CHAPTER - 1
INTRODUCTION

1.1 ABOUT THE DOMAIN

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touch screen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

1.2 INTERFACE

Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard. Game controllers and full-size physical keyboards are supported via Bluetooth or USB. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide hap tic feedback to the user. Internal hardware, such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented, or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering.

Android devices boot to the home screen, the primary navigation and information "hub" on Android devices that is analogous to the desktop found on personal computers. (Android also runs on regular personal computers, as described below). Android home screens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto-updating content, such as the weather forecast, the user's email inbox, or a news ticker directly on the home screen. A home screen may be made up of several pages, between which the user can swipe back and forth, though Android's home screen interface is heavily customizable, allowing users to adjust the look and feel of the devices to their tastes. Third-party apps available on Google Play and other app stores can extensively re-theme the home screen, and even mimic the look of other operating systems, such as Windows Phone. Most manufacturers, and some wireless carriers, customize the look and feel of their Android devices to differentiate themselves from their competitors. Applications that handle interactions with the home screen are called "launchers" because they, among other purposes, launch the applications installed on a device.

Along the top of the screen is a status bar, showing information about the device and its connectivity. This status bar can be "pulled" down to reveal a notification screen where apps display important information or updates, such as a newly received email or SMS text, in a way that does not immediately interrupt or inconvenience the user. Notifications are persistent until read by tapping it, which opens the relevant app, or dismissed by sliding it off the screen. Beginning on Android 4.1, "expanded notifications" can display expanded details or additional functionality; for instance, a music player can display playback controls, and a "missed call" notification provides buttons for calling back or sending the caller an SMS message.
Android provides the ability to run applications that change the default launcher, and hence the appearance and externally visible behavior of Android. These appearance changes include a multi-page dock or no dock, and many more changes to fundamental features of the user interface.

1.3 APPLICATION

Applications (apps), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language that has complete access to the Android APIs. Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ and C programming language is supported since its version 1.4, which can also be used exclusively although with a restricted set of Androids APIs. The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials.

Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plug-in; in December 2014, Google released Android Studio, based on Intelligent IDEA, as its primary IDE for Android application development. Other development tools are available, including a native development kit (NDK) for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks. In January 2014, Google unveiled an framework based on Apache Cordova for porting Chrome HTML 5 web applications to Android, wrapped in a native application shell.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers; as of July 2013, there are more than one million applications available for Android in Play Store. As of July 2013, 50 billion applications have been installed. Some carriers offer direct carrier billing for Google Play application purchases, where the cost of the application is added to the user's monthly bill.

1.4 MEMORY MANAGEMENT

Since Android devices are usually battery-powered, Android is designed to manage processes to keep power consumption at a minimum. When an application is not in use the system suspends its operation so that, while available for immediate use rather than closed, it does not use battery power or CPU resources.

1.5 ABOUT THE PROJECT

Android manages the applications stored in memory automatically: when memory is low, the system will begin invisibly and automatically closing inactive processes, starting with those that have been inactive for longest. Life hacker reported in 2011 that third-party task killers were doing more harm than good.

The charging system converts the engine’s mechanical energy into electrical energy. This electrical energy is used to maintain the battery’s state of charge and to operate the loads of the automotive electrical system. In this chapter we will use the conventional theory of current: Electrons move from positive to negative. During cranking, all electrical energy for the vehicle is supplied by the battery. After the engine is running, the charging system must produce enough electrical energy to recharge the battery and to supply the demands of other loads in the electrical system. If the starting system is in poor condition and draws too much current, or if the charging system cannot recharge the battery and supply the additional loads, more energy must be drawn from the battery for short periods of time.

Mobile battery charger circuit is a device that can automatically recharge a mobile phone’s battery when the power in it gets low. Nowadays mobile phones have become an integral part of everyone’s life and hence require frequent charging of battery owing to longer duration usage. Battery chargers come as simple, trickle, timer based, intelligent, universal battery charger-analyzers, fast, pulse, inductive, USB based, solar chargers and motion powered chargers. These battery chargers also vary depending on the applications like mobile phone charger, battery charger for vehicles, electric vehicle batteries chargers and charge stations. Charging methods are classified into two categories: fast charge method and slow charge method. Fast charge is a system used to recharge a battery in about two hours or less than this, and the slow charge is a system used to recharge a battery throughout the night. Slow charging is advantageous as it does not require any charge detection circuit. Moreover, it is cheap as well. The only drawback of this charging system is that it takes maximum time to recharge a battery.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM

The aim of the project is to create an application which consumes less battery power when the mobile device is turned off especially services like Bluetooth, WIFI and GPS. Our project is used to manage the battery level in mobile.

2.2 SCOPE

- Existing System analyzes sensory data, once received by an application, would be transformed into various forms and used by different application components.
- Using existing method difficult scheduling event handlers in android programming.
- In previous method is only monitor the charging level and it intimate the alert message for the user.
- In this project, we proposed a two-layered permission-based detection scheme for detecting malicious Android applications.
- Proposed monitor application for execution and perform dynamic data flow analysis at a byte code instruction level.
- Implement Java Path Finder (JPF) which is an explicit-state model checker that directly works with Java byte code instructions.
- Improved Green droid tool finds application execution while sharing application at time of blue tooth sharing.
- In this application is used to monitor the charging status of the mobile and it disconnect the charger when the user accesses the application such as calls and music player during the mobile charging.
- . When the battery percentage gets (goes) lower than, the threshold value, the screen becomes weak and automatically the services like Blue-tooth, Wi-Fi, GPS get turn off.

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODE;

ALGORITHM USED

3.1 ALGORITHM

An algorithm is a step by step method of song a problem. It is commonly used for data processing, calculation and another related computer and mathematical operations.

Technically, computers use algorithms to list the detailed instructions for carrying out an operation. For example, to compute an employee's pay check, the computer uses an algorithm. To accomplish this task, appropriate data must be entered into the system. In terms of efficiency, various algorithms are able to accomplish operations or problem solving easily and quickly.

STEP 1: Open android studio.
STEP 2: Select the option START A NEW ANDROID STUDIO PROJECT.
STEP 3: Select the Activity (empty activity).
STEP 4: Configure your project (name, package name, save location, language and minimum api level).
STEP 5: Select finish.
STEP 6: Open the file **mainactivity.java**
STEP 7: Type the java programming code.
STEP 8: Open **activity_main.xml**
STEP 9: Type the xml code.
STEP 10: Save the program
STEP 11: Run the application using android virtual device.
STEP 12: Export the application file.
STEP 13: Copy the .apk file from the bin folder and run the application in the mobile.
STEP 14: Open the **greendroid** application.
STEP 15: Select the services which you want to disable. Turn off the device

3.2 HARDWARE REQUIREMENT

Processor	:	Dual core processor 2.6.0 GHZ
RAM	:	1GB
Hard disk	:	160 GB
Monitor	:	15-inch color monitor.

3.3 SOFTWARE REQUIREMENT

Operating System	:	Android OS
Front End	:	ANDROID SDK (JAVA)
IDE	:	Eclipse
Back End	:	SQLite

3.4 INTRODUCTION TO ANDROID

3.4.1 HISTORY

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touch screen mobile devices such as smart phones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

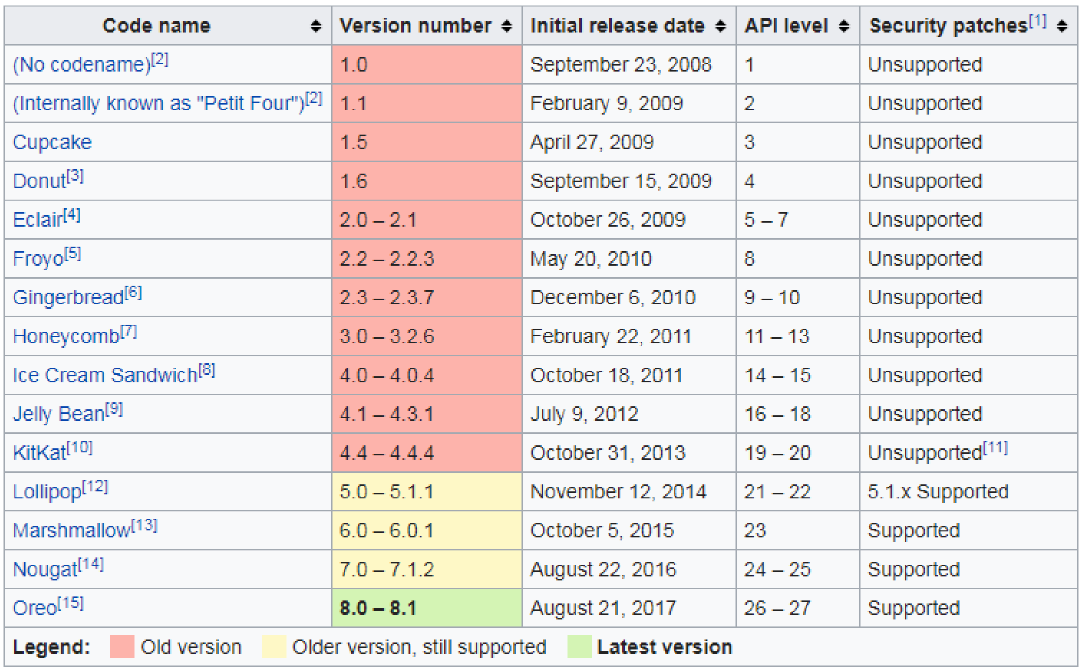
Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 8.1 "Oreo", released in December 2017.

Android has been the best-selling OS worldwide on smart phones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of 2017, the Google Play store features over 3.5 million apps.

The early intentions of the company were to develop an advanced operating system for digital cameras, and this was the basis of its pitch to investors in April 2004. The company then decided that the market for cameras was not large enough for its goals, and by five months later it had diverted its efforts and was pitching Android as a handset operating system that would rival Symbian and Microsoft Windows Mobile.

Since 2008, Android has seen numerous updates which have incrementally improved the operating system, adding new features and fixing bugs in previous releases. Each major release is named in alphabetical order after a dessert or sugary treat, with the first few Android versions being called "Cupcake", "Donut", "Eclair", and "Froyo", in that order. During its announcement of Android KitKat in 2013, Google explained that "Since these devices make our lives so sweet, each Android version is named after a dessert", although a Google spokesperson told CNN in an interview that "It's kind of like an internal team thing, and we prefer to be a little bit how should I say a bit inscrutable in the matter, I'll say".

Google introduced the Pixel and Pixel XL smart phones in October 2016, marketed as being the first phones made by Google, and exclusively featured certain software features, such as the Google Assistant, before wider rollout. The Pixel phones replaced the Nexus series, with a new generation of Pixel phones launched in October 2017.



Code name	Version number	Initial release date	API level	Security patches[1]
(No codename)[2]	1.0	September 23, 2008	1	Unsupported
(Internally known as "Petit Four")[2]	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut[3]	1.6	September 15, 2009	4	Unsupported
Eclair[4]	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo[5]	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread[6]	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb[7]	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich[8]	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean[9]	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat[10]	4.4 – 4.4.4	October 31, 2013	19 – 20	Unsupported[11]
Lollipop[12]	5.0 – 5.1.1	November 12, 2014	21 – 22	5.1.x Supported
Marshmallow[13]	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat[14]	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
Oreo[15]	**8.0 – 8.1**	August 21, 2017	26 – 27	Supported

Legend: 🟥 Old version 🟨 Older version, still supported 🟩 **Latest version**

Fig 3.1 Android versions

3.5 ARCHITECTURE OF ANDROID

The figure the four major layers of Android operating systems. The layers are

(i) Applications layer

(ii) Application Framework layer

(iii) Libraries and Android Runtime layer

(iv) Linux kernel layer

The Android SDK includes a variety of custom tools that help develop mobile applications on the Android platform. The most important of these are the Android Emulator and the Android Development Tools plugin for Eclipse, but the SDK also includes a variety of other tools for debugging, packaging, and installing r applications on the emulator.

Android Emulator A virtual mobile device that runs on computer use the emulator to design, debug, and test r applications in an actual Android run-time environment. Android Development Tools Plug-in for the Eclipse IDE.

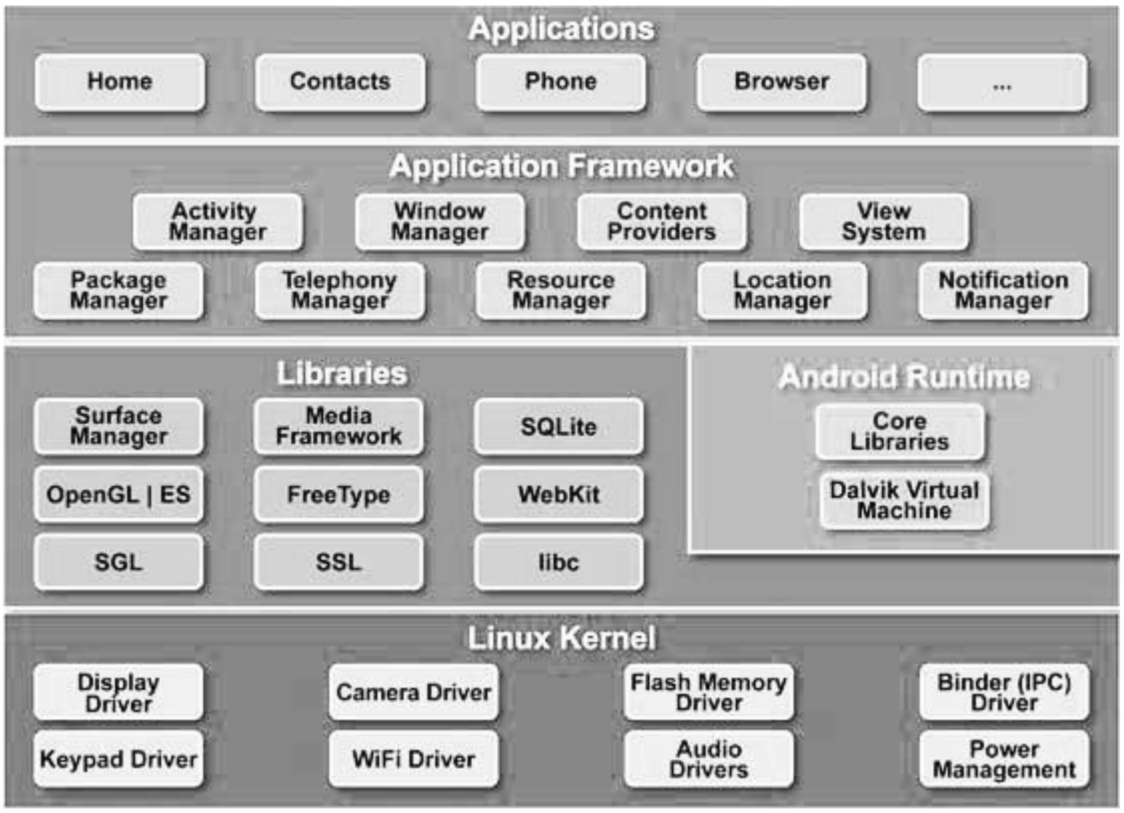


Fig 3.2 Android Architecture

The ADT plug-in adds powerful extensions to the Eclipse integrated environment, making creating and debugging r Android applications easier and faster. If use Eclipse, the ADT plug-in gives an incredible boost in developing Android applications:

It gives access to other Android development tools from inside the Eclipse IDE. For example, ADT lets access the many capabilities of the DDMS tool — taking screenshots, managing port-forwarding, setting breakpoints, and viewing thread and process information directly from Eclipse.

It provides a New Project Wizard, which helps quickly create and set up all of the basic files will need for a new Android application. It automates and simplifies the process of building r Android application. It provides an Android code editor that helps write valid XML for r Android manifest and resource files. Dalvik Debug Monitor Service

Integrated with Dalvik, the Android platform's custom DVM, this tool lets manage processes on an emulator or device and assists in debugging. Can use it to kill processes, select a specific process to debug, generate trace data, view heap and thread information, take screenshots of the emulator or device, and more.
Android Debug Bridge. The ADT tool lets install application's .apk files on an emulator or device and access the Emulator or device from a command line. Can also use it to link a standard debugger to application code running on an Android emulator or device. Android Asset Packaging Tool. The adapt tool lets create .apk files containing the binaries and resources of Android applications. Android Interface Description Language. Aidl Lets generate code for an inter process interface, such as what a service might use. sqlite3. Included as a convenience, this tool lets access the SQLite data files created and used by Android applications.

CHAPTER 4
RESULT AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 RESULTS AND DISCUSSION

This project highlights the results of the Application and the snapshots for each of the activities are shown along with the discussion of each activity describing it's working Each snapshot describes every single step of Greendroid Three main activities as well the options provided to the users in each activity such as Front end, Back end and Styling, these activities which are created on the click of these options are shown and described This Application has successfully completed various types of test cases which were applied to this project has impressed with it's amazing User interference and the design It could easily be able to calculate the loan amount within short span of time. Code was neither too complex nor to simple and it was neat and clean with well indentation and with proper styling.

4.2 EXISTING SYTEM

Existing System analyzes sensory data, once received by an application, would be transformed into various forms and used by different application components. Using existing method difficult scheduling event handlers in android programming. In previous method is only monitor the charging level and it intimate the alert message for the user.

Disadvantages

- Manual instrumentation is undesirable because it is labor-intensive and error-prone.
- There is still no well-defined metric for judging ineffective utilization of sensory data automatically.
- Existing Green droid tool analyze GPS and Sensory based application in smart phones.
- It is no off the mobile charging states.

4.3 PROPOSED SYSTEM

In this project, we proposed a two-layered permission based detection scheme for detecting malicious Android applications. Proposed monitor application for execution and perform dynamic data flow analysis at a byte code instruction level. Implement Java Path Finder (JPF) which is an explicit-state model checker that directly works with Java byte code instructions. Improved Green droid tool finds application execution while sharing application at time of blue tooth sharing. In this application is used to monitor the charging status of the mobile and it disconnect the charger when the user accesses the application such as calls and music player during the mobile charging.

Advantages

- Increase battery usages while running various state spaces.
- To turn off the charging state of the mobile.

CHAPTER 5
CONCLUTION AND FUTURE ANALYSIS

5.1 SUMMARY

Existing System analyzes sensory data, once received by an application, would be transformed into various forms and used by different application components. Using existing method difficult scheduling event handlers in android programming. In previous method is only monitor the charging level and it intimate the alert message for the user. In this project, we proposed a two-layered permission-based detection scheme for detecting malicious Android applications. Proposed monitor application for execution and perform dynamic data flow analysis at a byte code instruction level. Implement Java Path Finder (JPF) which is an explicit-state model checker that directly works with Java byte code instructions. Improved Green droid tool finds application execution while sharing application at time of blue tooth sharing. In this application is used to monitor the charging status of the mobile and it disconnect the charger when the user accesses the application such as calls and music player during the mobile charging. When the battery percentage gets (goes) lower than, the threshold value, the screen becomes weak and automatically the services like Blue-tooth, Wi-Fi, GPS get turn off.

5.2 CONCLUSION

Green Droid is useful and effective for diagnosing energy problems in Android applications, and its idea may also complement and contribute to existing resource leak detection work on the Android platform. In this application is developed by monitoring the charging status level. To disconnect the charger while user tries to attend the calls during the charging and it also disconnect the charging state for protect the human health.

5.3 FUTURE ANALYSIS

Future work analyzes the desired properties of risk signals and relative risk scores for Android apps in order to generate another metric that users can utilize when choosing apps. We present a wide range of techniques to generate both risk signals and risk scores that are based on heuristics as well as principled machine learning techniques. And also extend our approach to implement this application in windows OS and Linux OS system.

REFERENCE

1. Morrill, Dan (September 23, 2008). "Announcing the Android 1.0 SDK, release 1". Android Developers Blog. Google. Retrieved March 11, 2017.]]
2. J. Gao, X. Bai, W.-T. Tsai, and T. Uehara, “Mobile application testing: A tutorial,” Computer, vol. 47, no. 2, pp. 46–55, Feb 2014.
3. M. Couto et al. · GreenDroid: A Tool for Analysing Energy Consumption in the Android Ecosystem
4. F. Ding, F. Xia, W. Zhang, X. Zhao, and C. Ma, “Monitoring energy consumption of smartphones,” *CoRR*, 2012.
5. A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, “Mobile malware detection through analysis of deviations in application network behavior,” Computers & Security, 2014.
6. K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, “Copperdroid: Automatic reconstruction of android malware behaviors,” in NDSS Symp. Internet Society, February 2015.
7. D. Kirat, G. Vigna, and C. Kruegel, “Barecloud: bare-metal analysis-based evasive malware detection,” in Proc. USENIX SEC’14., 2014, pp. 287–301.
8. S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, and A. Sadeghi, “Xmandroid: A new android evolution to mitigate privilege escalation attacks,” Tech. Universitat Darmstadt, Tech. Rep., 2011.
9. J. Calvet, J. M. Fernandez, and J.-Y. Marion, “Aligot: cryptographic function identification in obfuscated binary programs,” in Proc. ACM, ser. CCS ’12. ACM, 2012, pp. 169–182.
10. S. Schrittwieser, S. Katzenbeisser, P. Kieseberg, M. Huber, M. Leithner, M. Mulazzani, and E. Weippl, “Covert computation: hiding code in code for obfuscation purposes,” in Proc. 8th ACM SIGSAC, ser. ASIA CCS ’13. New York, NY, USA: ACM, 2013, pp. 529–534.
11. M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, “Semantics-aware malware detection,” in Security and Privacy, 2005 IEEE Symposium on, May 2005, pp. 32–46.

APPENDIX

A.SOURCE CODE

```
package com.example.greendroid;
import android.support.v7.app.ActionBarActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
public class MainActivity extends ActionBarActivity {
        ImageButton set1,b1,b2;
        @Override
protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                set1=(ImageButton)findViewById(R.id.imageButton1);
                b1=(ImageButton)findViewById(R.id.imageButton2);
                b2=(ImageButton)findViewById(R.id.imageButton3);

set1.setOnClickListener(new OnClickListener(){
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                //startActivity(new
Intent(getApplicationContext(),main_setting.class));
                                Intent i=new Intent(Intent.ACTION_MAIN);
                                startActivity(new
Intent(getApplicationContext(),main_setting.class));
                        }
                });
b1.setOnClickListener(new OnClickListener(){
```

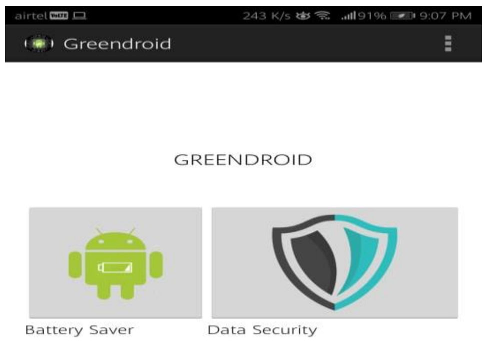
```
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                startActivity(new
Intent(getApplicationContext(),power_save.class));
                        }
                });
b2.setOnClickListener(new OnClickListener(){
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                startActivity(new
Intent(getApplicationContext(),data_security.class));
                        }
                });
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.main, menu);
                return true;
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
        }
}
```

```
package com.example.greendroid;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
public class netban extends Activity{
        ImageButton ab,cb,ib,kb,sb;
@Override
protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.netban);
                ab=(ImageButton)findViewById(R.id.imageButton1);
                cb=(ImageButton)findViewById(R.id.imageButton2);
                ib=(ImageButton)findViewById(R.id.imageButton3);
                kb=(ImageButton)findViewById(R.id.imageButton4);
                sb=(ImageButton)findViewById(R.id.imageButton5);
                ab.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
        goToUrl("https://retail.axisbank.co.in/wps/portal/rBanking/axisebanking/AxisR
etailLogin/!ut/p/a1/04_Sj9CPykssy0xPLMnMz0vMAfGjzOKNAzxMjIwNjLwsQp0MDB
w9PUOd3HwdDQwMjIEKIoEKDHAARwNC-
sP1o_ArMYIqwGNFQW6EQaajoiIAVNL82A!!/dl5/d5/L2dBISEvZ0FBIS9nQSEh/");
                        }
        private void goToUrl(String string) {
                                // TODO Auto-generated method stub
                                Uri uri=Uri.parse(string);
                                Intent WebView=new Intent(Intent.ACTION_VIEW,uri);
                                startActivity(WebView);
                        }
```

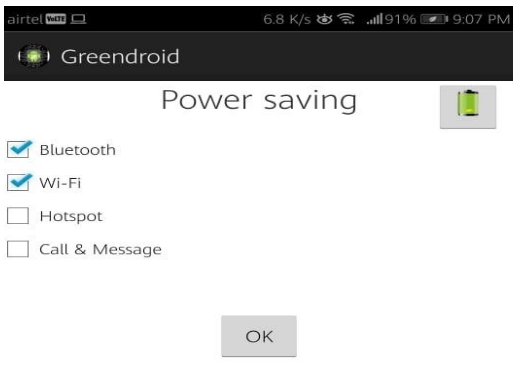
```
          });
cb.setOnClickListener(new OnClickListener(){
     @Override
     public void onClick(View arg0) {
                    // TODO Auto-generated method stub
     goToUrl("https://www.centralbank.net.in/servlet/ibs.servlets.IBSLoginServlet");

               }
     private void goToUrl(String string) {
                    // TODO Auto-generated method stub
                    Uri uri=Uri.parse(string);
                    Intent WebView=new Intent(Intent.ACTION_VIEW,uri);
                    startActivity(WebView);
               }
          });
ib.setOnClickListener(new OnClickListener(){
     @Override
     public void onClick(View arg0) {
                    // TODO Auto-generated method stub
                    goToUrl("https://www.iobnet.co.in/ibanking/login.do");
               }
     private void goToUrl(String string) {
                    // TODO Auto-generated method stub
                    Uri uri=Uri.parse(string);
                    Intent WebView=new Intent(Intent.ACTION_VIEW,uri);
                    startActivity(WebView);
               }
          });
kb.setOnClickListener(new OnClickListener(){
     @Override
     public void onClick(View arg0) {
                    // TODO Auto-generated method stub
                    goToUrl("https://www.kvbin.com/B001/ENULogin.jsp");
               }
          private void goToUrl(String string) {
                              // TODO Auto-generated method stub
                              Uri uri=Uri.parse(string);
                              Intent WebView=new Intent(Intent.ACTION_VIEW,uri);
                              startActivity(WebView);
                         }
               });
     sb.setOnClickListener(new OnClickListener(){
          @Override
          public void onClick(View arg0) {
                              // TODO Auto-generated method stub
                              goToUrl("https://retail.onlinesbi.com/retail/login.htm");
                         }
          private void goToUrl(String string) {
                              // TODO Auto-generated method stub
                              Uri uri=Uri.parse(string);
                              Intent WebView=new Intent(Intent.ACTION_VIEW,uri);
                              startActivity(WebView);
                         }
               });
          }
```

SCREEN SHOTS

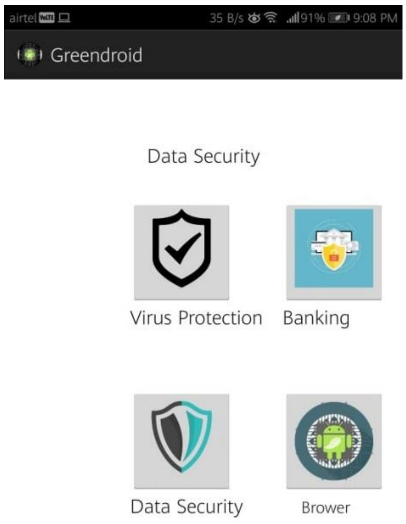


Application

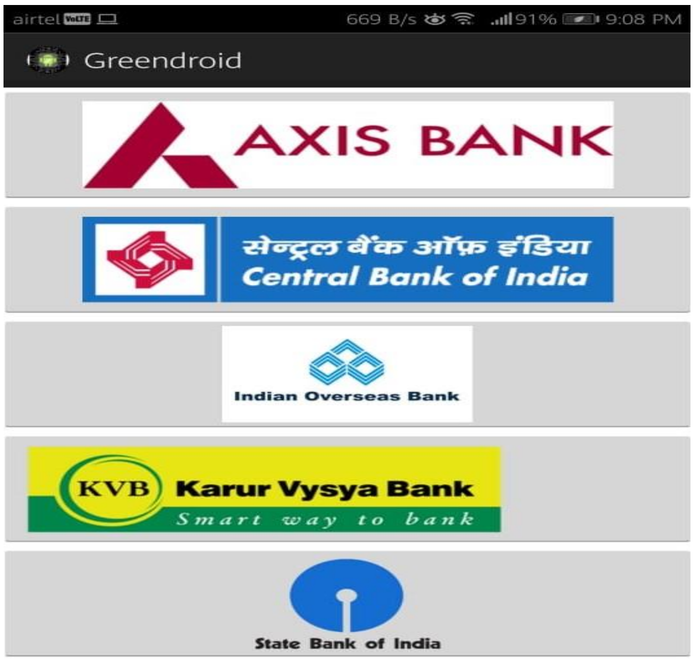


- [✓] Bluetooth
- [✓] Wi-Fi
- [ ] Hotspot
- [ ] Call & Message

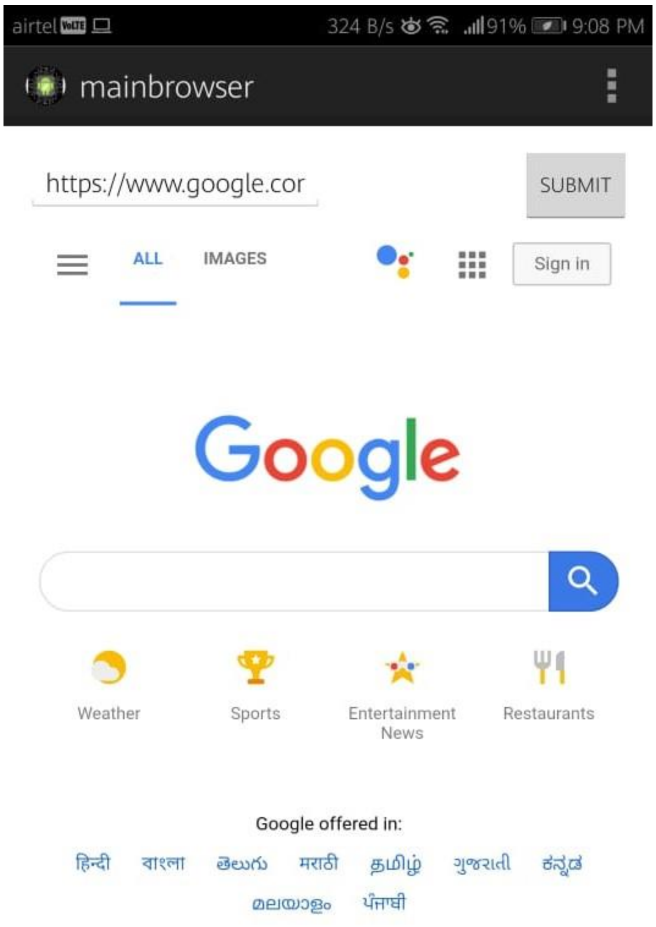
Power Saving Mode



Data Security



Banking



Secure Browser