

OWASP Seraphimdroid guide and documentation

By Nikola Milosevic, Furquan Ahmed and Kartik Kohli

Introduction

Android users face many threats and risks. Since modern mobile devices are almost all the time exposed to the internet and other types of mobile networks, they are more exposed to the attacks. From the open WiFi networks that can be spoofed to the Trojan malware applications on the app stores, threats are everywhere around. Many of the attacks are successful because users are not aware of the risks and threats. They may act naive and expose themselves to the attacks even more. These attacks may lead to the identity theft, money theft, losing privacy or they devices may start acting as part of the botnet network.

In order to prevent attacks on the users, this project aims to develop a set of guidelines and application that will ensure that users are using their devices in a secure manner. Project is and always will remain open for everyone to participate and all project deliverables will be free and open source.

Mission of OWASP Seraphimdroid project is to create, as a community, an open platform for education and protection of Android users against privacy and security threats.

On Android, every application operates in a basic sandbox and is prevented from accessing additional services that require users consent. These services can only be accessed if users allows the application to use them. Granting of permission is static and can only be done at the time of the installation of the application. Android security model leaves most of the hard work for security related approach to the user. SeraphimDroid aims to provide detailed explanation and documentation on the permission that android application uses. Some of the permission could cause harm to user's money and data, SeraphimDroid scans the applications and predicts potential malicious behaviour using state of the art Machine Learning algorithm. SeraphimDroid will also evolve to provide a system for blocking access to premium services without user's permission.

An unaware user might keep his device on settings that can open up security vulnerabilities, SeraphimDroid's "Settings Check" feature scans vulnerable security settings and notifies user of the potential harm, and the optimal setting that he should switch to.

Although the android architecture provides a more than solid platform which is secure and at the same time robust but there are few bits that are not included in the default android systems. Android has a layered architecture and the each process runs separately from one another in its own sandbox but this doesn't ensure the protection of the device from all the privacy attacks and clearly not from device theft. SeraphimDroid was developed keeping in mind only the looped aspect of the Android security architecture and was focused on securing user from losing money and giving a documentation of permissions, but it has been evolved to an anti-theft and privacy protecting application.

The Application Locker has been introduced which is a privacy enhancement. As the title suggest it will put a lock over applications which user choose to lock. It will deny access to those application to other people in case the passcode entered is incorrect.

The Service Locker provides protection mechanism for services such as Wi-Fi, Bluetooth and Mobile Data. It will prevent switching on/off these services if the user is unable to enter the correct passcode.

Another such useful enhancement is Geo-Fencing. An anti-theft advancement for the application. It allows the user to create a virtual fencing for the user's device and if the device moves out of the fencing, it is assumed that the device is being stolen and it starts performing operations such as locking the device, wiping data, ringing siren etc. to protect itself. Of course user could control which operation to enable and not to enable. This looks quite complete but in case users forgets to enable fencing he still has to power to perform these operation which could be triggered using a special SMS. The SMS will contain a secret code on receiving which the phone will perform these features and moreover it could send you its current location coordinates.

SeraphimDroid overview

SeraphimDroid takes a heuristic and machine learning approach to find out the potentially malicious or harmful application installed on the user's phone. These are based on the permissions these application uses, but besides that it provides some other security and privacy features as well. All the features that SeraphimDroid provides are:

1. Permission Scanner
2. Settings Checker
3. Call / USSD blocker

4. SMS Interceptor
5. Application Locker
6. Service Locker
7. Geo-Fencing
8. Remote Lock / Wipe

All these features are helpful to the user, and helps him prevent his phone from harmful application, data theft, and unwanted money loss to premium services and device theft. These services are explained briefly.

Permission Scanner

The scanner will go through all the installed application on the user's device and will scan all the permissions each application uses. Then it will fetch the details about each permission and show the application as red (harmful) or green (safe). The Machine Learning algorithm will predict the malicious behaviour of the application based on the respective permissions. The labels predicted are:

1. Green: The application is unlikely to show malicious behaviour
2. Red: The application is likely to harm the user's device, data or both

Settings Checker

The Settings Checker scans the user's device for vulnerable settings and informs him about the potential vulnerabilities that can arise from these. It also gives him a one-click shortcut to go directly to the respective settings page, so he can change it directly.

SeraphimDroid by default performs a daily scan of the settings, and notifies the user via a notification. The user can choose to perform a weekly, fortnight or monthly scan by selecting the respective option in the settings.

Call / USSD Blocker

The blocker is built in the application to block outgoing numbers which is not saved in the user contact list but now it has been evolved to provide the blocking control to user. The user could choose to block calls as he wants from the setting. A blacklist is also implemented which will allow user to block only certain numbers on his will.

On the other hand the USSD blocker is something kept out of the reach of user but blocks all the dangerous USSD that could be entered. USSDs includes code to

factory reset, delete user's data or lock the phone. The USSD blocker prevent these harmful codes from being executed.

SMS Interceptor

SMS Interceptor catches outgoing and incoming messages, scans it and deem the message as being malicious or sent without users notice. Currently, SeraphimDroid only notifies the user about the danger and user have to take action. More advancement will be done in upcoming versions.

Application Locker

This could be considered as more of a privacy feature as user will be able to prevent access to certain applications like gallery, people, etc. to others who might access his phone. This will secure others access to user's content and hence is essentially a privacy enhancement.

User will have the power to lock any application and unlock it. Whenever a locked application is started a password prompt is shown, on entering the correct password only the locked application can be accessed else the application will be terminated.

Service Locker

This is a protection mechanism from unauthorized use of essential services such as Bluetooth, Wi-Fi and Mobile Data. It can save the user from both malicious use, Bluetooth for instance, also cost from services like Mobile Data.

User can lock these services from the application. Whenever the state of these services is changed, either switched on or off, the user will be prompted for a password. If he's unable to provide the correct password, the service will be restored to its original state.

Geo – Fencing

Geo - Fencing is something of a new addition to SeraphimDroid. Enabling Geo-fencing create a virtual fence around the device's current location. If the mobile goes out of the range it starts performing some specific action which users selects while enabling the service. Also, user could enable location updates in case phone got stolen.

Remote Lock / Wipe

If user forgets to turn on the Geo-fencing and phone gets lost this feature can come in handy. This allows user to send a secret code to the user's device which activates the service. The phone then lock the phone, wipe the user's data or send the current location of the device or all of at once.

Implementation details

According to new android navigation guidelines best way to provide navigation in an application is navigation drawer. The drawer uses fragments for each layout that is displayed on the screen. Following the above guideline access to each service is given through the fragments. So like any other application SeraphimDroid starts with a MainActivity.java file which displays the navigation drawers consisting of 8 main fragments, namely,

2, SettingsCheckFragment.java

3. BlockerFragment.java

4. ApplicationLockerFragment.java

5. ServiceLockerFragment.java

6. GeoFencingFragment.java

7. SettingsFragment.java

8. AboutFragment.java

Implementation detail for each fragment is given below.

Permission Scanner

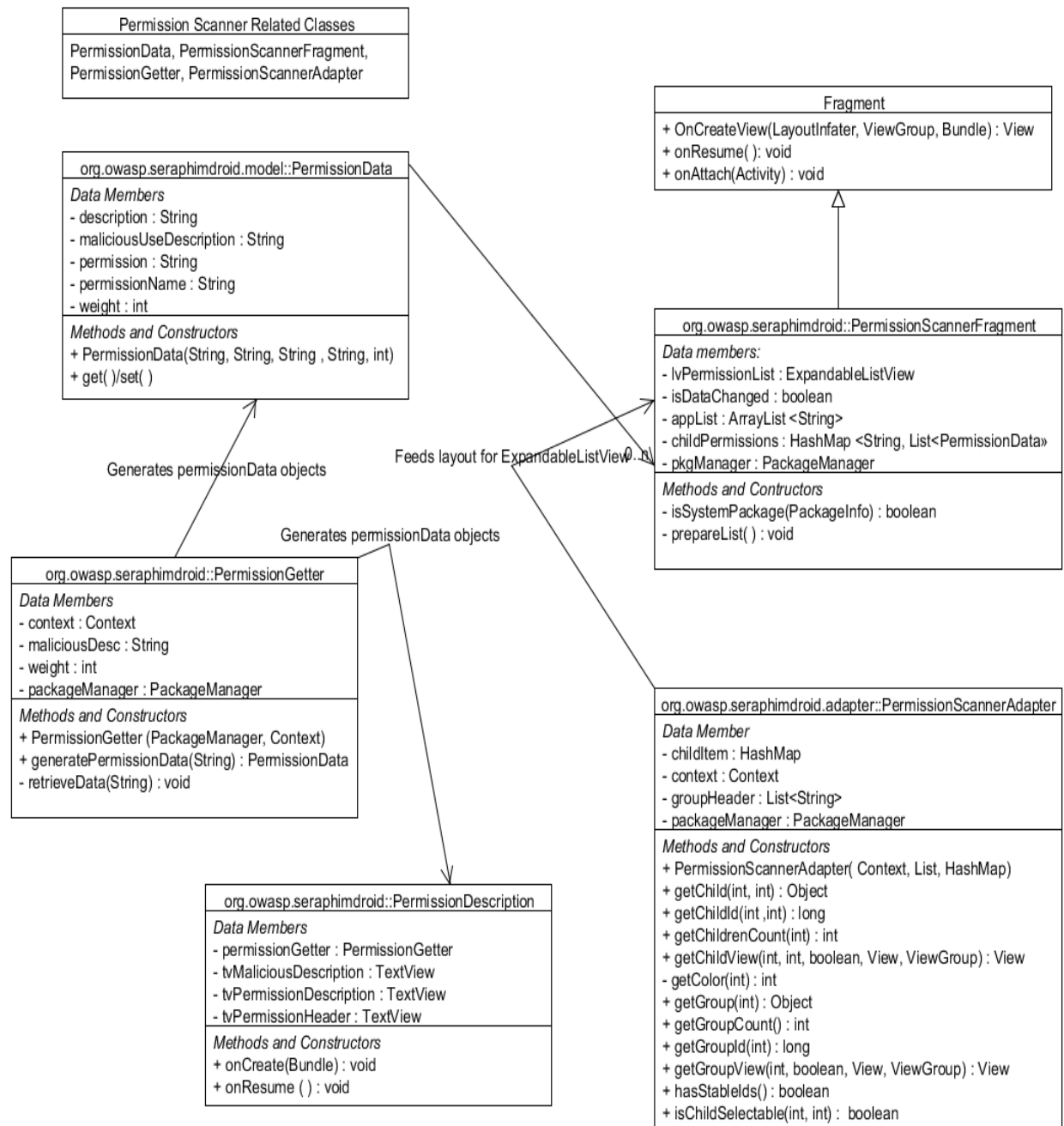
Permission scanner scans permission for all the installed application, for that, it need to get the list of all the installed application. **PackageManager** class provides the required method for that

```
List < ApplicationInfo> appList =  
getActivity().getPackageManager().packageManager.getInstalledA  
pplication(PackageManager.GET_META_DATA);
```

appList stores the information of all the installed application, then for each item in the list permission are fetched and the details for these permissions are retrieved from the database using a custom class **PermissionGetter**. All this task is performed in a separate thread using class **AsyncTask**, which contains an object **svmModel**, which loads a pre-trained **SMO** Weka model. This instance is then used to predict the the nature of application, using it's permissions as a 0/1 feature vector. The Prediction accuracy of our current model is ~88%

ExpandableListView is used to display the installed applications and the child item for each item displays the permissions that is used by that application. With each installed application an indication is displayed to show the danger level for that app and user could uninstall the application by long pressing on the name.

PermissionDescription displays the details about each permission, the threat it poses, what access it provides to the application and how can it be used to damage your data or affect your privacy. It has been themed as a dialog box, because that suits best for displaying some information.



Settings Checker

Settings Checker scans the Device's settings, using the **Settings.Secure** API provided by Android. It can be used to fetch settings for different parameters, such as Settings.Secure.**ADB_ENABLED** to check if USB Debugging is enabled, similarly

Settings.Secure.**INSTALL_NON_MARKET_APPS** to check if Application install from unknown sources is permitted.

Settings Checker also has a background service, implemented using Android's **Alarm Manager** class, which performs automated checks at regular time intervals. By default, the interval is set to 1 Day. It can be changed in the settings to a Week, Fortnight or a Month.

If the service scans and finds that any of the setting is not set to its optimal value, it fires a notification. Upon clicking this notification, the user is taken to the Settings Checker fragment, wherein he can fix the respective settings.

Blocker logs

The UI is a tabbed view for showing the logs for each Call, SMS and USSD that could be malicious. The tabbed layout is created using **TabHost** and **ViewPager** together in combination to achieve the required navigation flow and good UI design.

The broadcast receivers are used to keep check on the malicious activities, for example unaware message sending, premium calls, execution of harmful USSDs etc. These Broadcast receiver includes

1. CallRecepter
2. SMSRecepter

The **CallRecepter** class is used to intercept incoming and outgoing calls. Every call that is placed by any application is passed through this receiver and then it checks if the called number meets the requirement set by the user i.e. if the number is saved in users contact details or is not present in the user's blacklist. User could access the preferences regarding call blocker in the settings. Any call not meeting the requirement is logged in the Call logs and is shown in the blocker logs, with the reason for the blocking.

The **SMSRecepter** class does pretty much the same but with messages. The receiver can only handle received messages, for outgoing messages another service is used because the Android mechanism for outgoing SMSs are different. In case of calls the calls could be blocked or cancelled but for messages its different and specially with outgoing messages. There is no way yet known which could be used to alter the content of outgoing messages, or better block them. The Incoming SMS are deemed malicious and is reported to the user, if the message content contains numbers that are not saved in the contact list of user.

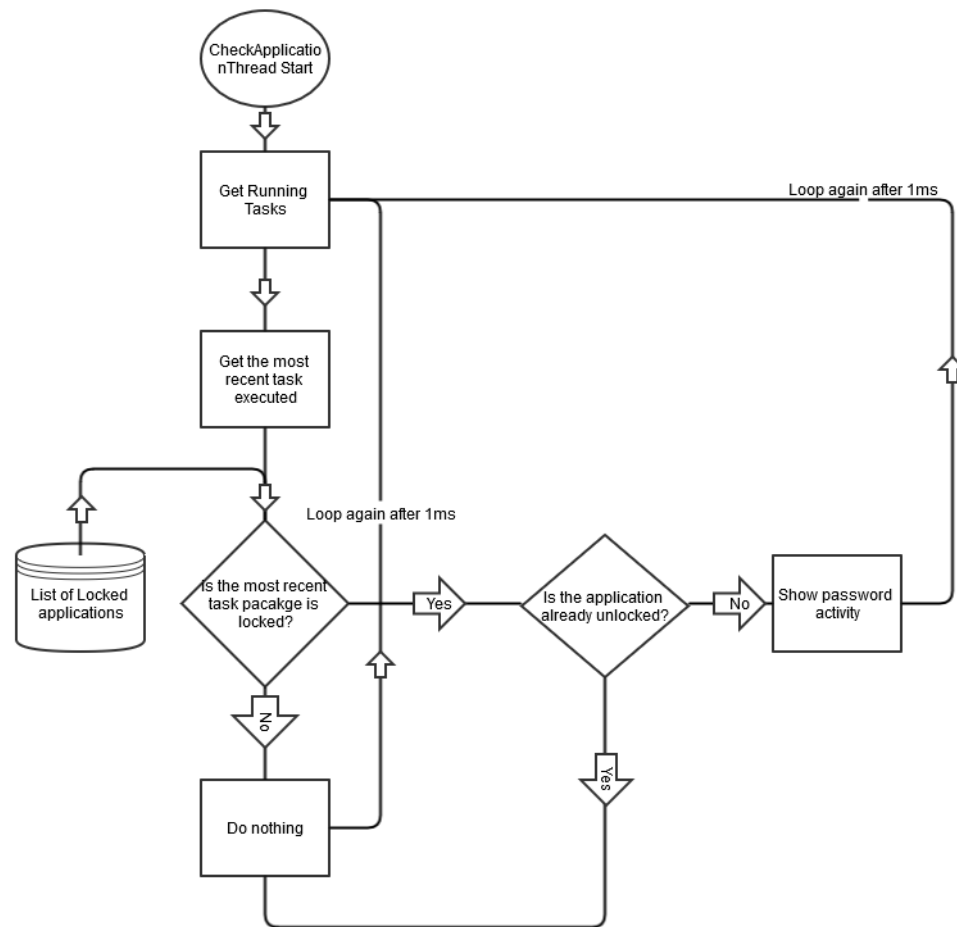
OutgoingSMSRecepter is the class which checks if the outgoing SMS is sent by the user or if the message is sent by some other application. If latter is true a notification is generated giving the application name which sent the message, also the content of the message it sent. More control over SMS can be expected in further development.

Application Locker

Application locker does just what the title suggests. The locker interface is simple and it displays the list of all the installed application. With each application there is toggle button which lets user to lock or unlock the particular application. The mechanism is simple as is, a database table is created which store the package name of all the locked application and is empty if no application is locked.

The heart and soul of application locker is the background service that keeps track of all the application that is launched. This service is sticky and need to be run always in the background. Android provides API to get the list of recent running task on the device, the **AppLockService** calls **getRunningTasks()** method of **ActivityManager** class object. The list is arrange with the top element being the most recently executed. Using the top most element of the list it is determined whether the application launched is locked or not. If locked the password activity is launched.

The flow chart for the process is shown in the diagram.



The password activity is implemented as given below.

Password Activity

The password prompt is implemented using a separate activity and is displayed each SeraphimDroid is started or a locked app is started. The password is stored in the database in the form of byte array, which is actually the SHA256 hashed passcode string. Java provides the required method to hash the string

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
```

```
hash = digest.digest(passwordConfirm.getBytes("UTF-8"));
```

hash is then stored in the Database.

Whenever the user enters the password it is validated by getting the byte array from the database and comparing it with the string hash of the passcode entered by the user. If they match password is confirmed and access is granted.

Services Locker

Services Locker's implementation involves a listener for each of the services. For ex. `Wi-FiStateReceiver` for Wi-Fi. This receiver listens to change in state of the Device's Wi-Fi. If there is a change (User switches on/off the Wi-Fi), SeraphimDroid password prompt is shown, and the service is returned to it's previous state.

For ex, if the user switches ON Wi-Fi, Password prompt will be shown, and Wi-Fi will be disabled by the command:-

```
Wi-FiManager.setWifiEnabled(false);
```

Switching ON/OFF these services require special permissions -

Wi-Fi - android.permission.**CHANGE_WIFI_STATE**

Bluetooth - android.permission.**BLUETOOTH_ADMIN**

Mobile Data - android.permission.**CHANGE_NETWORK_STATE**

Geo-Fencing

Geo-fencing is the most resource intensive service, it requires you to provide Device Administrator Privilege to the application, which controls the device lock, and wiping of data. Moreover, it asks you to enable GPS for better location tracking, so GPS is must too. Once these features are enabled only then you could access the feature of this service to the full extent.

The class that does most of the work is **GPSTracker** class, this class gets the current device location from the GPS or network provider whichever is available. Using the location from this class the GeoFencing service keeps track of the device, if it is still inside the virtual fence or moved out of it. The center is set to the location at the time the user starts the service, which acts the center of the fence. After the service is started the location is queried every 10 seconds. That's not too much battery draining but also not losing the timely location tracking that's needed.

The other important class is **GeoFencingService**, this class takes care of all the actions that device must perform in case the device is assumed stolen. The device is said to be out of the range by calculating the distance from the center

which is the location coordinates used at the time service was started and the current device location coordinates. The API provides the method to do so which is **Location1.distanceTo(Location2)**. This class keeps track of the device location and in case the device is found to be out of the fence, it checks the location 3 more times and when it confirms the phone is out of the fence, it activates the alarming sound using **AlarmManager**. Similarly, current location is sent to a secure number set by user enables the feature.

As this service requires a working GPS, a prompt is made if GPS is unavailable and unless user enables the GPS the service is inaccessible. A secure number is required to enable the service to get location and Device Administrator privilege is also must.

Remote Lock / Wipe

The Geo-fencing requires user to enable the service first but if the user forgets to enable the service he could still lock or wipe data of his phone using remote wipe. Remote wipe uses the same locking mechanism of **DevicePolicyManager**. It uses the same broadcast receiver created for received SMSs. The receiver checks the message for the secret code set by the user, if the message contains the secret code, it activates the remote services and locks the device and if enabled wipes the user data from the device.

The main methods responsible for performing the remote actions are:

```
devicePolicyManager.wipeData (0);  
  
devicePolicyManager.lockNow ();
```

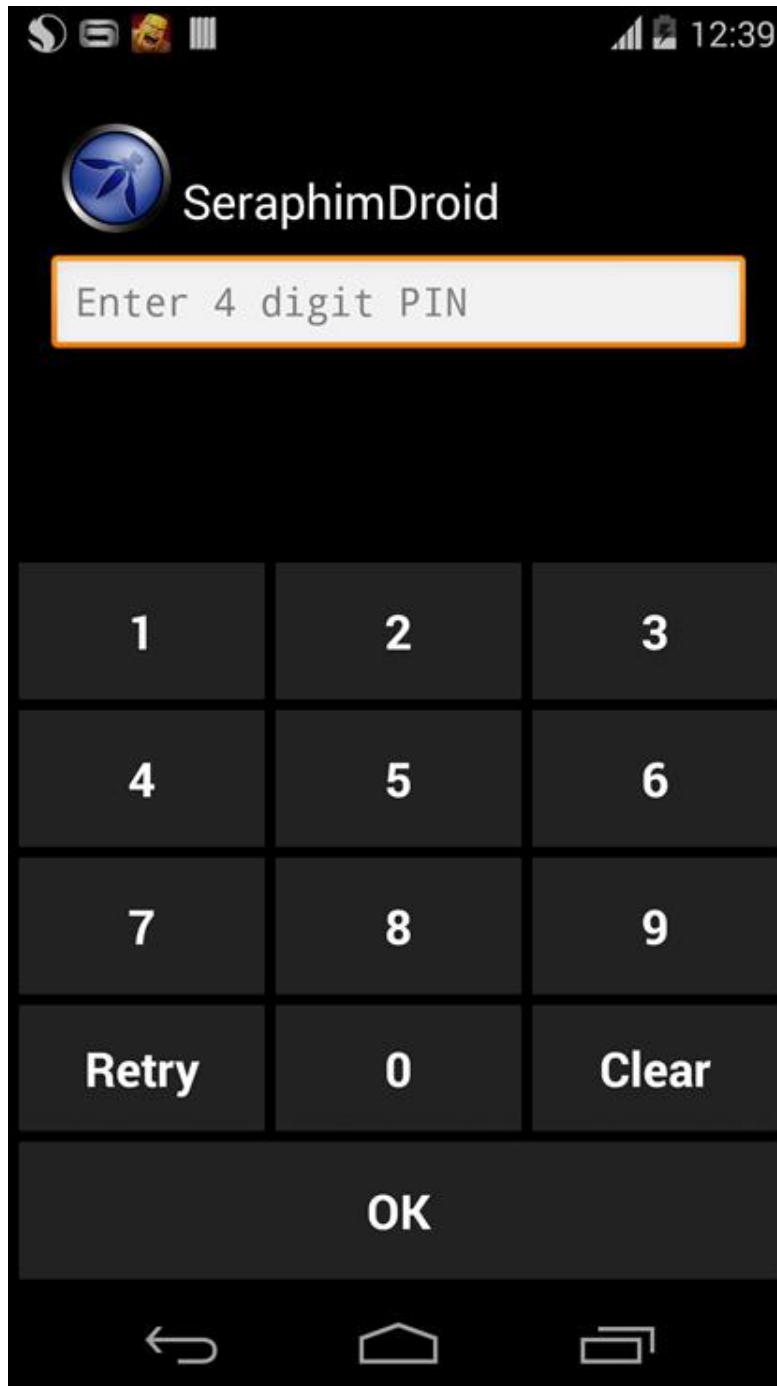
For sending current location, current location is fetched using **GPSTracker**, and using **SMSManager** API a SMS is sent to the secure number that user believes could be reached in case the user's device is stolen. The secret code is stored in application preferences which is retrieved each time a new message is received to check if the message contains the secret code.

Usage details

(Here you can write some details on how it is used. Some kind of short user guide. You may add some screenshots as well.)

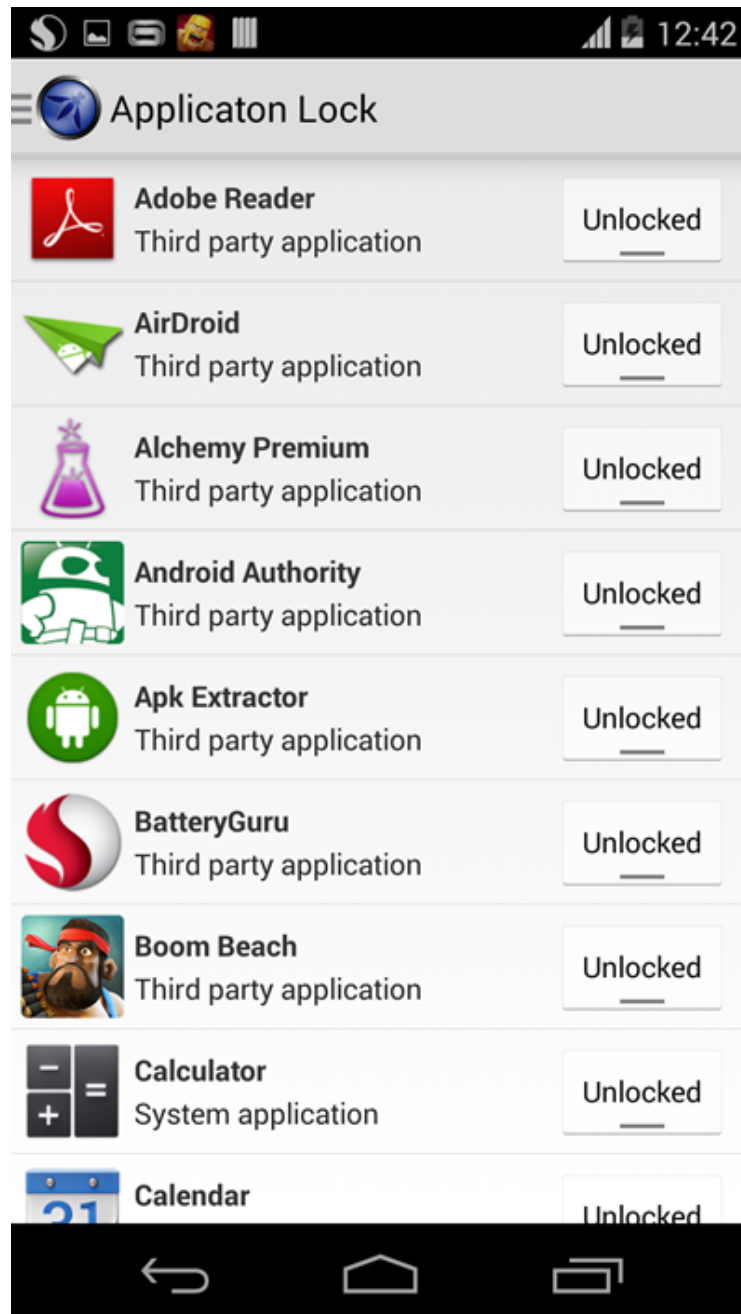
First Start

Before SeraphimDroid can actually be accessed, user needs to create a PIN code to lock the application. The same code will be used to unlock locked applications. The process to create a PIN is really simple. When SeraphimDroid is started for the first time a prompt is displayed to create the PIN which looks like the image below



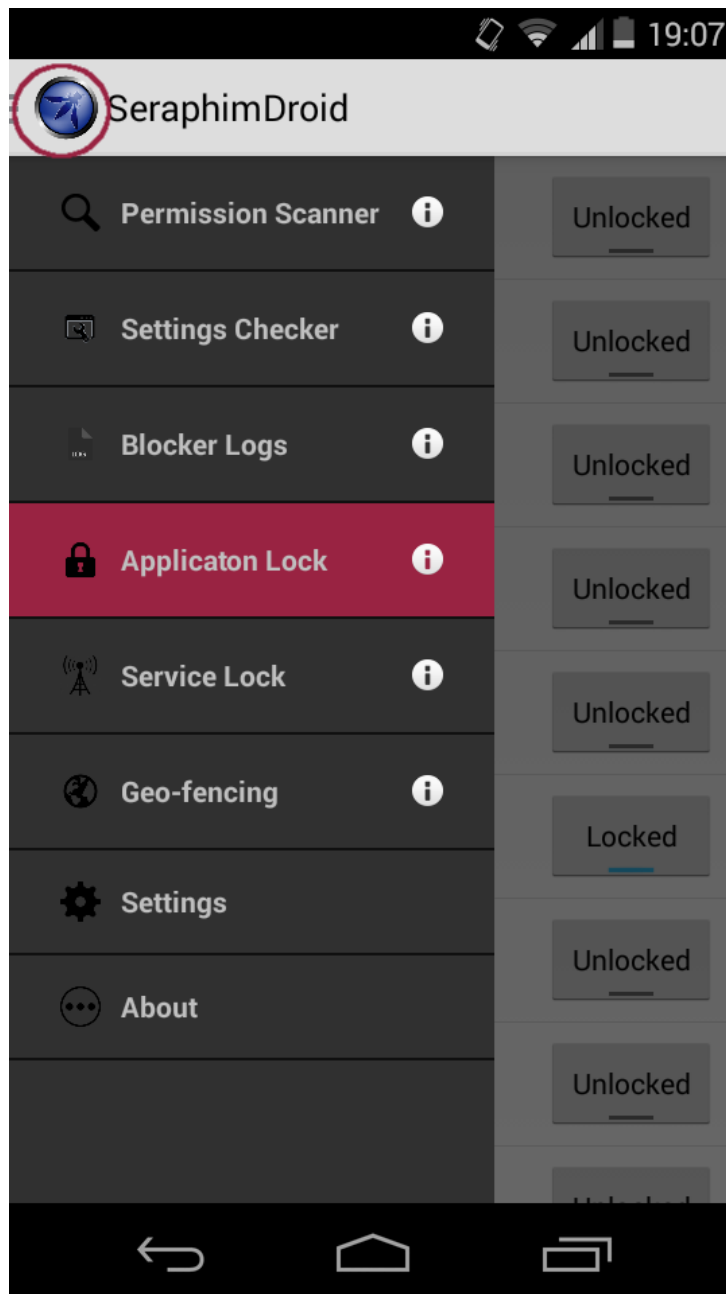
The PIN needs to be entered twice to make sure the user enters the correct PIN. Once the PIN is created user will be asked to enter it every time SeraphimDroid is launched. ***The PIN needs to be at least 4 digits long.***

After the access is granted for SeraphimDroid, you will see the Application locker, with the list off all the installed application that could be locked, just like in the image



Navigation

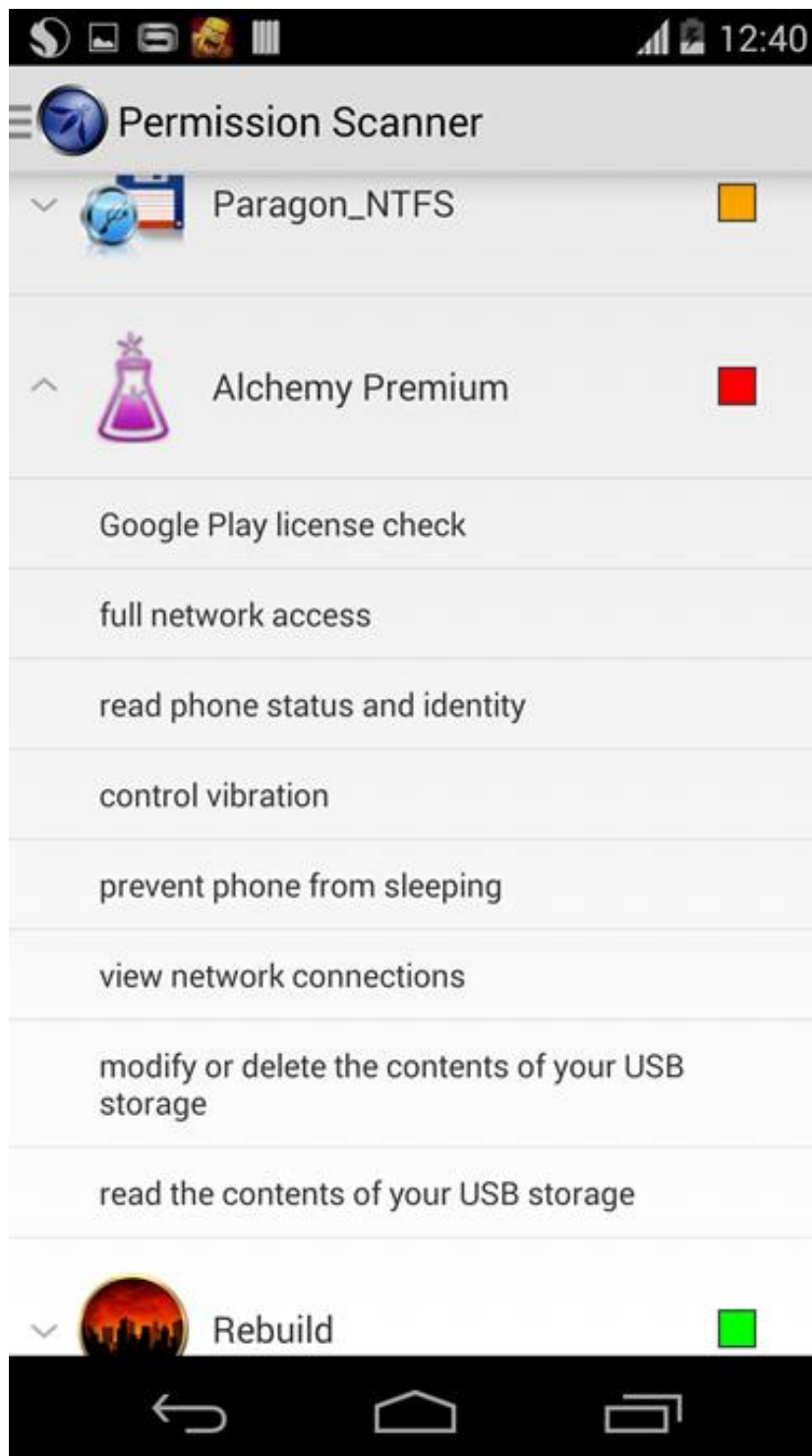
SeraphimDroid is pretty easy to use, all the features can be accessed using the navigation drawer. The navigation drawer can be access either by sliding from left or by tapping on the app icon in the action bar as shown in the image below.



In the navigation drawer you can see all the services that SeraphimDroid provides. Use of each of the service is provided below. In the application, user can see a brief introduction for each service by tapping on the info icon.

Permission Scanner

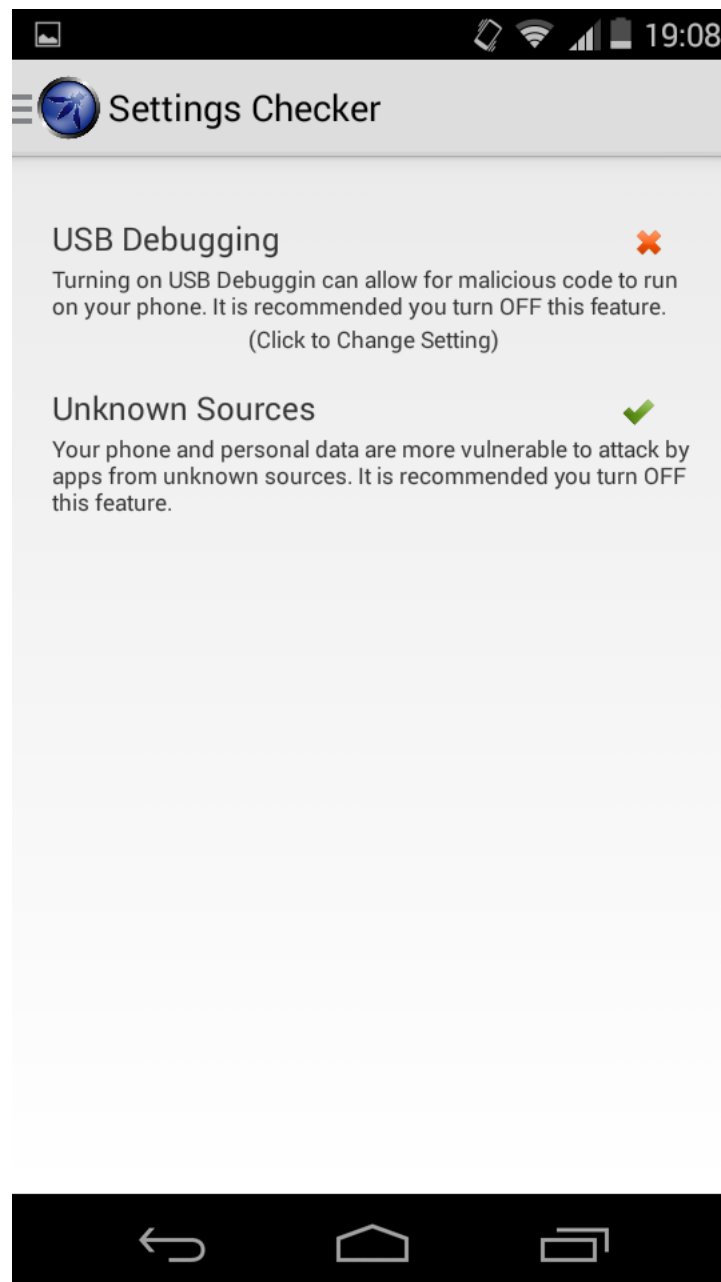
Permission Scanner is the first service that SeraphimDroid provides. It scans all the installed application and display its danger level using colour codes, as mentioned in above section. Expanding any application user can see what permission that application uses. A description about the permissions intended use and malicious use is also provides within the permission scanner. User can see the description by tapping on the permission and a dialog will be displayed showing the details about that particular permission, which includes the general use for that permission and malicious use as well. User can also uninstall the particular application just by long pressing the application name in the list. The permission scanner looks like the image shown below.



Settings Checker

The next service in the drawer is the Settings Checker. This displays the device's settings, which are prone to malicious activity if not set to its recommended state. The one which are not set to its optimal state, a red cross is shown, and also a "click to go to settings" option which will take the user to the respective settings screen where he can change the setting. The options which are set to the

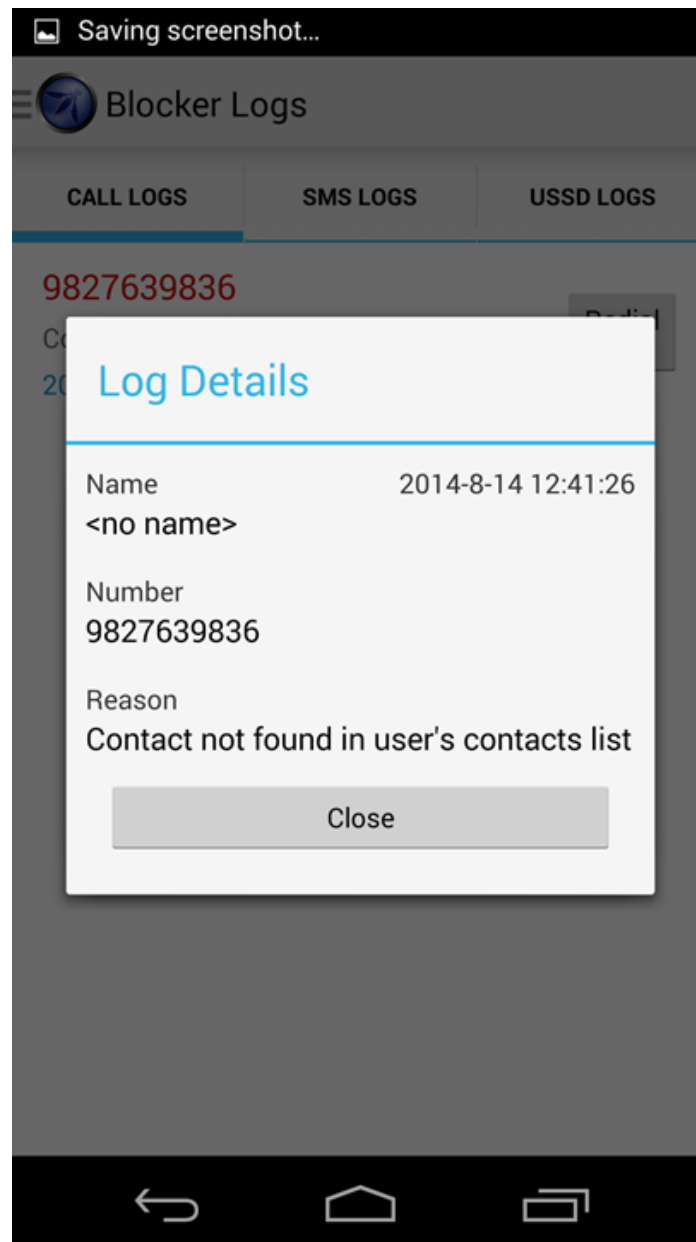
recommended setting, a green tick is shown. The screenshot below shows the Settings Checker Fragment.



Blocker Log

The next service in the drawer is Call blocker and SMS Interceptor. The Blocker log displays all the logs for blocked calls, malicious SMSs and harmful USSDs. The log displays useful information for the user about the blocked call (for example time of call, number called etc.). Tapping the log item shows more details about that particular log. User can swipe between tabs or tap the name of the tab to view details

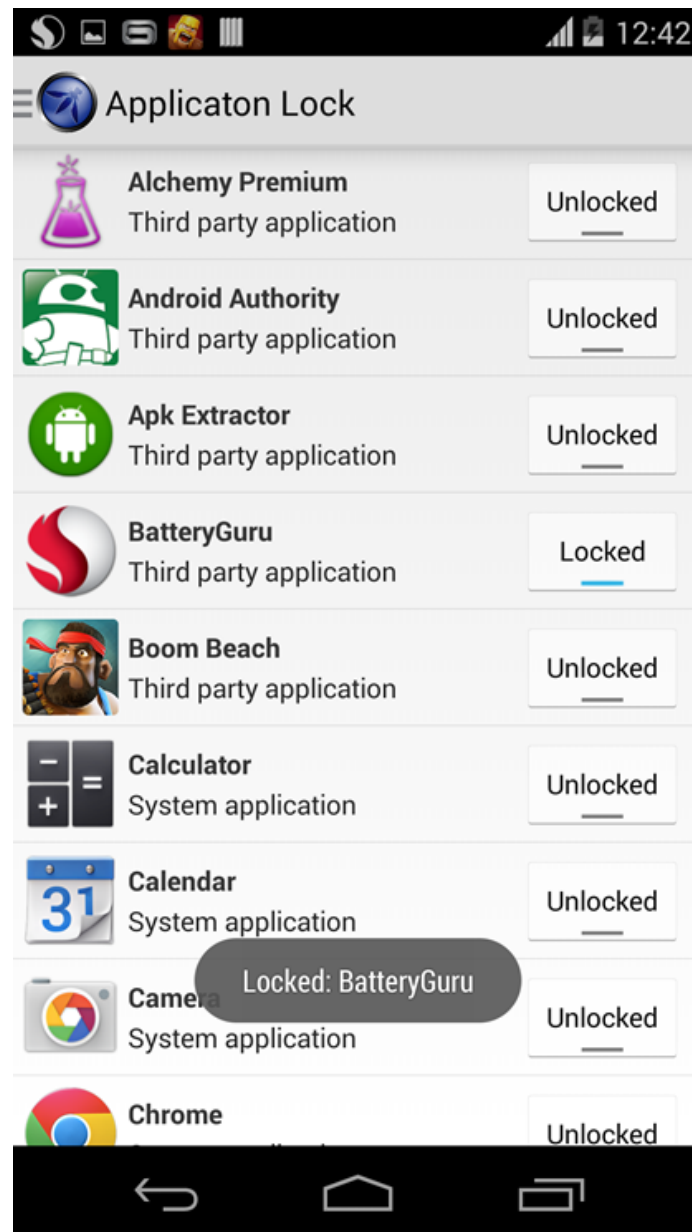
about other logs (SMS Log and USSD log). The navigation kept as simple and as user friendly as possible. A sample for the log is shown in the screenshot



Application Lock

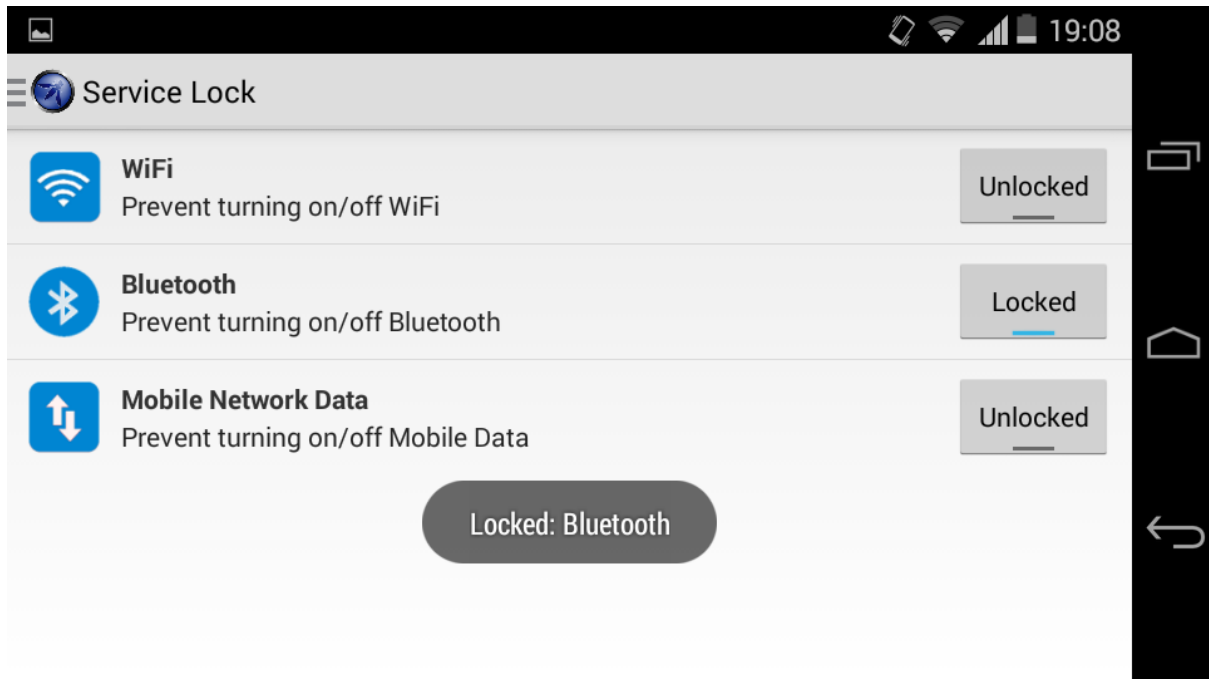
Going down the list of service the next on is Application Locker. Quite easy to understand the use and even more easy to use. All user needs to do is lock the application he wants to protect and the work is done. Any time the locked application is launched, the password prompt is displayed on the screen, which prevents the access to the locked app. Only on providing the correct PIN code can the application

be accessed. The interface is pretty much simple and is shown below. The screenshot also displays one of the application being locked.



Service Lock

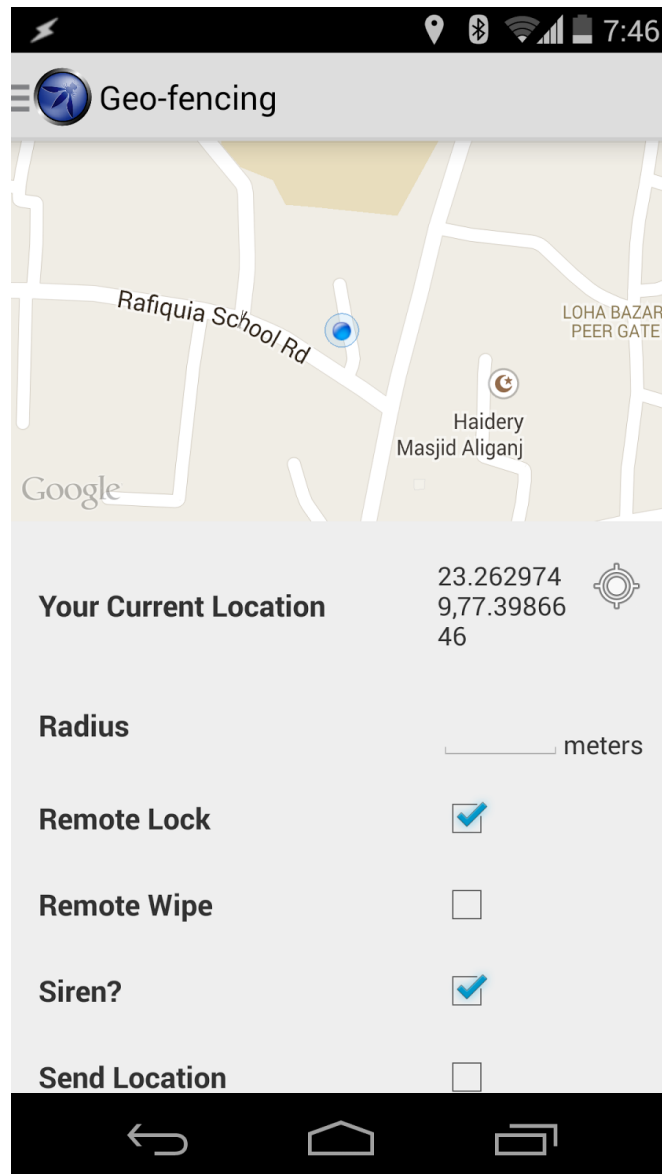
Much like the Application lock is used to lock target applications, The Service Lock is used to lock essential Android services, such as Wi-Fi, Bluetooth & Mobile Data. Whenever a user tries to change the state of any of the above locked services, SeraphimDroid will prompt for PIN, and only upon successfully entering the PIN shall the state be changed. Otherwise, the service will be restores to the previous state. Below is a screenshot of the Service Lock Fragment.



Geo-Fencing

Geo-fencing contains a Google Maps to show user his current location. Then there is a button to get his current location for the service to start. Then follows the range for the perimeter, it can't be less than 200m because the precision of GPS is not so accurate. After that there are four check boxes, one for enabling each feature i.e. lock phone, wipe data, siren and send location.

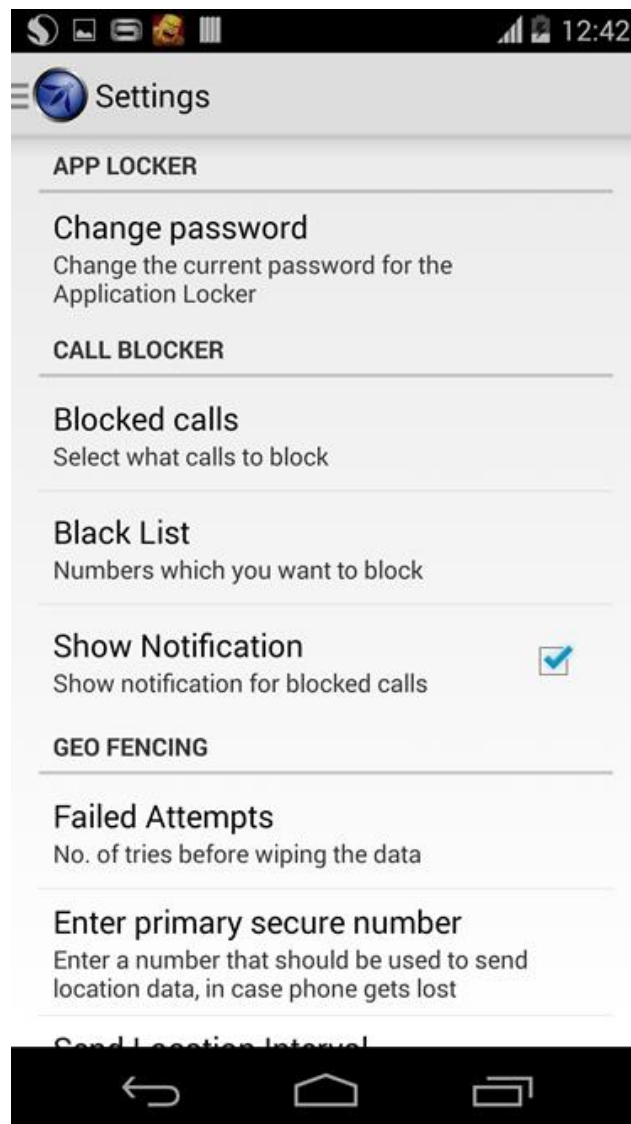
The service won't start unless the GPS is on so in case user does not turn the GPS of the device on a prompt is shown, and if user cancels that prompt, then the button for starting the Geo-fencing is changed to display GPS prompt. Only when the GPS is turned on can the service be enabled. This provides better GPS tracking and location accuracy. The screenshot for Geo-fencing is the below image.



Settings

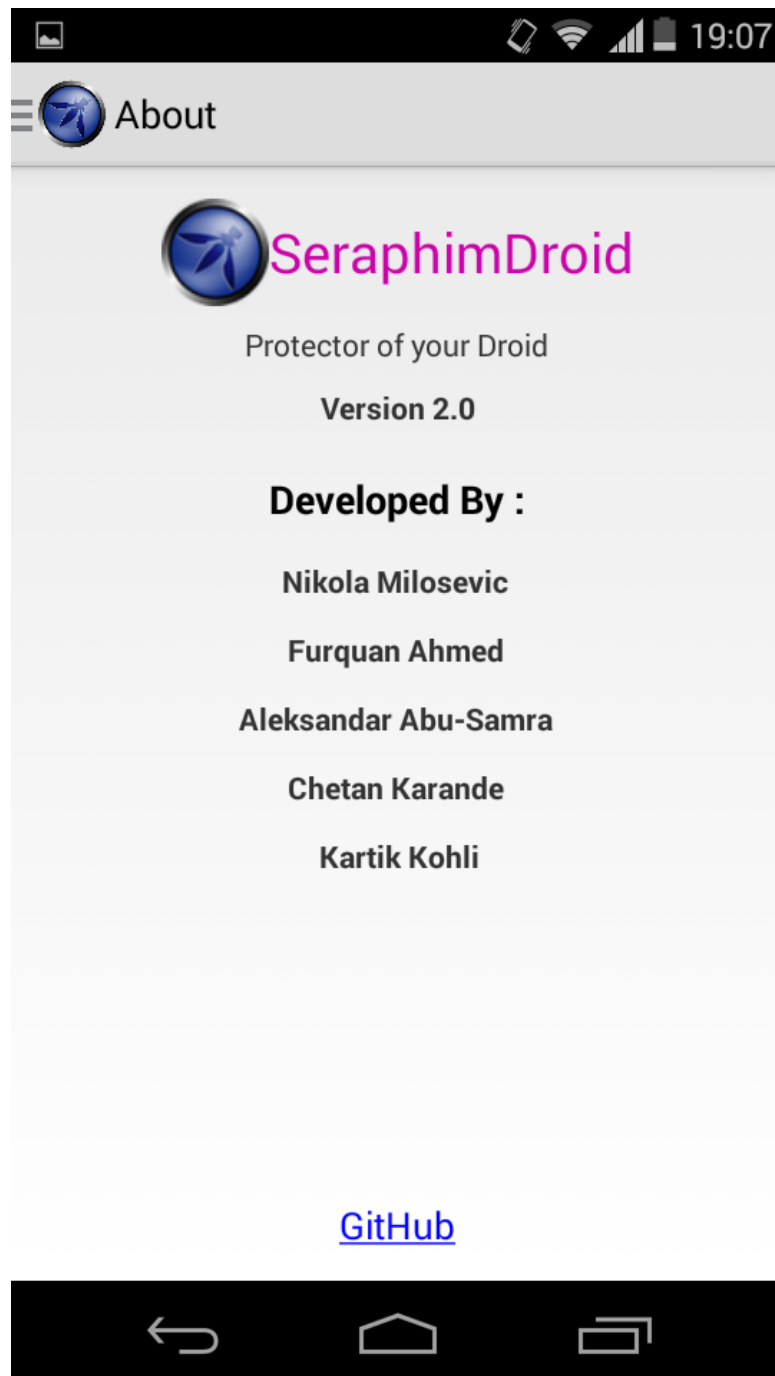
The settings is next in the drawer menu, this is the place for setting preferences for the SeraphimDroid. Here user change the PIN code of SeraphimDroid, he can set the category of calls that should be blocked and add numbers to blacklist to block those particular numbers. The settings page also contains regarding the geo-fencing, user can set secure number, the number of times after locking the data should be wiped and the interval with which the device should send its current location.

The remote service preferences are also accessed from the setting fragment. These include remote lock, remote wipe, remote location and remote secret code. Remote Secret code is set by the user to trigger the remote services. These services are disabled by default, user needs to enable these services before the secret code trigger could work. A snapshot for the settings fragment is attached.



About

The about fragment contains the more information about the project. It has the link to the project page and the code on the GitHub. Also it shows the names of the developer who contributed to the project along with the name of the owner of the project. The version information is also displayed in the same fragment.



Conclusion

SeraphimDroid helps the user to understand the android security architecture more by letting him to the insights of the access modes used by application to request some services. These services are only allowed if user allows them which makes user responsible for making choices before installing applications. This might be a good practice but a general user doesn't actually seem to know about all the details, SeraphimDroid provides this details.

Along with the documentation SeraphimDroid also provides some security and privacy features which helps user secure his phone from theft and also from malwares. The best part it does is alert the user about the money costing apps and saving users money.

Overall the application is functional and is useable but it's a software and there is always possibility for it to be improved. In its latest update, SeraphimDroid uses state of the art Machine Learning algorithm to classify apps into "Malware" or "Goodware". An online database can also be created with the list of all the harmful application names which could cause user harm, this list will be created by people reporting the malware apps.

Also the SMS detection could be improved and in current version the SMS is rated harmful if it contains unsaved phone numbers which could be improved again by having a database containing the details about the malicious SMS.

Furthermore, a widget could be created to enable or disable the Geo-fencing with the touch of the button. This will make it really easy to enable the anti-theft feature of the phone. That's all.