempts to use the permission and are granted at install time. *Dangerous Permissions* are used to access the information of the users and may pose threat to the security of the user and operational activities of other applications in the device. Another set of permissions called *Special Permissions* are present which are sensitive in nature and the applications should not use them.

Permissions are grouped and formed as a group keeping in view the features and capabilities of the device. A permission group consists of different permissions that fall under one category and are defined as a group in the manifest file. The use of permissions groups is that it avoids the number of permission requests to the user and the developer is given access to ask only the permissions needed. The drawback is that, if an application asks for a particular permission present in a group, the application framework asks for the higher level group, so that if the application needs any permission present in that group other than the requested permission, Android automatically grants the permission without asking the user. Hence close monitoring of group permissions is also necessary.

Permissions work in the following way, when a user attempts to download an application, it asks for the device and data permissions before user can install it. For instance, if a photography application is downloaded, user should give access to camera, for satellite navigation applications access should be given to GPS signal etc. But the malicious activities are taking place by misusing the permissions like an alarm clock application asking permission to access photos, camera and microphone, free gaming application asking access to texts and contact list etc., This is a signal that the application the user wish to download may access personal data that is not needed for its functionality and also transfer the data to malware developers while the app is running. Many methods are being proposed to detect Android malware based on permissions. These methods are analysing the features of the application by applying various learning based techniques but they failed to catch several types of dynamically generated malwares. Research is in progress to develop techniques which are high in accuracy in detection of malware in applications.

III. PROPOSED WORK AND METHODOLOGY

Permissions misuse is due to the increased number of permissions required for performing a particular application functionality. The permissions which are under the category of dangerous permissions are not the only dangerous set of permissions but the other group of permissions are also posing a serious threat to the users. Permission groups are also to be analysed because when a user grants a permission which comes under a group, they are not aware of the complete list of permissions present in the group. As the flags of the permissions present in the group are unknown to the users, they may unknowingly grant the access to a malware author. Another aspect that requires analysis is API call signatures, commands signature permissions etc., Analysing these permissions is very crucial in determining whether an application is having any untrusted permission which may be threat to the confidential data present in the device. These issues are addressed in this work by following analysis techniques on the permissions.

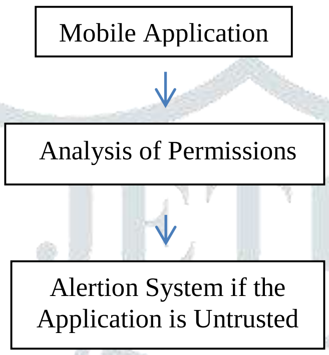


FIGURE 2: PROPOSED SYSTEM ARCHITECTURE

The implementation of the above proposed method is carried out in three stages.

Extraction: The entire information about the application are present in the application packages. Once the application is installed, the extraction is performed.

- getPackageManager is a function present in PackageManager class, to know the available packages in the device.
- Using InstalledPackages function present in the getPackageManager function, details of the installed applications can be extracted.
- After extracting the packages of all the installed applications, each package is studied using PackageInfo class which contains the metadata of the application.

Analysis: Every application declares its permissions in the application files. The manifest file of the application contains all the defined permissions i.e permissions which are required by the application and which will be used in the future for updation purpose. The installation application key (.apk key) is uncompressed and then the two important files which contain the metadata of the application are parsed. The Manifest.xml file present in the metadata is converted into readable format and permissions which are needed by the application are extracted. From the classes.dex file the information about the API calls which are sensitive in nature like chmod, that might be used for changing users permissions on files, chown, an API that might be used to change the group of files and content resolver delete, that might be used for deleting users messages or contacts etc., are examined.

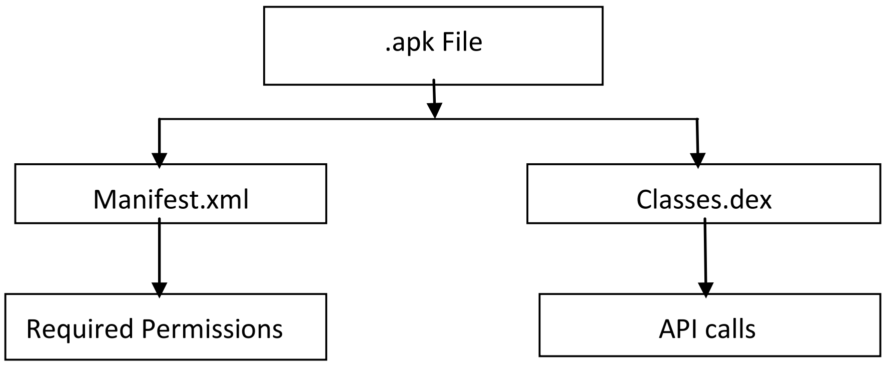


Figure 3: Files Analyzed in Analysis

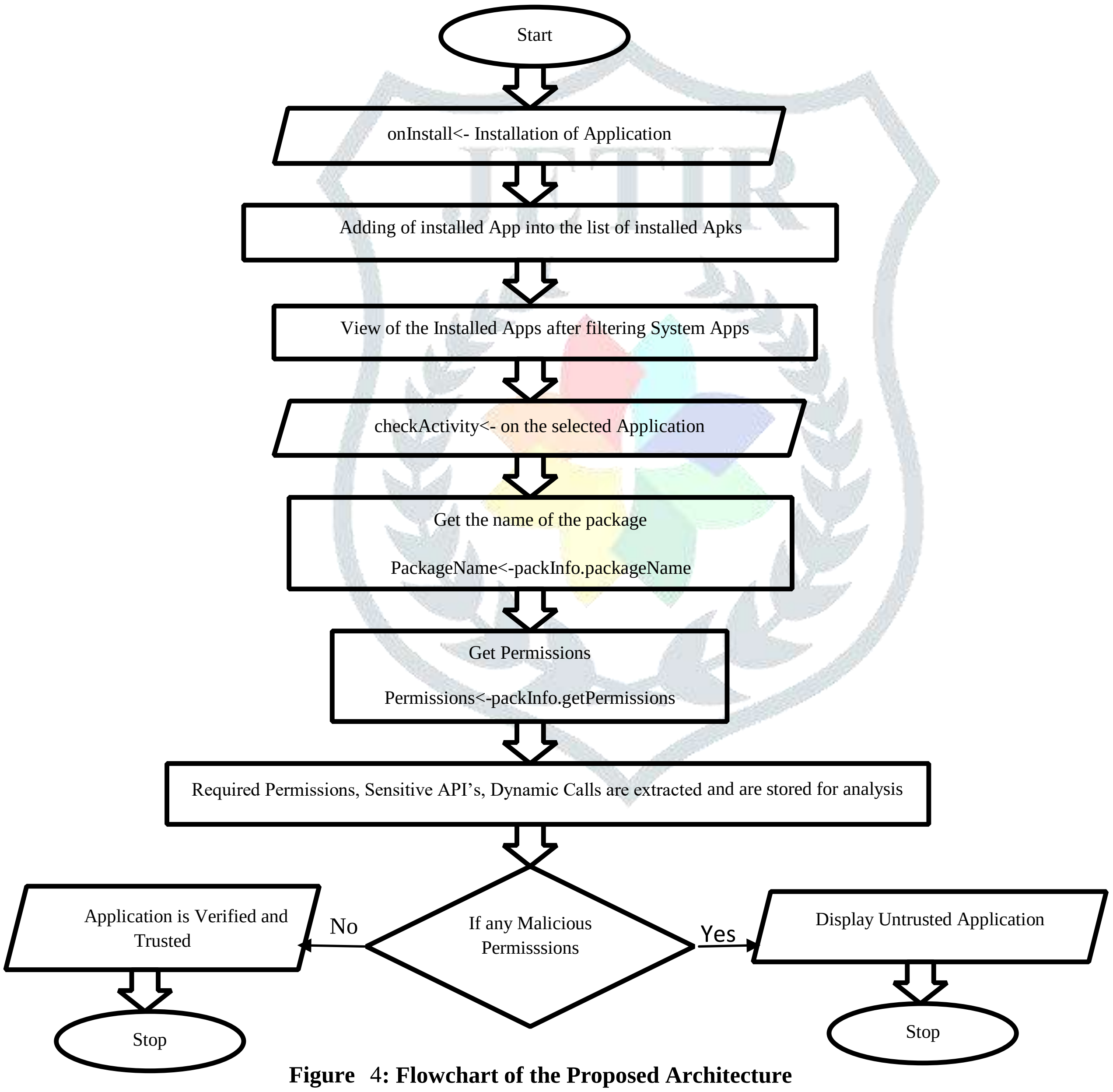

Start
onInstall<- Installation of Application
Adding of installed App into the list of installed Apks
View of the Installed Apps after filtering System Apps
checkActivity<- on the selected Application
Get the name of the package
PackageName<-packInfo.packageName
Get Permissions
Permissions<-packInfo.getPermissions
Required Permissions, Sensitive API's, Dynamic Calls are extracted and are stored for analysis
If any Malicious Permisssions
No
Yes
Application is Verified and Trusted
Display Untrusted Application
Stop
Stop

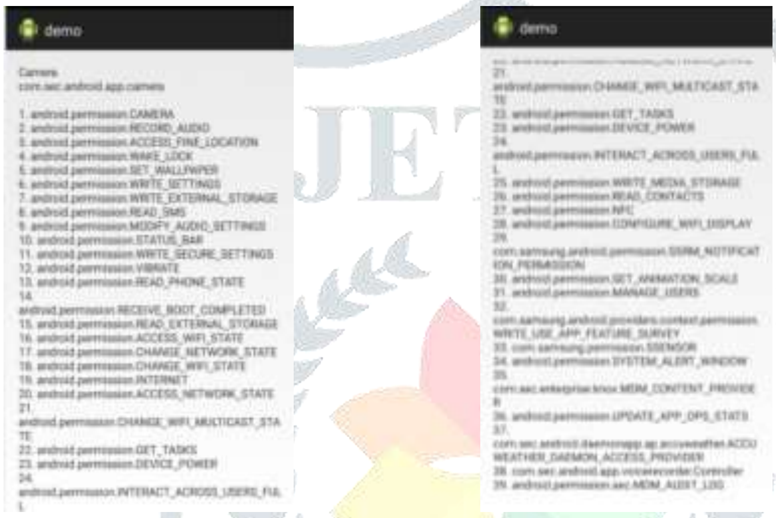
Figure 4: Flowchart of the Proposed Architecture

The features are categorized into three types: permissions which are needed, API calls and dynamic behaviors. Dynamic behaviors are known from the flags of the permissions related to application actions such as information leaks, network and file input/output, mobile phone calls etc., like actions that sends data over the network, makes a phone call, and used to send SMS messages. The decompiled code is examined for java reflection features by searching the methods and constructors present in the code and the application is tested if it is loading classes from the .jar files.

Alertion System: Based on the feature set obtained from application installed, they are classified either as untrusted or trusted applications and an alert message is shown to the user if it is untrusted.

IV. RESULTS AND DISCUSSION

The functionality of the proposed method is tested by downloading different applications from various open source sites and playstore. The keys are combination of both benign and malware applications. When the user installs the application, the features of these applications are retrieved with the help of package manager, a part of application framework in the operating system of the mobile.

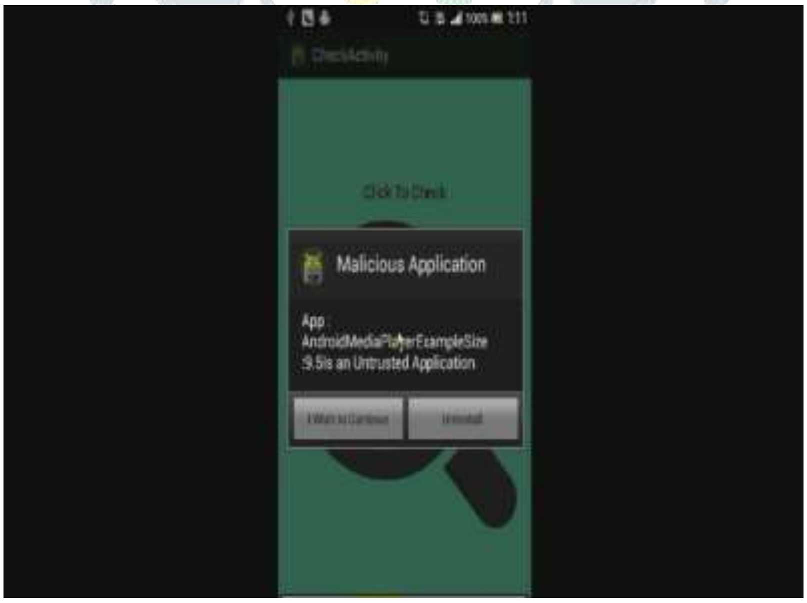


```
Camera
com.sec.android.app.camera

1. android.permission.CAMERA
2. android.permission.RECORD_AUDIO
3. android.permission.ACCESS_FINE_LOCATION
4. android.permission.WAKE_LOCK
5. android.permission.SET_WALLPAPER
6. android.permission.WRITE_SETTINGS
7. android.permission.WRITE_EXTERNAL_STORAGE
8. android.permission.READ_SMS
9. android.permission.MODIFY_AUDIO_SETTINGS
10. android.permission.STATUS_BAR
11. android.permission.WRITE_SECURE_SETTINGS
12. android.permission.VIBRATE
13. android.permission.READ_PHONE_STATE
14.
android.permission.RECEIVE_BOOT_COMPLETED
15. android.permission.READ_EXTERNAL_STORAGE
16. android.permission.ACCESS_WIFI_STATE
17. android.permission.CHANGE_NETWORK_STATE
18. android.permission.CHANGE_WIFI_STATE
19. android.permission.INTERNET
20. android.permission.ACCESS_NETWORK_STATE
21.
android.permission.CHANGE_WIFI_MULTICAST_STA
TE
22. android.permission.GET_TASKS
23. android.permission.DEVICE_POWER
24.
android.permission.INTERACT_ACROSS_USERS_FUL
L
```

```
21.
android.permission.CHANGE_WIFI_MULTICAST_STA
TE
22. android.permission.GET_TASKS
23. android.permission.DEVICE_POWER
24.
android.permission.INTERACT_ACROSS_USERS_FUL
L
25. android.permission.WRITE_MEDIA_STORAGE
26. android.permission.READ_CONTACTS
27. android.permission.NFC
28. android.permission.CONFIGURE_WIFI_DISPLAY
29.
com.samsung.android.permission.SSRM_NOTIFICAT
ION_PERMISSION
30. android.permission.SET_ANIMATION_SCALE
31. android.permission.MANAGE_USERS
32.
com.samsung.android.providers.context.permission.
WRITE_USE_APP_FEATURE_SURVEY
33. com.samsung.permission.SSENSOR
34. android.permission.SYSTEM_ALERT_WINDOW
35.
com.sec.enterprise.knox.MDM_CONTENT_PROVIDE
R
36. android.permission.UPDATE_APP_OPS_STATS
37.
com.sec.android.daemonapp.ap.accuweather.ACCU
WEATHER_DAEMON_ACCESS_PROVIDER
38. com.sec.android.app.voicerecorder.Controller
39. android.permission.sec.MDM_AUDIT_LOG
```

Figure 5: Permissions Extraction from an Application



Malicious Application

App : AndroidMediaPlayerExampleSize 9.5is an Untrusted Application

Figure 6: Alert Message to User

The proposed model extracted different sets of features related to the calls done explicitly to the standard API's, methods related to the Intents and Content Providers. The inherited classes defined by the application are also monitored to differentiate between application defined methods and pre-defined methods by the Android. Both Android framework and linux kernel handle the permission system in Android. Most of the permissions are analyzed and studied from the framework of Android, and other file system and network related permissions are also studied.

V. CONCLUSION

The malware evaluation techniques present clearly indicate that it is a matter of importance to make changes in terms of analysis in malware detection. In this paper, the background application service developed extracts the different permissions from applications and analyses them on startup of the application, as any malware application must have some characteristics that define them as malicious. The features are extracted from the unique ID of the application and analyzed at three different levels. The complete details about the application are extracted and tested for untrusted application permissions and misusing the functionality of permissions. This method is also tested for repackaging attacks. The user is alerted whether it is legitimate or not based on the analysis. In this paper, focus was laid on reducing the false positives and unauthorized access to resources of a smartphone during permission analysis and an attempt is made to alert the user about malicious applications misusing permissions on installation of application.

REFERENCES

[1] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, A survey of mobile malware in the wild, in Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), pp. 3–14, 2011.

[2] Z. Fang, W. Han, and Y. Li, Permission based Android security: Issues and countermeasures, Computers & Security, vol. 43, pp. 205–218, 2014.

[3] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, Whyper: Towards automating risk assessment of mobile applications, in Proceedings of the 22nd USENIX Security Symposium (USENIX Security), pp. 527–542, 2013.

[4] D. Geneiatakis, I. N. Fovino, I. Kounelis, and P. Stirparo, A permission verification approach for Android mobile applications, Computers & Security, vol. 49, pp. 192–205, 2015.

[5] Y. Zhou and X. Jiang, Dissecting Android malware: Characterization and evolution, in Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland), pp. 95–109, 2012.

[6] D. Arp, M. Spreitzenbarth, M. Hbner, H. Gascon, K. Rieck, and C. Siemens, Drebin: Effective and explainable detection of Android malware in your pocket, in Proceedings of the 21th Annual Symposium on Network and Distributed System Security (NDSS), 2014.

[7] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, Semantics-aware Android malware classification using weighted contextual api dependency graphs, in Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS), pp. 1105–1116, 2014.

[8] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth, Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones, in Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2010.

[9] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, Profiledroid: Multi-layer profiling of Android applications, in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 137–14,2012.

[10] Chan, J.T. and Yang, W., Advanced obfuscation techniques for Java bytecode, Journal of Systems and Software 71, No. 2. pages 1-11, 2015.

[11] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick D. McDaniel. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014.

[12] Adwait Nadkarni and William Enck. 2013. Preventing accidental data disclosure in modern operating systems. In 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany,1029–1042, 2013.