



Received: 15 July, 2022  
Accepted: 01 September, 2022  
Published: 02 September, 2022

\*Corresponding author: Seyedmostafa Safavi, Unit of Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia, Tel: +60176938059; E-mail: safavi@takhosting.info

ORCID: <https://orcid.org/0000-0003-3984-5169>

Keywords: Smartphone; Mobile privacy; Permission system; Data security; Android OS; Zygote; Dalvik VM

Copyright License: © 2022 Safavi S, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

<https://www.peertechzpublications.com>



## Review Article

# CenterYou: Android privacy made easier the Cloud Way

Seyedmostafa Safavi\* and Zarina Shukur

Unit of Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia

## Abstract

The Smartphone industry has expanded significantly over the last few years. According to the available data, each year, a marked increase in the number of devices in use is observed. Most consumers opt for Smartphones due to the extensive number of software applications that can be downloaded on their devices, thus increasing their functionality. However, this growing trend of application installation brings an issue of user protection, as most applications seek permissions to access data on a user's device. The risks this poses to sensitive data are real to both corporate and individual users. While Android has grown in popularity, this trend has not been followed by the efforts to increase the security of its users. This is a well-known set of problems, and prior solutions have approached it from the ground up; that is, they have focused on implementing reasonable security policies within Android's open-source kernel. While these solutions have achieved the goals of improving Android with such security policies, they are severely hampered by the way in which they have been implemented them. In this work, a framework referred to as CenterYou is proposed to overcome these issues. It applies pseudo data technique and cloud-based decision-making system to scan and protect Smartphone devices from unnecessarily requested permissions by installed applications and identifies potential privacy leakages. The current paper demonstrated all aspects of the CenterYou application technical design. The work presented here provides a significant contribution to the field, as the technique based on pseudo data is used in the actual permissions administration of Android applications. Moreover, this system is user and cloud-driven, rather than being governed by over-privileged applications.

## Introduction

Mobile devices, Smartphones, and tablets, in particular, have truly become handy companions to many individuals. By simply taking advantage of built-in sensors, portable mobile phones have found applications in large-scale sensing, and the data they capture can be used to analyze social conduct and a multitude of other phenomena [1-3].

Many different new software applications have been developed utilizing geo-location data via GPS UNIT. Software applications such as Instagram enable capturing and sharing photos on different Internet sites, even though Foursquare makes it possible for the user to tag the location. In addition, several commercial agencies, such as Jawbone [4,5], have developed computer hardware that is synchronized with Apple's iOS, thus allowing them to monitor user behaviors.

For example, extant applications can observe a user's resting behavior as well as the number of footsteps made each day, and subsequently compute statistics on energy consumption, health status, and other vital facts that can assist an individual in maintaining a desired lifestyle. While these and many other interesting applications have emerged in recent years, further discussion on their reliability and utility is beyond the scope of the present study. Nonetheless, they demonstrate that Smartphone software applications have become immensely profitable and have rapidly surpassed the functionality of simple geo-location recognition. Each one is intended to meet a specific consumer need and thus gain a share in a very competitive market.

A recent behavioral study of 101 popular Smartphone apps by the Wall Street Journal [6] discussed in detail the types of mobile data that are being tracked and distributed

by these applications with complete disregard for users' privacy. According to the study findings, marketers are using the collected data for compiling a wide array of targeted user analyses. Many users are not aware that large corporations and governments, as well as various strata of unauthorized users, are collecting user data for a multitude of purposes, many of which are questionable and infringe on our privacy [7]. Put simply, mobile devices can be compared to digital leaky buckets. While e-wallet services, location-based services, and numerous popular applications have undeniably made life simpler for consumers, the price they pay for this convenience is a loss of privacy.

While this issue is well known, securing mobile devices is much more challenging [8] compared to traditional computing platforms, since they possess numerous special capabilities, such as personalization, flexibility, and connectivity, and are also equipped with restricted resources. Thus, in order to improve their security, Smartphone OSs, such as the Android operating system [9], a Google open source Smartphone system, deploys several protection components like UIDs, as well as permission labeling and software application signature, along with sandboxing [9–11]. Thus far, most of the Smartphone development focus has been devoted to improving the underlying security infrastructure, while omitting addressing user privacy protection. Current handles in context-aware intelligent products use static data and are also fixed. As a result, in most practical situations, the consumer is actually requested to decide to share sensor data, such as location, or other sensitive information, including user contacts, calendar objects, and network access at install time. As many individuals are unaware of the security risk such actions pose, they share their data and thus endanger their privacy.

Presently, Google's Android operating system is the most popular and most widely distributed mobile phone OS. Android is a great open-source package that enables the development and computer programming of third-party application programs. According to TechCrunch & CNET [12,13], whose analysis is shown in Figure 1, each year, Android's market share is increasing relative to the competitors.

### Android's vulnerabilities

Owing to the growing prevalence of smart gadgets, issues pertaining to protection and privacy are emerging. For example, an increasing number of consumers are using Smartphones for mobile banking and sharing other sensitive healthcare information [1], thus opening up possibilities for unauthorized

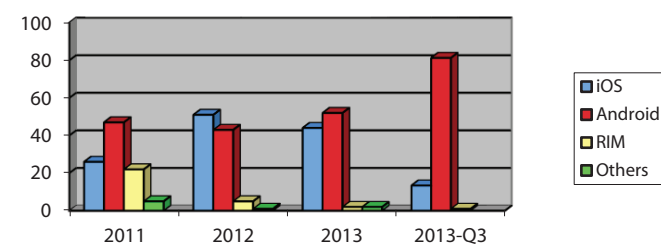


Figure 1: Mobile OS market share

access to their data. A further issue pertains to the computer storage and processing of classified information, which also present significant security and privacy risks. Android is exposed to a growing risk from several different attacks. For instance, it can allow the installation of harmful applications and libraries that misuse user data [14–16], or even use root exploits [14,17] to disable security and access private sensitive data. Several malware applications are also taking advantage of insecure interfaces [18–21] and files [22]; confused deputy attacks [23], or employ collusion attacks [24,25].

On the other hand, Android offers a public marketplace named Google Play, equipped with various tools aimed at preventing malware. In the Android platform, developers cannot directly deliver their applications through Google Play without going through a strict review process. However, application creators can bypass this restriction by uploading their programs to the non-official marketplace [1] (i.e., Applanet, AppBrain, and so on) [26]. Proposed a computerized Android malware recognition system using the results yielded by the sandbox. In another study [27] proposed a particular Android software program evaluation method capable of creating numerous examination situations. Their approach was based on fuzzing a software application and a test bed, which produced the actual analyzed conditions. Hence, applications can be executed in parallel, as several emulated Androids can operate within the cloud server. According to [28,29], a sandbox might also be used to enhance the performance associated with traditional anti-virus programs designed for the actual Android OS.

Mobile phone usage exposes consumers to a wide range of risks, including information leakage caused by phone damage or even robbery, accidental disclosure of information, phishing attacks, network spoofing attacks, monitoring assaults, etc. [30] pointed out that hackers mostly execute email attacks, expecting to retrieve confidential data stored on spam victim's portable device. In practice, hackers most commonly attract their victims by downloading useful applications and popular video games to which they attach the malware source before releasing them into the market. When the users install these on their mobile devices, they unwittingly introduce malware to the system [31]. In 2010, Costin, Raiu, and Kaspersky reported that they discovered TrojanSMS—the first Trojan working with the Android operating system. Within the following 12 months, the number of similar malware applications rapidly expanded, and they evolved into very complex tools that affected cell phones globally. This pattern started to be noticeable within the third quarter of 2011 when Kaspersky discovered more than 1000 applications harmful to the Android operating system. According to the available statistics, this is equivalent to all Smartphone malware that had been created prior to that point [32].

### Privacy

Online stability and security aim to prevent unauthorized data usage while facilitating authorized access. In 1890, Justice Louis Brandeis published the landmark article entitled “The Right to Privacy” attempting to establish the rights US citizens have to privacy [33]. More than a century later, in 1999, Sun

Microsystems chief executive officer Scott McNealy made a prophetic judgment of online privacy, stating “You have zero privacy anyway.”

In simple terms, privacy might be defined as the power of individuals to choose when, how and what type of data about them is revealed to others. In sum, privacy principles [34,35] require that systems minimize personalized data accumulation by, for instance, data anonymization.

### Problem statement

Android’s system architecture and security mechanisms have undergone thorough inspections, and several plug-ins for Android’s access management and control framework are actually suggested to address specific issues discussed below.

Applications are sometimes over-privileged, as many require access to resources they do not need to function. Owing to this feature, such applications increase the impact of vulnerabilities and exposure to risk. The problem this study aims to address is allowing users to manage application permissions without the need for excessive technical knowledge [36]. In addition, the goal is to allow users to spend less time responding to warning messages that ask for permission to access resources on the device [37,38]. It is expected that having proper protection would make Smartphone users more satisfied with the device.

### CenterYou design and development

As noted in the previous paper [39], the DSR method was adopted in this study, comprising five distinct and sequential phases. This section explains the technical design and development, which are described in detail.

### Problem identification and motivation phase

CenterYou aims to provide a new security architecture for the Android operating system that would address the challenges the current system faces. This can be achieved by designing a security framework, which would serve as an appropriate ecosystem for different security and privacy-protecting models. In the first step, it is essential to gain the necessary knowledge of Android Dalvik Virtual Machine while applying the CenterYou service.

The Dalvik Virtual Machine is the software that actually executes Android applications. Android applications are typically written in Java. Thus, the developers first write their applications in the Java programming language, allowing the Java compiler to compile the Java source code files into multiple Java bytecode files. Next, a tool called Dex transforms the Java byte codes into a single file, which is in a different byte code format called dex. This bytecode file is usually called *classes.dex*. Next, the dex file is packaged with other application resources and is installed on the device. Finally, when the user launches the application, the Dalvik VM will execute the *classes.dex* file.

Xposed extended the `/system/bin/app_process` executable to load a JAR file on startup (hooking). The classes of the Xposed

file will thus be included in every process (including the one for system services) and can act according to the powers they are given. Xposed also allows developers to replace any method in any class with the help of the JAR file. Thus, using Xposed, the developer may force the application to use a different API, due to the ability to manipulate the classes through Xposed.

The CenterYou framework, with the help of Xposed, can change the parameters for the method call, modify the return value or skip the call to the method completely. The aim of CenterYou is not to change application permissions, but rather to replace real, privacy-sensitive data with fake data, thus increasing user protection from malware and over-privileged applications. In particular, the advantage of the CenterYou framework stems from the fact that it does not make any changes to the Android architecture. Rather, it merely intercepts function results by hooking functions. Method hooks can prevent the original method from executing, alter the parameters before executing the original method, and alter the result of the original method, as shown in Figure 2.

### As shown in Figure 3, CenterYou includes:

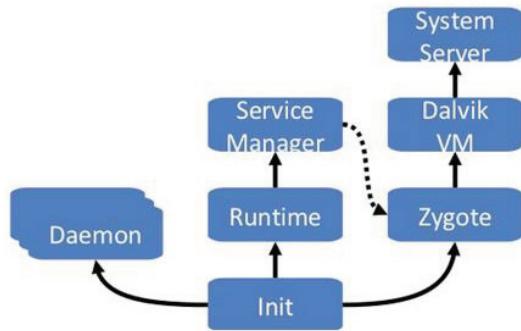
- Privacy database: The SQLite DB to store restrictions and settings set by the user, or to respond via the cloud service.
- Usage database: This database stores information about method usage (restricted or not), and keeps all processes and activities documented.
- Privacy service: This service provides secure cross-process access to the privacy database shown in Figure 3.
- User interface: This is the context in which interactions between humans and applications occur. It is also used to manage the privacy database and view the database usage.
- Cloud smart decision-making system: The most important job of the cloud in the CenterYou application is to manage the application permission list and send data to devices. However, the GCM service is also included in the cloud, allowing improved one-way communication with the application user.
- Notification service: This process helps provide enhanced control and manage permission notifications, as well as facilitates the one-way secure notification functionality.

The CenterYou application installation and execution process is explained below, providing all steps that are performed on the user’s device.

### CenterYou applied android file system structure

CenterYou, when applied to Android, uses several directories that are identical to the original Android file system structure depicted in Figure 4. The main function of directories is to organize files and folders. The main six directories are

### Original Process



### CenterYou Process

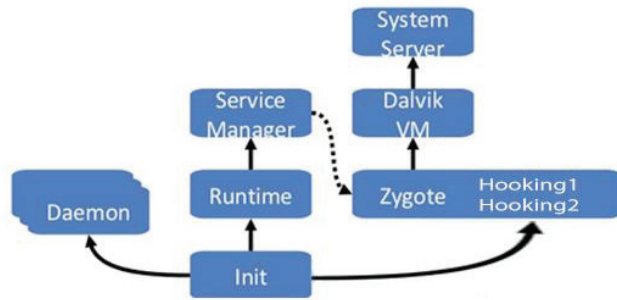


Figure 2: Comparison between the original and the CenterYou framework.

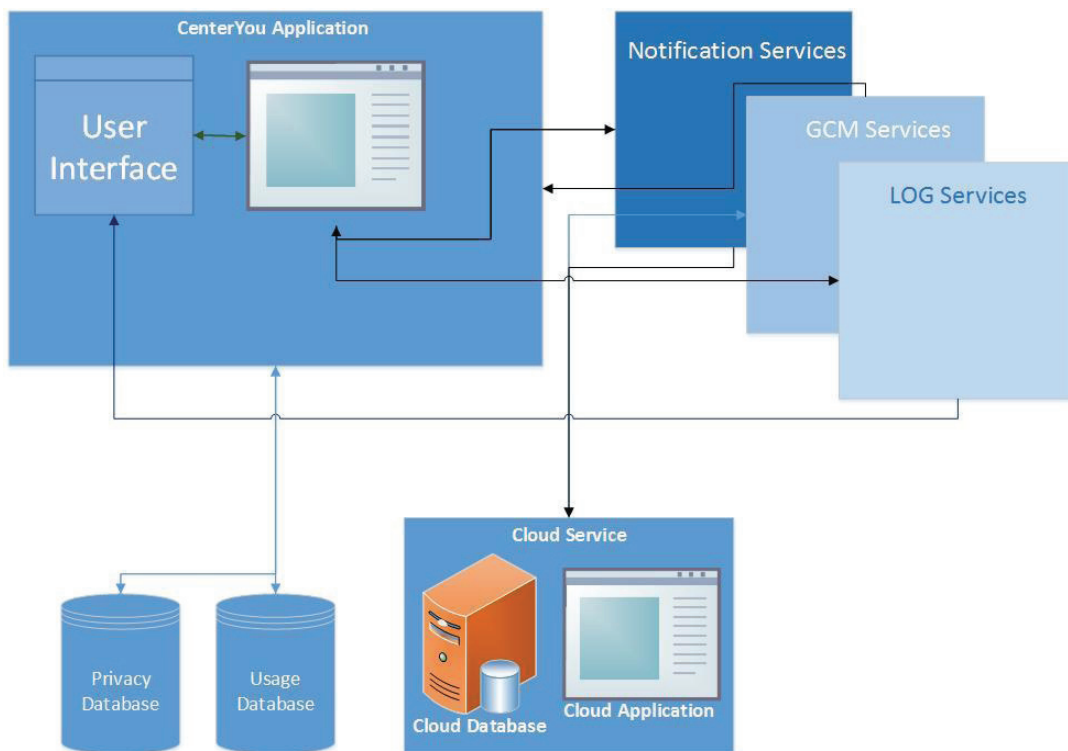


Figure 3: CenterYou architecture.

also present in CenterYou applied to Android devices. The only exception is that CenterYou creates the new sub-directories and files, required for managing application permissions. In addition, they enable making a copy of almost all application permission pseudo data configurations. Figure 4 depicts the directory list for this new File System.

### CenterYou application installation process flow

#### I. General process

- a. **Package manager and package installer:** The CenterYou application does not change any parts of PackageInstaller, which uses a user interface (UI) to manage *applications/packages*. PackageInstaller calls

InstallAppProgress activity to receive instructions from the user. Next, InstallAppProgress makes a request to the Package Manager Service to install the package via *installed*.

The CenterYou framework modifies the Package Manager, whereby the set of permissions in the *system/CenterYou* folder is duplicated to ensure that every single authorization possesses both the 'Real' and a 'Pseudo' type.

When the CenterYou application is installed, all requested permissions (such as read phone status, read contacts, full network access, run at startup) are granted. Thus, at this stage, CenterYou will inject the first and the last part of the package

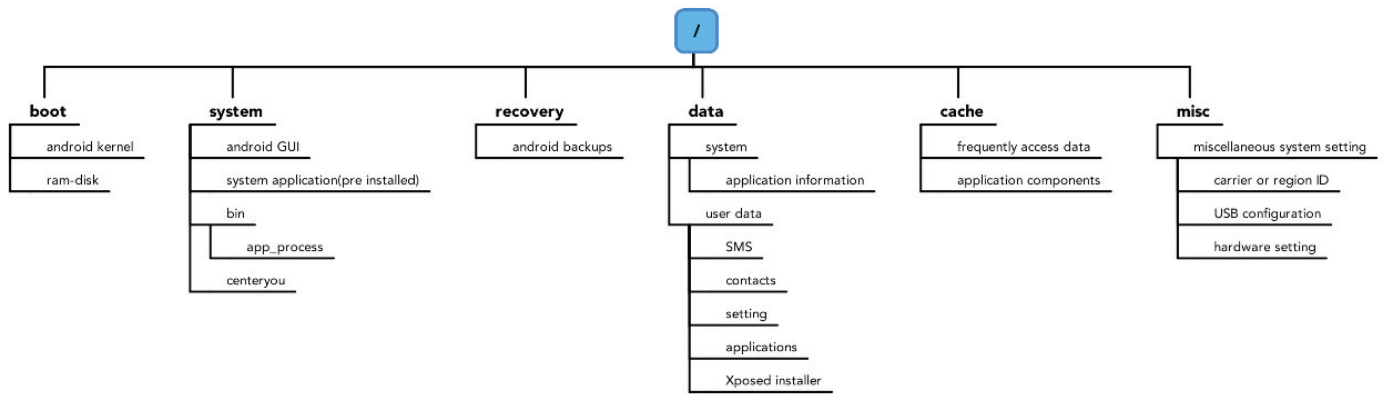


Figure 4: CenterYou applied to Android file system structure.

available in the Dalvik Machine. In doing so, CenterYou changes the permissions requested and places them into the new permissions group. The modified version of the Package Manager service uses the Linux kernel to notify the service to monitor changes to files in this directory and updates its in-memory cache of pseudo permissions.

**b. APK files storage location in Android for CenterYou:**

CenterYou will be installed in `/system/app/` and the configuration file will be created in `/system/CenterYou` to hold a copy of all permissions related to all applications already installed or about to be installed on the same device. The researcher chose to make this directory in `/system/` because of the need to keep the information safe from other applications and non-technical users.

**c. APK installation process:** The installation process involved in this case is similar to that of using the Google Play Store. However, as this application is not registered in Google Play Store, the user must download it from the CenterYou website and install it as a third-party application. After receiving the APK file, the user must click on it and install the application by accepting the permissions list (read phone status, read contacts, full network access, run at startup). Upon completion of the aforementioned steps, the installation will start for the first time. The CenterYou application can be installed on all Android-based devices that have root access, with a version greater than Android 4.0.1.

## II. Process details

**a. APK installation process in detail:** The CenterYou application installation process flow was explained in the last section, while this section focuses on its technical aspects. As can be seen in Figure 5, Package Manager Service will be involved in the APK installation process, which executes in Package Manager Service.

**b. Detailed process for package manager:** To appreciate the differences between CenterYou and the original architecture for the Package Manager, in the list below, the new items are marked by the “NEW” logo. The new process for package managers is as follows:

- Wait
  - Add a CenterYou package to the queue for the installation process
  - Determine the appropriate location of the package installation
  - Copy the apk file to `/system/app/`
  - Determine the UID of the app
  - NEW Create a folder for the new app in the `system/CenterYou`
  - Request the *installed* daemon process
  - Create the application directory and set permissions
  - NEW Copy the application directory and set permissions in the `system/CenterYou`
  - Extract the dex code to the cache directory
  - Reflect and packages.list / system / data / packages.xml the latest status
  - Broadcast to the system, along with the name of the effect of the installation as a complete package
  - NEW Execute `Intent.ACTION_PACKAGE_ADDED`
- c. Package manager and data storage:** Package Manager stores the CenterYou application information simultaneously in `/data/system` and in `system/CenterYou`. The `centeryoupackages.xml` file stores (1) the Name of the package, (2) Permissions, and (3) Package (application) code. The code is helpful while importing the CenterYou settings from the device storage or a designated cloud service because, at the time of the import, the CenterYou application can import the settings, while comparing them with the application code recorded in `centeryoupackages.xml`.

## CenterYou application execution and dalvik virtual machine

CenterYou application uses Zygoter to enable the injection

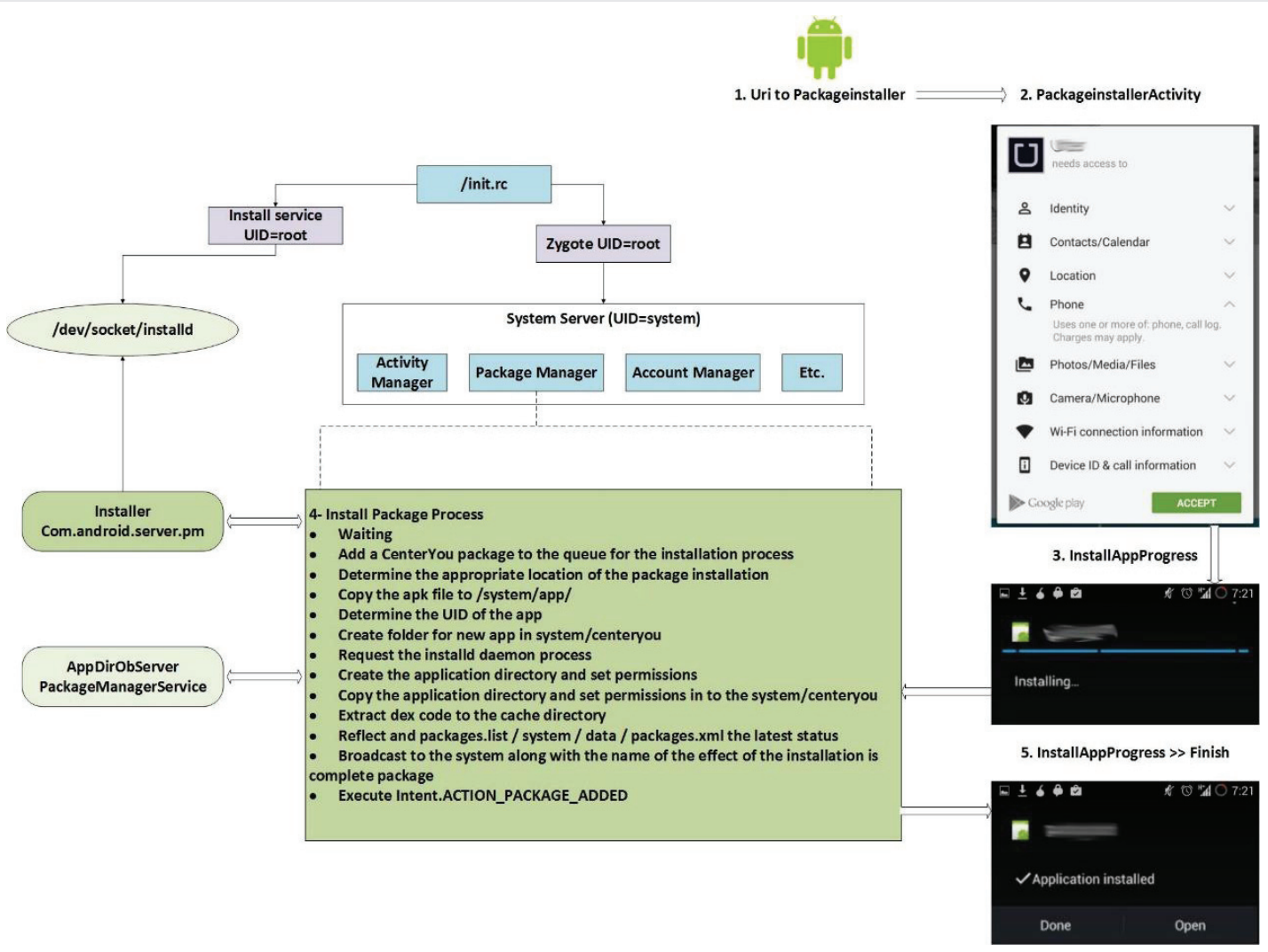


Figure 5: APK installation process (Google 2012).

service. Once the framework installation is complete, an extended *app\_process* executable is copied to */system/bin*. This extended startup process adds an additional jar to the classpath and calls the methods contained within. For instance, this process can occur just after the VM has been created, even before the main Zygote method has been called. In addition, inside this method, Xposed is part of the Zygote and can act in its context. The jar is located at */data/data/de.robv.android.xposed.installer/bin/XposedBridge.jar* and is called at the beginning of the process. Some initializations are performed there as well, along with loading the modules, as shown in Figure 6.

Once the code is loaded, the new permissions list can overwrite the application permissions and allow the user to easily change the settings pertinent to over-privileged applications by introducing a new list that is set up through the CenterYou application.

**Cloud system**

CenterYou’s goal is to be simple enough for a non-technical user to benefit from it while obtaining similar quality of protection as has been proposed in extant studies in this field.

One of the differences between CenterYou and other solutions discussed as a part of the related study is that the former benefits from the cloud-based support system. This feature allows the user to keep the application in Autopilot (easy mode) and thus not have to worry about protecting his/her privacy from newly installed applications. One of the best examples of this service is antivirus software that has been in use for a long time. Still, the key difference between the CenterYou framework and any antivirus software currently on the market is the service they are supposed to provide to their respective users.

The cloud service is responsible for handling and managing all application permissions lists, sending notifications (settings and one-way secure messaging) to a particular user, and having space for backup of permissions lists sent by each device application. To perform the aforementioned functions, this researcher used Amazon Web Services (AWS) [40,41] for the cloud service and PHP and Java programming languages to design and develop the website and notification system under the GCM service [42,43].

In this research, the database of permissions lists is

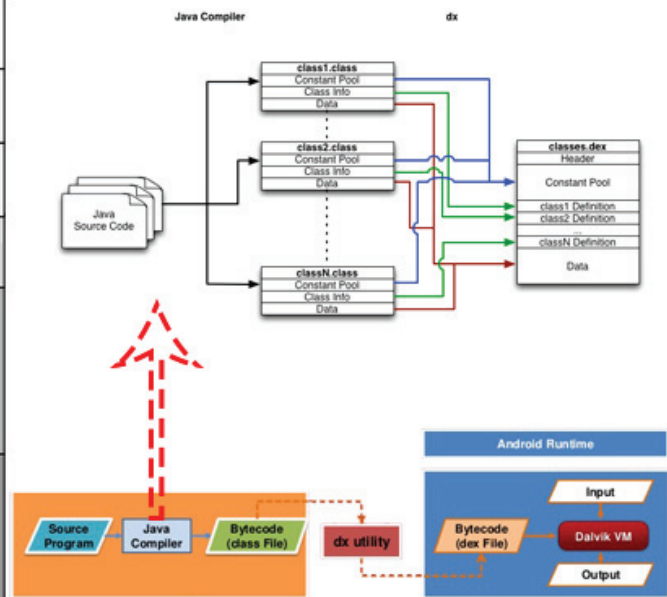
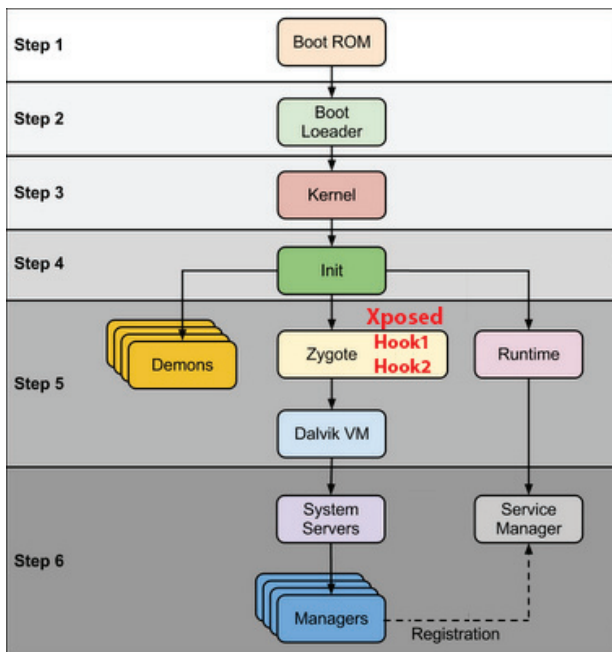


Figure 6: CenterYou applied architecture for application.

designed and recorded in the cloud to make the service more accessible and reliable, by updating this database frequently. After the user confirmation, if the user has selected the easy mode in the CenterYou application menu (in the Android device), the cloud service will receive a unique ID of the device and record it in a separate database. This ID will be the only way to identify the device remotely. After the first setting-up stage, the cloud service will be on standby, ready for the application permissions list process, listening to application requests. While in this mode, it can process and send secure notifications to the specific application user.

After receiving an application permissions list request from a particular application, the service will record this request in a different database. Next, it will search the database of the existing apps and, if the requested application is available in that database, the relevant information will be retrieved. Using the notification service, the cloud will send a reply to the device that, in this context, is viewed as an information requester. Conversely, if no record of the application is found in the database, the service will flag the request for an immediate response from the moderator and will send a notification to the user's device, in addition to a "block" command. In doing so, it will make sure that the user will not use the application until the moderator has had the opportunity to specify the best permissions list for it. These processes are shown in Figure 7.

Cloud database structure design for the CenterYou automated decision-making system service is shown below:

**a. User Database Structure**

Field\_name

centeryo\_filexmi.user.imei

centeryo\_filexmi.user.file\_name

centeryo\_filexml.user.created\_on

**b. Application Database Structure**

**CenterYou application database structure**

centeryo\_privacy.app.id

centeryo\_privacy.app.application\_name

centeryo\_privacy.app.package\_name

centeryo\_privacy.app.package\_version

centeryo\_privacy.app.package\_version\_code

centeryo\_privacy.app.modified

**CenterYou per application database structure**

Field\_name

centeryo\_privacy.id

centeryo\_privacy.android\_id\_md5

centeryo\_privacy.android\_sdk

centeryo\_privacy.version

centeryo\_privacy.package\_name

centeryo\_privacy.package\_version

centeryo\_privacy.package\_version\_code

centeryo\_privacy.restriction

centeryo\_privacy.method

centeryo\_privacy.restricted

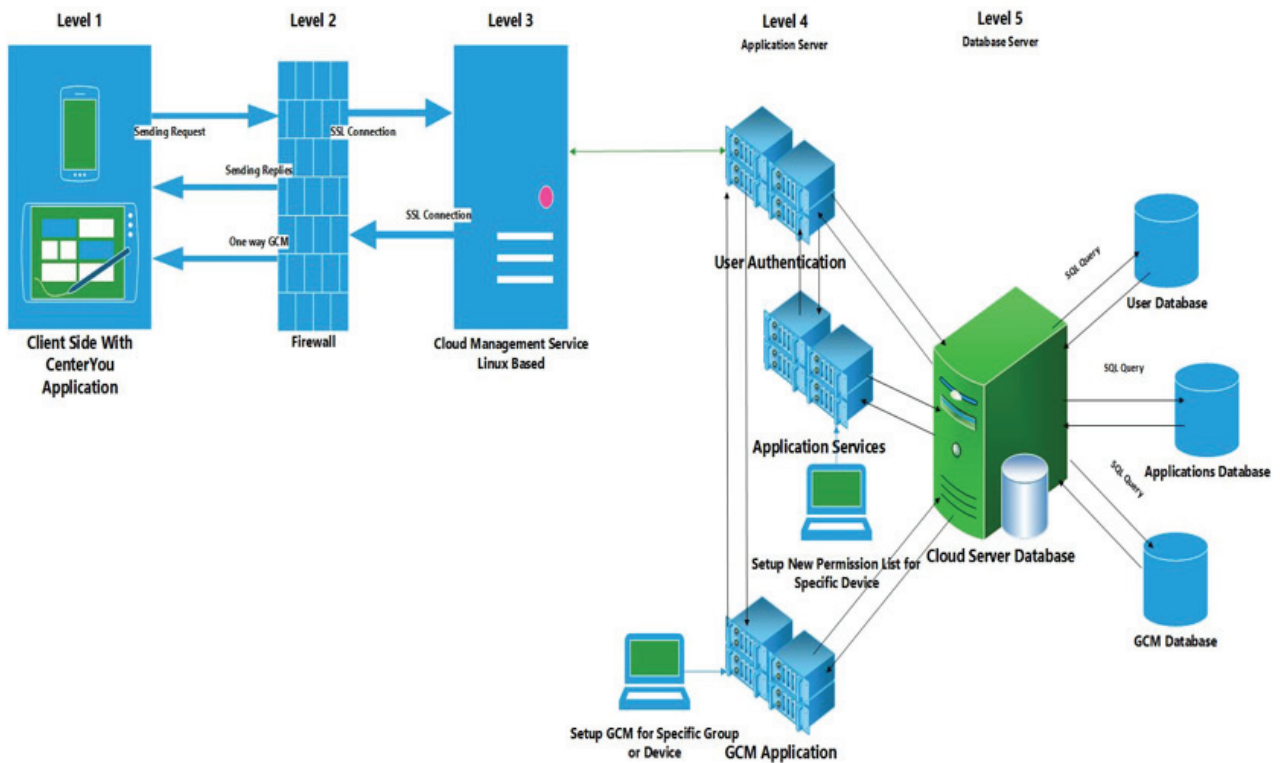


Figure 7: Cloud side.

centeryo\_privacy.allowed  
 centeryo\_privacy.used  
 centeryo\_privacy.modified  
 centeryo\_privacy.update

### C. GCM Database Structure

Field\_name  
 Centeryo\_filexmi.tbi\_gcm.gcm\_id  
 Centeryo\_filexmi.tbl\_gcm.gcm\_token\_id  
 Centeryo\_filexmi.tbl\_gcm.gcm\_imei  
 Centeryo\_filexmi.tbi\_gcm.gcm\_status

### CenterYou Application Management through GCM Database Structure

Field\_name  
 centeryo\_filexmi.tbi\_appinfo.app\_id  
 centeryo\_filexmi.tbl\_appinfo.app\_name  
 centeryo\_filexmi.tbl\_appinfo.app\_package name  
 centeryo\_filexmi.tbl\_appinfo.gcm\_id  
 centeryo\_filexmi.tbl\_appinfo.status  
 centeryo\_filexmi.tbi\_appinfo.created\_on

### CenterYou architectural design and algorithm aspect in details

CenterYou, developed as a part of this research modifies and injects the new set of permissions at the start of each API call, with the help of the Xposed method. The application first checks whether the user has changed the permissions or the cloud is in charge of controlling them, depending on whether the user has selected the “Advanced” or “Autopilot” option, respectively. It maintains a separate state for each application, thus enabling the user to prevent specific applications from accessing the device resources while granting access to others. In the case of pseudo permission, the API call provides a fake result to the application. Design rationales can often warrant design judgments made as a part of the artifact development by means of showing the design flow, along with the causes of every step [44].

### Features that are not included in centeryou

The CenterYou framework cannot serve as antivirus and malware protection and thus cannot replace other services specifically designed to protect the phone from viruses and malware. This framework simply aims to protect personal information from leakage due to the permission requests made by over-privileged applications.

In other words, the research objective is to manage the manner in which permissions are listed and control them inside the user’s Smartphone. More specifically, the intent of this research is not to check the system security, and only



the framework design is used to prevent providing services to over-privileged applications that are not required for their installation and operation.

### Features in privacy implementation

After an extensive review of “Challenges, methodologies, and issues in the usability testing of mobile applications” and “Permission Tracking in Android” that have been provided by other authors [45,46], the researcher was able to discover the most popular points for considering features in privacy implementation within the existing frameworks. These are availability, user confirmation, pseudo technique, cloud support, log, and GUI.

Availability is the key to helping non-technical users to attain the required privacy while using legitimate applications provided by legal sources. In this work, two steps have been taken to apply availability to the design of CenterYou. First, CenterYou is separate from the Android operating system. Thus, in order to achieve this objective, CenterYou is designed as an APK file that can be installed and used standalone. Second, the path files that should be applied to the Android OS are addressed in this work with the help of the Xposed method. Thus, the CenterYou APK file and the Xposed method work jointly to separate the method and the application from the Android operating system.

In addition, user confirmation is added to the CenterYou application in order to help end-users to confirm access to a data resource through a specific application. To do this, the CenterYou application design includes two optional choices, manual (Advanced) and easy (Autopilot) mode. If the user asks for the manual option, the application will run manual settings and a manual menu, allowing the user to choose the settings as a standalone process, with no help from the cloud service. In this case, everything will be fully manual, and all auto-corrections and settings will be disabled in the CenterYou application. Nonetheless, if desired, the user is still given the option of utilizing the cloud services manually.

Permissions to access a particular data resource can be either revoked or granted by *Pseudo Technique*. This is a very effective way to handle such cases, as the application to which this applies will not recognize that the real access did not take place and will keep running as expected. To manage *Pseudo Technique* in the CenterYou application, while the application setting is enabled, CenterYou provides a predesigned library that protects the real information from unauthorized access. For instance, if the application requests information on location and “location pseudo” is enabled, the request will be redirected to the CenterYou library and the result will be taken from the fake setting that has been set up previously (either by the cloud smart decision-making process or manually by the user, depending on the setting applied in CenterYou).

With the help of the cloud support system, the application can be provided the latest information that has been set up and applied in real-time. To make a connection between the cloud and the CenterYou application, the researcher used the

fetch method to take data from the MySQL database located on the server to the device. In addition, the notification is run from the cloud to the CenterYou application, whereby GCM programming has been applied to the service provider as well as the application. PHP has been used as the main programming language in the cloud service provider.

To record processes handled by the CenterYou application, or to report a bug from the application to the cloud service, the application has a feature that is executed in order to save the debug log or log an error report in a text file. This menu is provided to the user, whereby he/she can choose to erase the data or send it to the service support team, for more details on bug reporting to the cloud consultation service.

A Graphical User Interface (GUI) is a computer interface that allows users to interact with a device through graphical elements, such as pictures and animations. To increase user satisfaction with CenterYou and make the application easier to use, in this study, the design and development were based on a simple user interface (UI), designed by Eclipse, using Java programming, as well as an integrated development environment (IDE) and software development kit (SDK). With the help of applications and tools, the GUI was designed to mimic the most popular formats favored by Smartphone users.

### Architectural design for CenterYou

The architectural design of CenterYou is outlined below, with the clear identification of all functional components, as shown in the following figures, which illustrate the architecture of CenterYou (in UML notation) of the CenterYou framework.

#### a. Class diagram

Class Diagram provides an overview of the CenterYou framework by describing classes inside the system and the relationships between them. The class diagram shows that the CenterYou framework has two parts, comprising the classes that create the CenterYou framework, and XHook classes that bring the pseudo feature to the CenterYou framework, respectively. The list of classes is shown in Tables 1,2.

The list of classes used for the pseudo feature is shown in Figure 8.

These classes and their inter-relationships are briefly explained in Figure 9, allowing the users to gain a better understanding of the internal design of CenterYou. As can be seen, the researcher has presented the application and pseudo-classes separately, in order to make a clear distinction between the two parts.

#### b. Sequence diagram

A sequence diagram is an interaction diagram that shows how processes operate and interact with one another in the CenterYou architecture. In addition, it enables the reader to visualize the order of every process incorporated into the design. Figure 10 depicts object interactions arranged in sequential order. To start a brief description of this phase of



**Table 1:** List of CenterYou main classes - part 1.

Activity Manager	ApplicationEx, PrivacyService, ActivityMain, ExportActivity, ActivityBase, ActivityApp, TypeActivity, ActivityUsage
Window Manager	CustomMultiPartEntity, management
Content Providers	CRestriction
View System	
Package Manager	restriction, Requirements, ActivityShare, CSetting, SettingsDialog, PackageChange, SharedPreferencesEx, DeviceAdministratorReceiver
Telephony Manager	
Resource Manager	UpdateService, PrivacyProvider
Location Manager	
Notification Manager	OnetimeAlarmReceiver, RState
Others	Meta, PSetting, Hook, SplashActivity, ScoreActivity, ApplicationInfoEx, PrivacyManager, BootReceiver, IniFile, Util

**Table 2:** List of CenterYou main classes - part 2.

Activity Manager	XActivity, XProcess, XActivityManagerService, XActivityManager, XClipboardManager, XMediaRecorder, XApplication, XActivityThread, XActivityRecognitionClient	XHook
Window Manager	XWebView, XWindowManager, XNetworkInterface, XWebSettings	
Content Providers	XContentResolver, XAdvertisingIdClientInfo, XAccountManager, XGoogleAuthUtil, XSettingsSecure, XSensorManager, XSmsManager	
View System	XSystemProperties, XEnvironment	
Package Manager	XPackageManager, XConnectivityManager, XRuntime	
Telephony Manager	XTelephonyManager	
Resource Manager	XBluetoothDevice, XCamera, XWifiManager, XInputDevice, XNfcAdapter, XNetworkInfo, XAudioRecord, XProcessBuilder, XBluetoothAdapter, XResources	
Location Manager	XLocationManager, XLocationClient,	
Notification Manager	XAppWidgetManager,	
Others	XParam, XIoBridge, XUtilHook, XInetAddress, XBinder	

research, it is essential to first discuss the standard Android runtime and the processes followed. This leads to the next stage, where new steps that bring the change to the Android runtime by installing the CenterYou application are defined, as shown in Figure 11. The diagram starts with some definitions pertaining to every aspect of the sequential diagram.

- **Init** is the starting point of all Linux applications and Zygote Android.
- **App\_Process** starts the Zygote and other Java programs. The code is found in *frameworks/base/cmds/app\_process/app\_main.cpp*, specified in *init.rc*.
- **AndroidRuntime** Here, this pertains to the runtime period during which a computer program is executing

Runtime library (in the program lifecycle phase), a program library designed to implement functions built into a programming language

- **RuntimeInit** is a startup mode from

“*com.android.internal.os.RuntimeInit*” and is performing the *startVM ()* and *startReg ()*

In order to facilitate a better understanding of the CenterYou design architecture, it is schematically represented in Figure 11, while the additional items are defined below:

- **The process** first checks whether Xposed is disabled. This function is performed by reading the */data/data/de.robv.android.xposed.installer /conf/disabled* file, to determine whether the Xposed framework is disabled. Thus, if there is a new version of XposedBridge, it is renamed *XposedBridge.jar* and a “false” result is returned. Next, a location of the *XposedBridge.jar* file is sought to determine whether there is such a file, and if not, a “false” result is returned. Otherwise, the *XposedBridge.jar* file is added to the CLASSPATH environment variable, and the “true” result is returned.
- **CenterYou Xposed** framework has a hook method as its main function. When the Android system is in the startup process, the Zygote process will start loading XposedBridge. For this, it will need to replace all methods with JNI methods. For example, *hookMethodNative* is replaced by *Native xposedCallHandler*, *XposedCallHandler* is transferred to *handleHookedMethod*, etc. Here, the Java method performs the user-specified Hook Function.
- **XposedBridge** has a private, native method *hookMethodNative*. Thus, it will change the method type to “native” and link the method implementation to its own native, generic method. As a result, every time the hooked method is called, the generic method will be called instead, without the caller being aware of this change. Here, the method *handleHookedMethod* in XposedBridge is called, passing over the arguments to the method call, such as reference, etc. Thus, this method takes care of invoking callbacks that have been registered for this method call.

*HandleHookedMethod* will be the hook code that is returned to the Java layer.

```
private static Object handleHookedMethod(Member method,
Object this Object, Object[] args) throws Throwable {
    if (disableHooks) {
        try {
            return invokeOriginalMethod(method, this Object, args);
        }
        catch (InvocationTargetException e)
        { throw e.getCause();}}
```

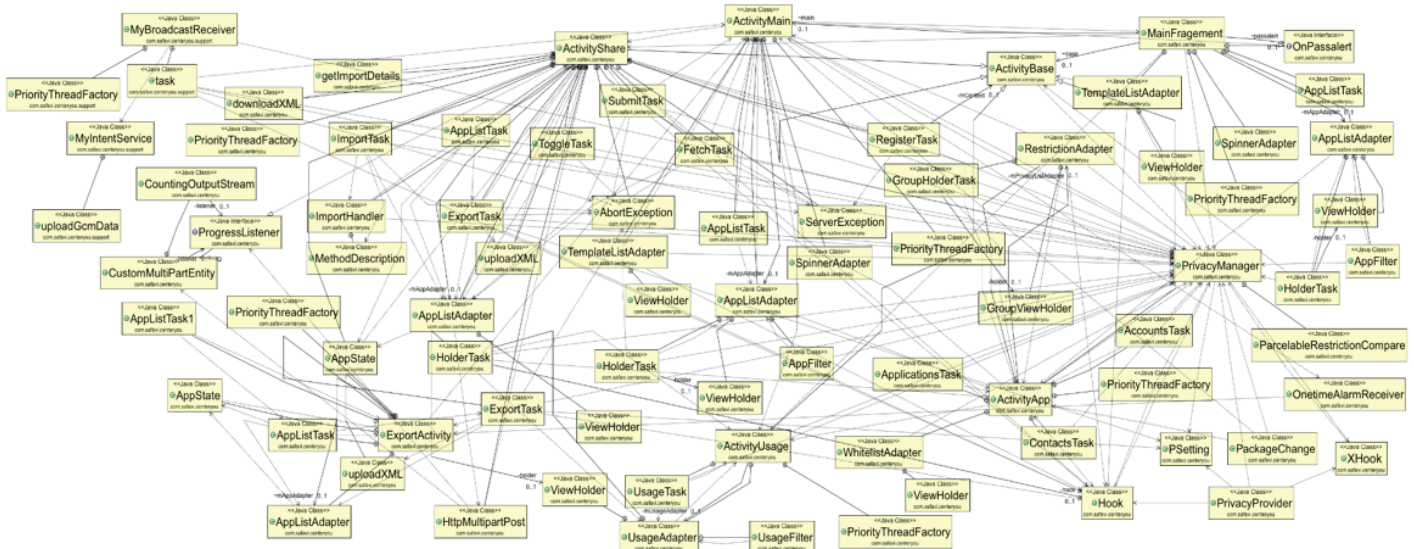


Figure 8: List of original classes.

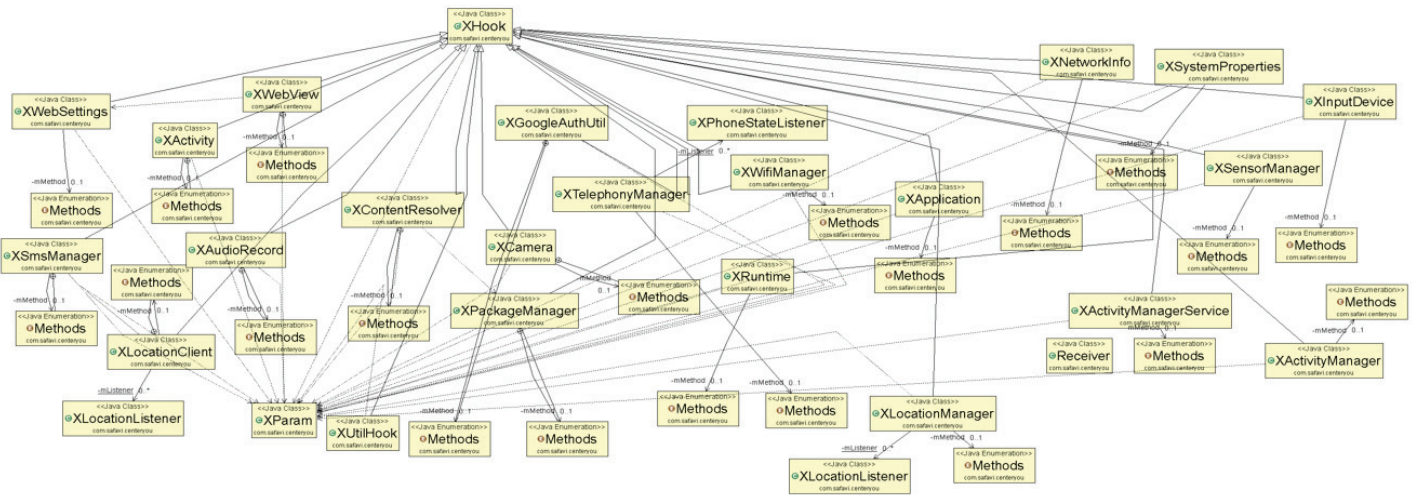


Figure 9: List of pseudo-classes.

`app_main()`, `runtime.start` calls the main function of `XposedBridge`, thus pre-loading resources to process the request and the running application. Hence, to complete the initialization in `XposedBridge`, it is also necessary to complete the call to `ZygoteInit.main`, as shown in the following code:

```
// call the original startup code
if (startClassName not null)
    RuntimeInit.main(args);
```

Figure 11 summarizes the steps taken by the Android application to conduct the runtime performance in a device to which CenterYou has been applied.

### Centeryou individual and organization perspective

The CenterYou framework design aims to cover all aspects

of user privacy. In order to achieve this goal, the framework has a special design that can be used in both individual and organization privacy protection solutions.

### Individual perspective

Application framework design aims to keep users' personal information safe. In order to do so, the framework needs to be controlled by a cloud or via the "Advanced" option, using manual settings, which allow the user to control the application permission settings.

When an individual is using the CenterYou framework, the user has the ability to obtain the permissions list from the cloud smart decision-making system and apply them with the help of the CenterYou framework. This service allows the user to make a backup of the settings information list by choosing either an automatic or manual backup session in the same menu.

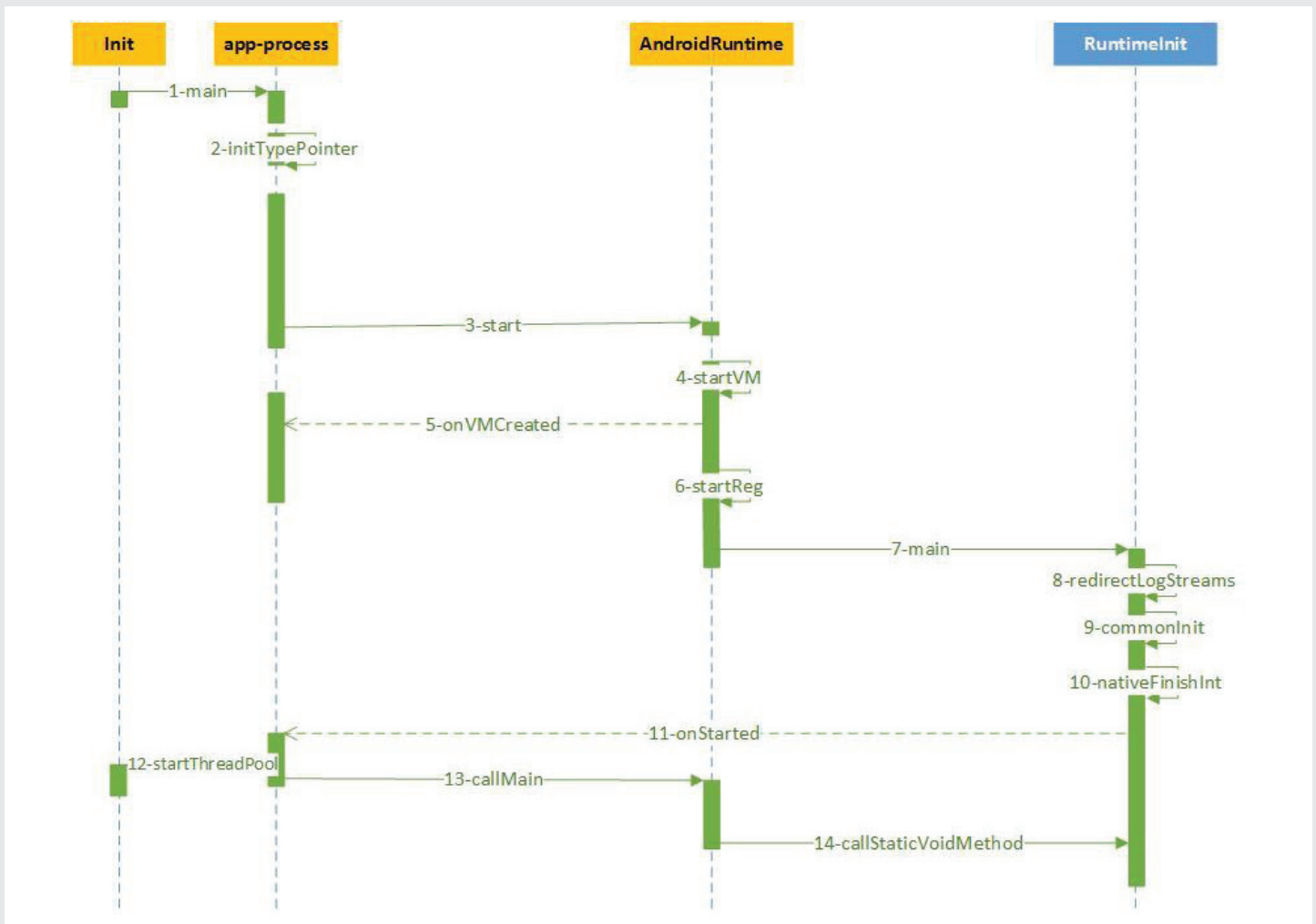


Figure 10: Standard Android Runtime.

The most important function of the cloud in this context is controlling the application installation, taking backups, and restoring the information design for the same device at any time. In Figure 12, the complete process is explained from the perspective of an individual user.

### Organization perspective

Owing to the prevalence of technology, every business needs to apply security levels that protect organizational information from unauthorized access or information leakage. The latter is particularly an issue on mobile devices, as they can be used by hackers to collect data from companies and use their sensitive information to disrupt their operations or create new technology by benefiting from the novel ideas of others.

For this reason, the CenterYou framework offers proper privacy protection levels on every Android device that organizations offer to their employees. The main difference between individual and organization frameworks stems from the management service that allows organizations to set up the server of their own choice and allow their own IT department to manage the information flow.

Another important aspect that differentiates this framework

from the individual one is the control that can bring a new level of protection to the company. For example, if the organization has a rule of not using the Facebook website and applications during working hours, the IT department may have the opportunity to ban the application execution at that particular time with the help of the cloud smart decision-making system.

The third and final important characteristic that makes this application different from that intended for individual use is the one-way notification service that allows the organization to specifically secure contacts with single employees, groups, or all staff at once. The process shown in Figure 12 addressed specifically the organization protection framework. As can be seen, with the help of these features, organizations can ensure complete protection of their data, preventing unauthorized access and information leakage.

### Types of users involved

To ensure that the organizational management is completely secure, the framework should address the user characteristics and the data needs of various user classes to have. For this reason, users are classified into three categories, (1) normal employees, (2) employees working in specific departments, and (3) moderators and administrators. Only the

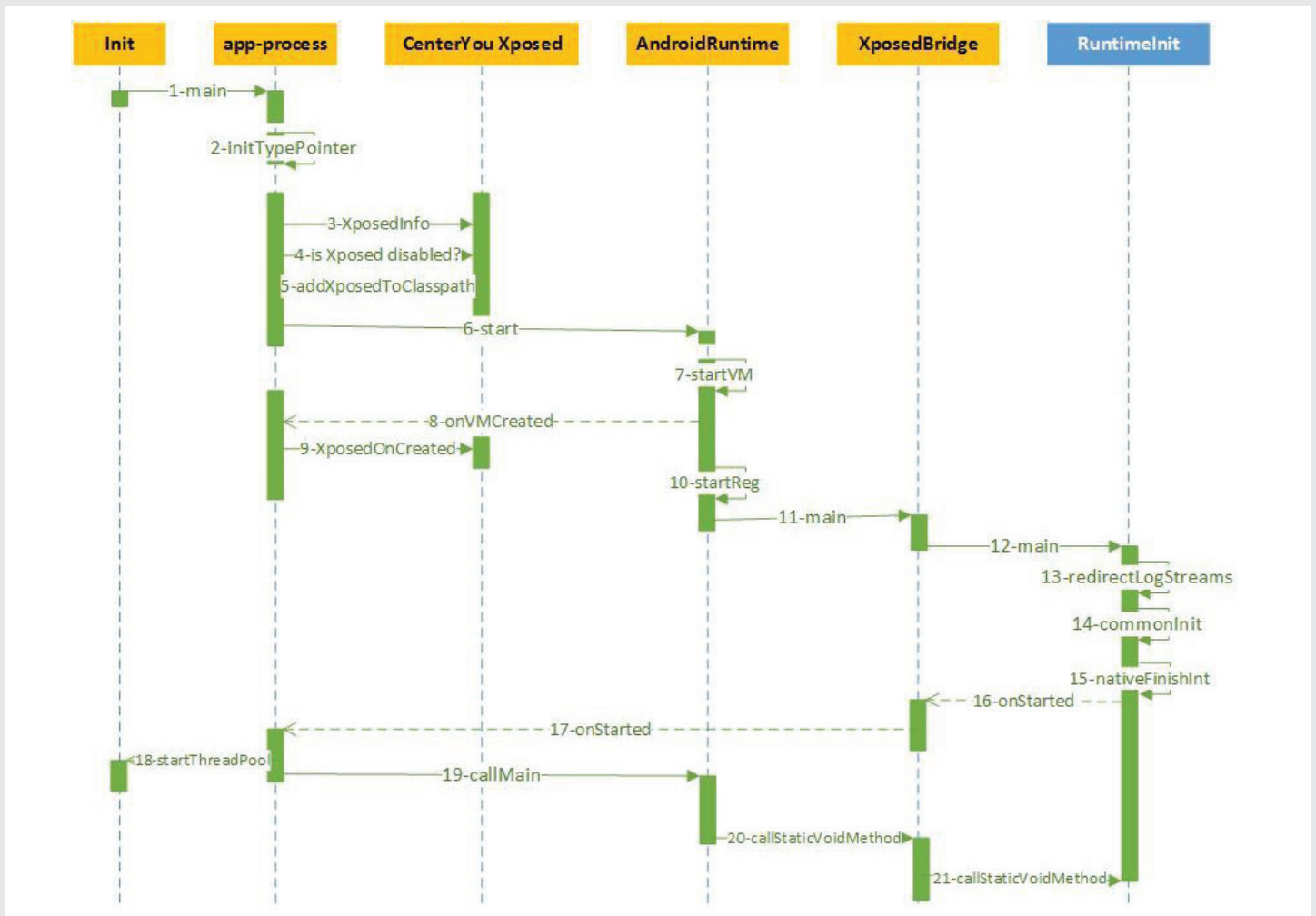


Figure 11: CenterYou Android runtime.

last user group has the ability to change the rules and policies of the framework. Figure 13 is designed with respect to this framework's usability.

### Related work

This section presents various means to increase protection from Smartphone applications that are increasingly requiring more detailed data. The researchers that have thus far attempted to address this issue have provided security plugins aiming to help solidify the privacy and security of mobile phone operating systems.

For example, MockDroid designed by [47], and TISSA developed by [48] provided security protection by introducing fake data directly into API calls created by applications. While these applications could still operate, the authors prevented them from accessing users' personal data. While faking data seems to be an easy and effective way to increase user security [49] developed AppFence involving TaintDroid, whereby they authorized end-users to identify methods that should be executed on their devices. Another approach is based on a hashed phone recognizer, whereby the information passed onto the application cannot be related to the actual end-users. On the other hand, software program designers can

still monitor software program utilization. Nauman, et al. [50] suggested Apex that offers management of useful sensitive resource consumption according to a specific situation as well as runtime constraints, including the exact position of the Smartphone or even the periods of time during which the data resource can be utilized. For this purpose, the researchers applied a long bundle installer service, referred to as Poly, which enables end-users to establish their own policy when setting up the application [50].

Jeon and colleagues suggested another option, which would enable a more widespread utilization, whereby the actual bytecode for Smartphone applications is edited, rather than changing the actual Android operating system [51]. While accessing sensitive resources, the applications modified in this manner exchange information at the privacy proxy level, rather than directly accessing Android APIs. In order to increase user security, Pearce, et al. [52] suggested adopting privilege separation regarding cell phone software applications as well as marketers within the Android operating system. According to the authors, presently, around 56% of software applications utilize users' place and location data simply to offer advertisements [52]. Thus, they proposed unifying all mobile advertisement libraries into process services, which could be built into the Android operating system. Based on

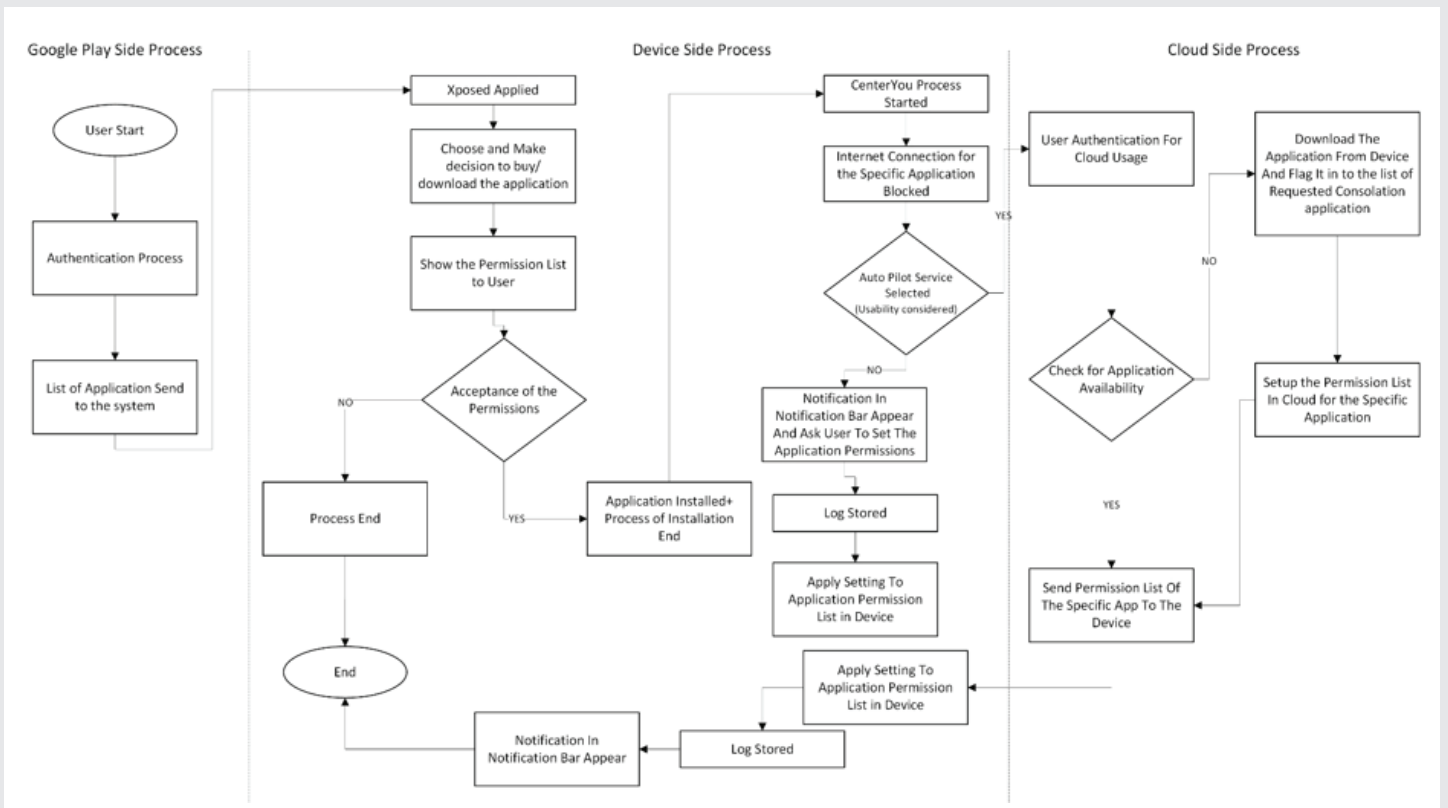


Figure 12: CenterYou Individual Perspective.

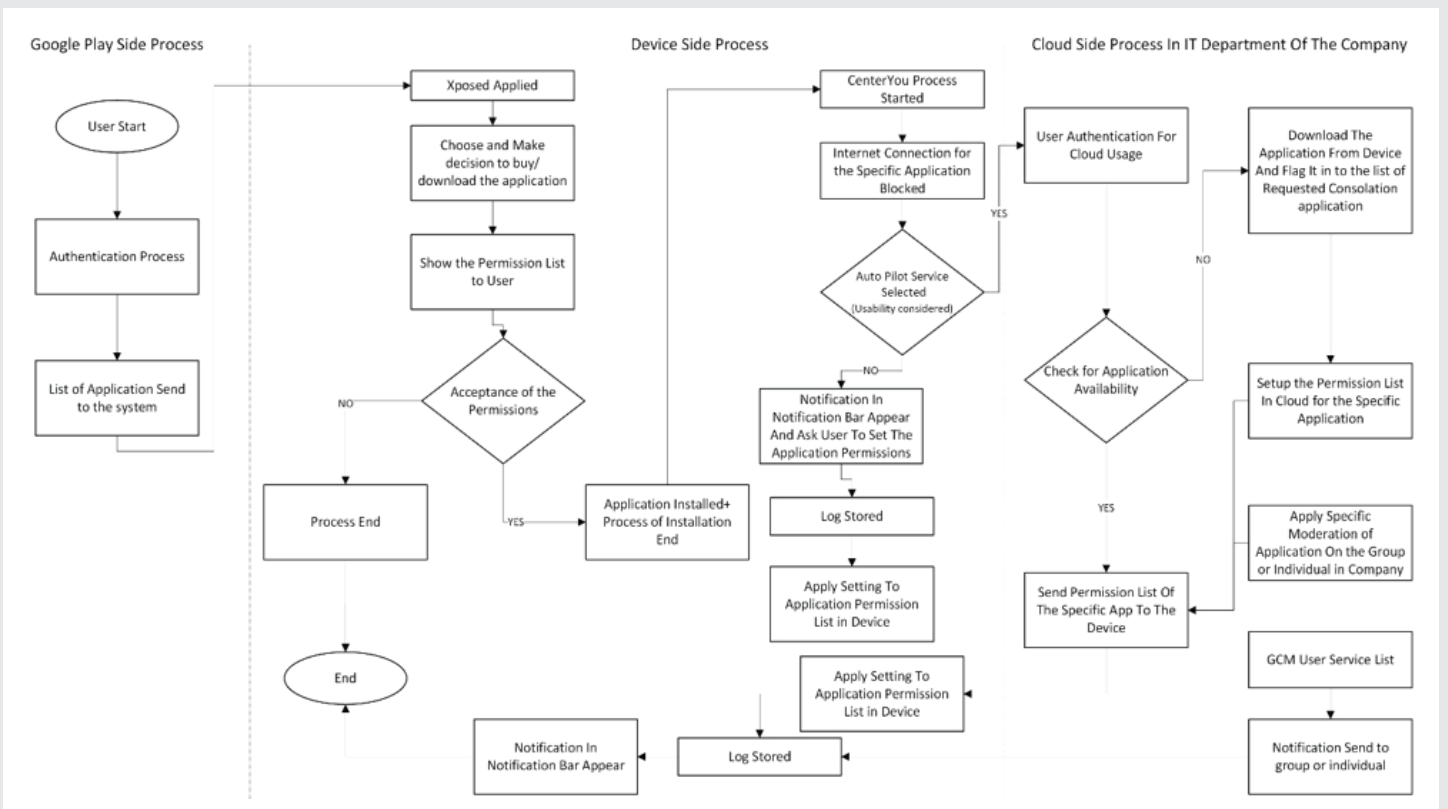


Figure 13: CenterYou Organizational Perspective.



this approach, within the suggested AdDroid platform system, the latest permission authorization advertisement must be stated by the application programmers whenever a particular Smartphone application intends to offer advertisements to the user. Even though these methods are clearly visible as well as useful, they are reliant on marketing organizations' cooperation, which is unlikely.

The types of privacy plug-ins discussed above focused on allowing the users additional management of applications. Thus, their designers implicitly assumed that customers would be able to configure these kinds of configurations properly. However, several user studies have shown that most individuals have limited knowledge and aptitude and are thus unable to fully benefit from this functionality. Moreover, these configurations require the users to be able to identify their own privacy choices, which is often very difficult, given the limited information they have at their disposal.

## Conclusion

The Smartphone industry has expanded significantly over the last few years. According to the available data, each year, a marked increase in the number of devices in use is observed. Most consumers opt for Smartphones due to the extensive number of software applications that can be downloaded on their devices, thus increasing their functionality. However, this growing trend of application installation brings an issue of user protection, as most applications seek permissions to access data on a user's device. The risks this poses to sensitive data are real to both corporate and individual users. While Android has grown in popularity, this trend has not been followed by the efforts to increase the security of its users.

This is a well-known set of problems, and prior solutions have approached it from the ground up; that is, they have focused on implementing reasonable security policies within Android's open-source kernel. While these solutions have achieved the goals of improving Android with such security policies, they are severely hampered by the way in which they have been implemented them. To protect users' personal information from over-privileged apps, a new mode of privacy is needed in Smartphones, whereby the access to users' personal information is controlled either by the user or by an automated process. Furthermore, the user should have run-time control to modify the previously given permission.

The aforementioned policies revert to the static nature of permission assignment, while simultaneously giving the user the power to grant and revoke individual permissions on a per-application basis. For permissions that access information, such as contacts, CenterYou can reliably return pseudo data. It is automated, as it relies on a cloud-based monitoring system, which provides additional advantages to this approach. The main benefit of this solution is allowing the cloud decision-making system to provide information and set up the control unit base.

In this paper, steps for the design and development phase of the CenterYou application were provided [53]. It commenced

by introducing the Android structure for CenterYou, in next the CenterYou framework architecture, and the installation and execution of CenterYou were explained. After discussing their respective architectural characteristics, the criteria for architecture design for CenterYou were described. In addition, the CenterYou device-side performance was elaborated on, with the help of graphs and figures, which helped identify every process flow in the device. Moreover, cloud system and smart decision-making systems with all characteristics specific to individual and organization models, and their databases, have been elaborated on. The characteristic perspective of this architecture has also been discussed, along with the individual and organization design, aimed at different levels of protection.

## Acknowledgement

The CenterYou solution design is registered for intellectual property (File Number: UKM3.2.29/108/2/718) of UKM, Malaysia. We are very thankful to anonymous reviewers for their comments, replies, and suggestions for CenterYou: Android privacy made easier the Cloud Way, which helps and improves future researchers.

## References

- Safavi S, Shukur Z. Conceptual privacy framework for health information on wearable device. *PLoS One*. 2014 Dec 5;9(12):e114306. doi: 10.1371/journal.pone.0114306. PMID: 25478915; PMCID: PMC4257553.
- Airoldi EM, Blei DM, Fienberg SE, Xing EP. Mixed membership stochastic blockmodels. *Advances in Neural Information Processing Systems*. 33-40.
- Ashbrook D, Starner T. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*. 2003; 7(5):275-286.
- Ghosh D, Joshi A, Finin T, Jagtap P. Privacy control in smart phones using semantically rich reasoning and context modeling. *Security and Privacy Workshops (SPW)*. 2012; 82-85.
- Boysen K. "Jawbone with MotionX technology." 2013; <http://content.jawbone.com/static/www/pdf/press-releases/up-press-release-110311.pdf>.
- Wade R, Veneroso F. The Asian crisis: the high debt model versus the Wall Street-Treasury-IMF complex. *New Left Review*: 1998; 3-24.
- Safavi S, Shukur Z. Improving Google glass security and privacy by changing the physical and software structure. *Life Science Journal*. 2014; 11(5).
- Mulliner CR. Security of smart phones, UNIVERSITY OF CALIFORNIA Santa Barbara. Doctoral dissertation. 2006.
- Google. "Android reference developers guide." 2012; <http://developer.android.com/guide/index.html>.
- Google.com. "Android security reference." 2012; <http://source.android.com/tech/security>
- Jesse B. "Android security reference." 2012; <http://www.blackhat.com/presentations/bh-usa-09/BURNS/BHUSA09-Burns-AndroidSurgery-PAPER.pdf>.
- Lunden I. "Tablets are eating into smartphones share of mobile content usage while android remains in lead overall finds jumtapt." 2013; <http://techcrunch.com/2013/03/05/tablets-are-eating-into-smartphones-share-of-mobile-content-usage-while-android-remains-in-lead-overall-finds-jumtapt/>



13. CNET. "Android snags record 81 percent of smartphone market." 2013; [http://news.cnet.com/8301-1035\\_3-57610229-94/android-snags-record-81-percent-of-smartphone-market](http://news.cnet.com/8301-1035_3-57610229-94/android-snags-record-81-percent-of-smartphone-market).
14. Felt AP, Finifter M, Chin E, Hanna S, Wagner DA. survey of mobile malware in the wild. Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. 3-14.
15. Grace MC, Zhou W, Jiang X, Sadeghi AR. Unsafe exposure analysis of mobile in-app advertisements. Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. 101-112.
16. Zhou Y, Wang Z, Zhou W, Jiang X. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. NDSS Journal. 2012.
17. Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution. Security and Privacy (SP). 2012; 95-109.
18. Lineberry A, DLR, Wyatt T. These aren't the permissions you're looking for. 2010.
19. Cai L, Chen H. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. HotSec.
20. Chin E, Felt AP, Greenwood K, Wagner D. Analyzing inter-application communication in Android. Proceedings of the 9th international conference on Mobile systems, applications, and services. 239-252.
21. Xu Z, Bai K, Zhu S. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM. 113-124.
22. Smith C. "Privacy flaw in skype android app exposed." 2012; <http://www.t3.com/news/privacy-flaw-in-skype-android-app-exposed/>.
23. Davi L, Dmitrienko A, Sadeghi AR, Winandy M. Privilege escalation attacks on android. Information Security. 2011; 346-360.
24. Schlegel R, Zhang K, Zhou Xy, Intwala M, Kapadia A, Wang X. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. NDSS. 11: 17-33.
25. Marforio C, Ritzdorf H, Francillon A, Capkun S. Analysis of the communication between colluding applications on modern smartphones. Proceedings of the 28th Annual Computer Security Applications Conference. 51-60.
26. Wei X, Gomez L, Neamtiu I, Faloutsos M. ProfileDroid: multi-layer profiling of android applications. Proceedings of the 18th annual international conference on Mobile computing and networking. 137-148.
27. Mahmood R, Esfahani N, Kacem T, Mirzaei N, Malek S, Stavrou A. A whitebox approach for automated security testing of Android applications on the cloud. Automation of Software Test (AST). 2012; 22-28.
28. Blasing T, Batyuk L, Schmidt AD, Camtepe SA, Albayrak S. An android application sandbox system for suspicious software detection. Malicious and unwanted software (MALWARE). 2010; 55-62.
29. Rasool Saqib. "Blockchain-enabled reliable osmotic computing for cloud of things: applications and challenges." IEEE Internet of Things Magazine 2020; 3.2: 63-67.
30. Lakshmi K, Priya SM, Rama AJK, Thilagam K. Modified AODV Protocol against Blackhole Attacks. in MANET", International Journal of Engineering and Technology. Vol. 2: Citeseer.
31. Safavi S, Shukur Z, Razali R. Reviews on Cybercrime Affecting Portable Devices. Procedia Technology. 2013; 11: 650-657.
32. Raiu C. Cyber-threat evolution: the past year. Computer Fraud & Security. 2012; 3:5-8.
33. Warren SD, Brandeis LD. The right to privacy. Harvard law review: 1890; 193-220.
34. Fischer-Hübner S. IT-security and privacy: design and use of privacy-enhancing security mechanisms: Springer-Verlag. 2001.
35. Federrath H. Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability. Berkeley. CA. USA. 2001.
36. Stanton JM, Stam KR, Mastrangelo P, Jolton J. Analysis of end user security behaviors. Computers & Security. 2005; 24(2): 124-133.
37. Maximilien M, Dimmock B, Streetman D, Weischedel B, Klissner P, Dusankar S, Kleinman R, McKinlay H. "Wincor-Nixdorf, Peter Duellings, Roger Lindsjö, Steve Turner, Paul Gay, et Boris Dainson. Java API for USB (javax. usb), JSR-80 specification. 2001;9.0: <http://javax-usb.org/>.
38. Leslie B, Chubb P, Fitzroy-Dale N, Götz S, Gray C, Macpherson L, Potts D, Shen YT, Elphinstone K, Heiser G. User-level device drivers: Achieved performance. Journal of Computer Science and Technology. 2005; 20(5): 654-664.
39. Safavi S, Shukur Z. Android Privacy Made Easier the Cloud Way. 2020.
40. Services AW. "Amazon Web Services (AWS) and Cloud Computing Services." 2014. <http://aws.amazon.com>.
41. Zhou, Zhenyu, et al. "Cloud miracles: Heterogeneous cloud RAN for fair coexistence of LTE-U and Wi-Fi in ultra dense 5G networks." IEEE Communications Magazine 56.6 (2018): 64-71.
42. Android G. Google Cloud Messaging for Android | Android Developers. 2014.
43. Al-Dulaimi Anwer. "A framework of network connectivity management in multi-clouds infrastructure." IEEE Wireless Communications 2019; 26.3:104-110.
44. Regli WC, Hu X, Atwood M, Sun W. A survey of design rationale systems: approaches, representation, capture and retrieval. Engineering with computers. 2000; 16(3-4): 209-235.
45. Zhang D, Adipat B. Challenges, methodologies, and issues in the usability testing of mobile applications. International Journal of Human-Computer Interaction. 2005; 18(3): 293-308.
46. Kern M, Sametinger J. Permission Tracking in Android. UBICOMM 2012, The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. 148-155.
47. Beresford AR, Rice A, Skehin N, Sohan R. MockDroid: trading privacy for application functionality on smartphones. Proceedings of the 12th Workshop on Mobile Computing Systems and Applications. 49-54.
48. Zhou Y, Zhang X, Jiang X, Freeh VW. Taming information-stealing smartphone applications (on android). Trust and Trustworthy Computing. 2011; 93-107.
49. Hornyack P, Han S, Jung J, Schechter S, Wetherall D. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. Proceedings of the 18th ACM conference on Computer and communications security. 639-652.
50. Nauman M, Khan S, Zhang X. Apex: extending android permission model and enforcement with user-defined runtime constraints. Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. 328-332.





51. Jeon J, Micinski KK, Vaughan JA, Reddy N, Zhu Y, Foster JS, Millstein T. Dr. Android and Mr. Hide: Fine-grained security policies on unmodified Android. Digital Repository at the University of Maryland. 2011.

52. Pearce P, Felt AP, Nunez G, Wagner D. Addroid: Privilege separation for applications and advertisers in android. Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. 71-72.

53. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S. A design science research methodology for information systems research. Journal of management information systems. 2007; 24(3): 45-77.

### Discover a bigger Impact and Visibility of your article publication with Peertechz Publications

#### Highlights

- ❖ Signatory publisher of ORCID
- ❖ Signatory Publisher of DORA (San Francisco Declaration on Research Assessment)
- ❖ Articles archived in worlds' renowned service providers such as Portico, CNKI, AGRIS, TDNet, Base (Bielefeld University Library), CrossRef, Scilit, J-Gate etc.
- ❖ Journals indexed in ICMJE, SHERPA/ROMEO, Google Scholar etc.
- ❖ OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting)
- ❖ Dedicated Editorial Board for every journal
- ❖ Accurate and rapid peer-review process
- ❖ Increased citations of published articles through promotions
- ❖ Reduced timeline for article publication

**Submit your articles and experience a new surge in publication services**  
(<https://www.peertechz.com/submission>).

*Peertechz journals wishes everlasting success in your every endeavours.*