Energy Efficient and Verifying Smart Phone Applications

Renugadevi. R[1], Dr. N. Shenbagavadivu[2]
[1, 2] Department of Computer Applications
[1, 2] Anna university BIT- Campus, Trichy

Abstract- *Energy efficiency in smart phone applications are most important but many android application suffer from serious energy inefficiency problem .The aim of the project is to develop mobile application which will manage energy utilization efficientlyin smart phones, tablet computers and other mobile devices. This application may reduce battery usages. Especially the Blue-tooth, Wi-Fi and GPS are the main concerns that drain the battery power. It manage the battery level in smart device . Initially the mobile user get the battery level from the mobile device are the current battery Percentage, battery health, voltage and temperature in next part we set a threshold value for the battery level when it gets lower the threshold value the screen gets weak Blue-tooth, WI-Fi , GPS gets turned off. So provide Green droid tool to analyse state space for each application using model checker concept. Then privacy protection against mobile applications on mobile devices is becoming a serious concern as user sensitive data may be leaked without proper justification.*

Keywords- smart phone, energy efficient, battery power, greendroid, temperature, privacy production

I. INTRODUCTION

One of the most widely used mobile OS these days is ANDROID. Android is a software bunch comprising not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been number of updates in the original version of Android. The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touch screen input, it has also been used in game consoles, digital cameras, regular P Cs, and other electronics. As of 2015, Android has the largest installed base of all operating systems Android is a Linux- based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first Android-powered phone was sold in October 2008.

These factors have allowed Android to become the world's most widely used smart phone platform, overtaking Sembilan in the fourth quarter of 2010, and the software of choice for technology companies who require a low-cost, customizable, lightweight operating system for high tech devices without developing one from scratch. As a result, despite being primarily designed for phones and tablets, it has seen additional applications on televisions, games consoles, digital cameras and other electronics. Android's open nature has further encouraged a large community of developers and enthusiasts to use the open source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially released running other operating systems. Android had a worldwide smart phone market share of 75% during the third quarter of 2012, with 500 million devices activated in total and 1.3 million activations per day. The operating system's success has made it a target for patent litigation as part of the so-called "smart phone wars" between technology companies.

Android devices are vulnerable to threats and some of the noticeable vulnerabilities are denial of service attacks, execution of code by the attackers using Android debugger bridge (adb), stack based buffer overflow resulting in arbitrary code execution, memory corruption to gain root privileges, SQL injection to retrieve useful information, cross site scripting to redirect to other vulnerable domains and to steal user credentials etc. Some other issues are intercepting SMS to gain user credentials and in some cases API's can be queried for retrieving information.

In this project apply static analysis with the aim to understand the significance of permissions for the identification of unseen samples. Investigation is carried out to identify the most prominent permissions that contribute in mobile malware classification with few features based on the permissions it request this project, the 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 find application execution while sharing application at time of blue tooth sharing can extend our framework to analyse model checking each state space.

Analyse both energy and malware detection in smart phone applications. Implement secure applications to avoid man in middle attack. One application is to help smart phones determine their approximate location or context. With the help of green droid a smart phone's software can approximately find its relative location to green droid in a store. Green droid can help a phone show notifications of items nearby that are on sale, and it can enable payments at the point of sale (POS) where customers don’t need to remove their wallets or cards to make payment Green droid technology works using the Bluetooth Low Energy (BLE) technology, also known as Bluetooth Smart. Green droid uses Bluetooth low energy proximity sensing to transmit a universally unique identifier picked up by a compatible app or operating system. The identifier can then be looked up over the internet to determine the device's physical location or trigger an action on the device such as a check-in on social media or a push notification .With these findings, the propose an approach to automatically diagnosing such energy problems in Android applications. Our approach explores an Android application’s state space by systematically executing the application using Java Path Finder (JPF), a widely-used model checker for Java programs .

It analyses how sensory data are utilized at each explored state, as well as monitoring whether sensors/wake locks are properly used and unregistered/released.JPF was originally designed for analysing conventional Java programs with explicit control flows .It executes the byte code of a target Java program in its virtual machine. However, Android applications are event driven and depend greatly on user interactions. Their program code comprises many loosely coupled event handlers, among which no explicit control flow is specified. At runtime, these event handlers are called by the Android framework, which builds on hundreds of native library classes. As such, applying JPF to analyse Android applications requires: generating valid user interaction events, and correctly scheduling event handlers. To address the first technical issue, propose to analyse an Android application’s GUI layout configuration files, and systematically enumerate all possible user interaction event sequences with a bounded length at runtime. It can show that such a bounded length does not impair the effectiveness of our analysis, but instead helps quickly explore different application states and identify energy problems. To address the second technical issue, present an application execution model derived from Android specifications. This model captures application generic temporal rules that specify calling relationships between event handlers. With this model are able to ensure an Android application to be exercised with correct control flows, rather than being randomly scheduled on its event handlers. As it will show in our later evaluation, the latter brings almost no benefit to the identification of energy problems in Android applications.

II. RELATED WORKS

[1]Anand, M. Naik discussed Mobile devices with advanced computing ability and connectivity, such as smart phones and tablets, are becoming increasingly prevalent. At the same time there has been a surge in the development and adoption of specialized programs, called apps, that run on such devices. Apps pervade virtually all activities ranging from leisurely to mission critical. Thus, there is a growing need for software-quality tools in all stages of an app’s life-cycle, including development, testing, auditing, and deployment. Apps have many features that make static analysis challenging: a vast SDK (Software Development Kit), asynchrony, inter-process communication, databases, and graphical user interfaces (GUIs). As a result, many proposed approaches for analyzing apps are based on dynamic analysis

[2]W. Enck, P. Gilbert suggest A key feature of modern smart phone platforms is a centralized service for downloading third-party applications. The convenience to users and developers of such “app stores” has made mobile devices more fun and useful, and has led to an explosion of development. Apple’s App Store alone served nearly 3 billion applications after only 18 months . Many of these applications combine data from remote cloud services with information from local sensors such as a GPS receiver, camera, microphone, and accelerometer. Applications often have legitimate reasons for accessing this privacy sensitive data, but users would also like assurances that their data is used properly. Incidents of developers relaying private information back to the cloud and the privacy risks posed by seemingly innocent sensors like accelerometers .

[3] E. Cuervo, D. Cho suggested One of the biggest obstacles for future growth of smart phones is battery technology. As processors are getting faster, screens are getting sharper, and devices are equipped with more sensors, a smart phone’s ability to consume energy far outpaces the battery’s ability to provide it. Unfortunately, technology trends for batteries indicate that these limitations are here to stay and that energy will remain the primary bottleneck for handheld mobile devices.

[4] Dillig, T. Dillig says While automatic memory management via garbage collection has enjoyed considerable success in many widely-used programming languages, such as Java and C#, there are a number of situations where a programmer still needs to pay attention to timely resource reclamation. Consider the code , where the underlined code segments deal solely with resource de-allocation . For example, should the programmer forget to call socket .close at lines 13 and 30—which apparently serve no functional purpose in the application logic—there would be a leak of a socket resource in the system. While the application-level socket object can be reclaimed by the garbage collector when the encapsulating Buffer

III. PROPOSED SYSTEM

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 is to find application execution while sharing application at time of blue tooth sharing.

The proposed framework to analyse model checking each state space also. Analyse both energy and malware detection in smart phone applications. Implement secure applications to avoid man in middle attack.

ADVANTAGE

Diagnosing energy problems arising from sensory data under utilization.

Overcome malicious application installation.

Increase battery usages while running various state spaces.

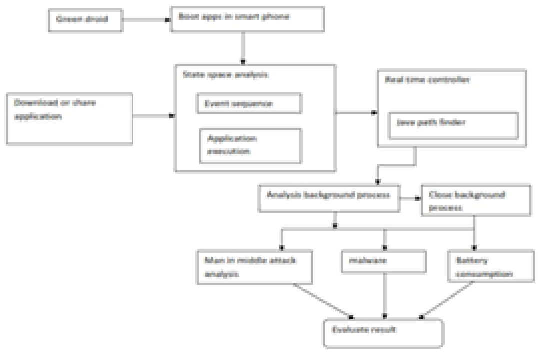


Fig.1. Green Drawing

IV. METHODOLOGY

1. Apps Booted:

In this module, helps to boot apps which are downloaded from internet using GPRS services. Shared apps from Bluetooth devices. List out the number apps in smart phones. The application scan the application's APK files that include the state space. State space includes start, kill, and pause and resumes states.

2. Active path findings:

This module, used to can get all state spaces for applications and executed file lists using Java Path Finder. This state space used to analyze in further process. Mobile application testing is a specialized and complex field. Due to mobile applications' event driven design and mobile runtime environment, there currently exist only a small number of tools to verify these applications. The JPF-ANDROID is built on Java Pathfinder, a Java model checking engine. JPF-ANDROID provides a simplified model of the Android framework on which an Android application can run. It then allows the user to script input events to drive the application flow. JPF-ANDROID provides a way to detect common property violations such as deadlocks and runtime exceptions in Android applications.

3. Malware Prediction:

Mobile virus is malicious software that targets mobile phones, by causing the collapse of the system and loss or leakage of confidential information. As phone networks have become more and more common and have grown in complexity, it has become increasingly difficult to ensure their safety and security against electronic attacks in the form of viruses or other malware. Finally it can predict and delete malwares from our smart phones. It is performed on destroy operation to delete malwares and preserve energy at the time of application usage. And also save the energy with three modes such as normal, aggressive and background mode. These modes based implementation save energy in mobile phones with improved accuracy rate.

4. Evaluation criteria:

Our approach provide improved prediction rate and minimized energy levels. The evaluation approach showed in graph in unites before or after application used. And also analyse power in normal and aggressive mode.

V. SAMPLE SCREEN SHOT

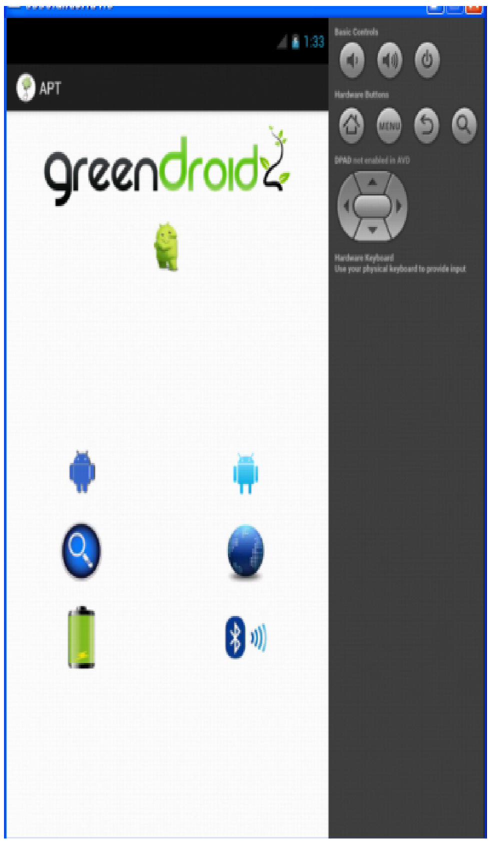
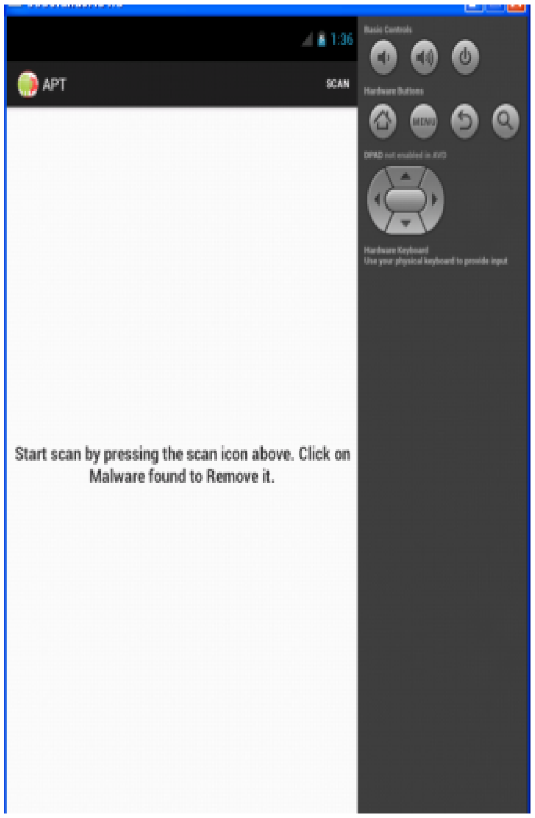
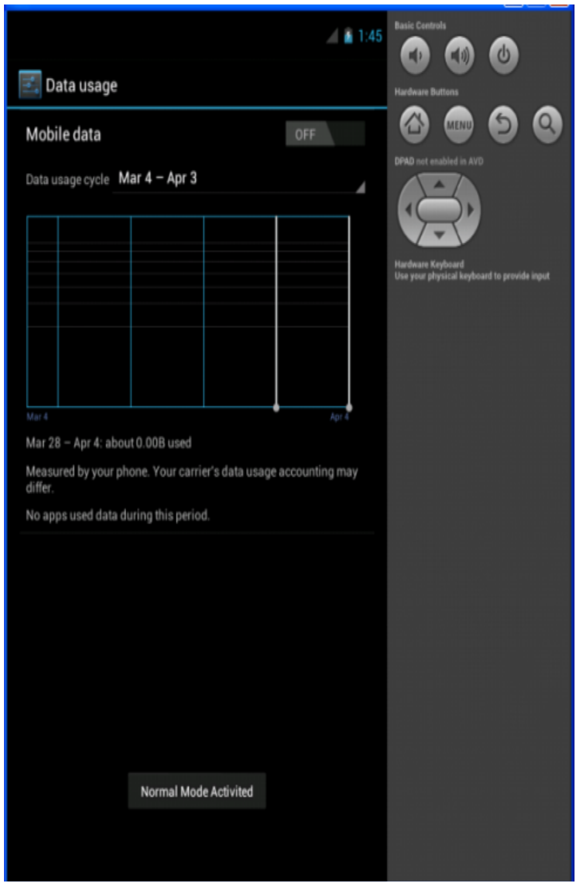


Fig. 2.Home page



Start scan by pressing the scan icon above. Click on Malware found to Remove it.

Fig. 3. Malware detection



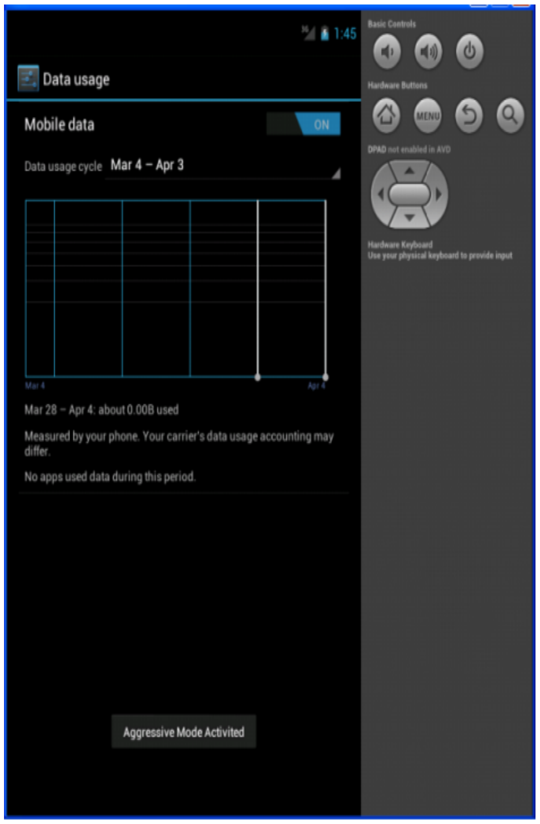
Mar 28 – Apr 4: about 0.00B used

Measured by your phone. Your carrier's data usage accounting may differ.

No apps used data during this period.

Normal Mode Activited

Fig. 3. Normal mode activation



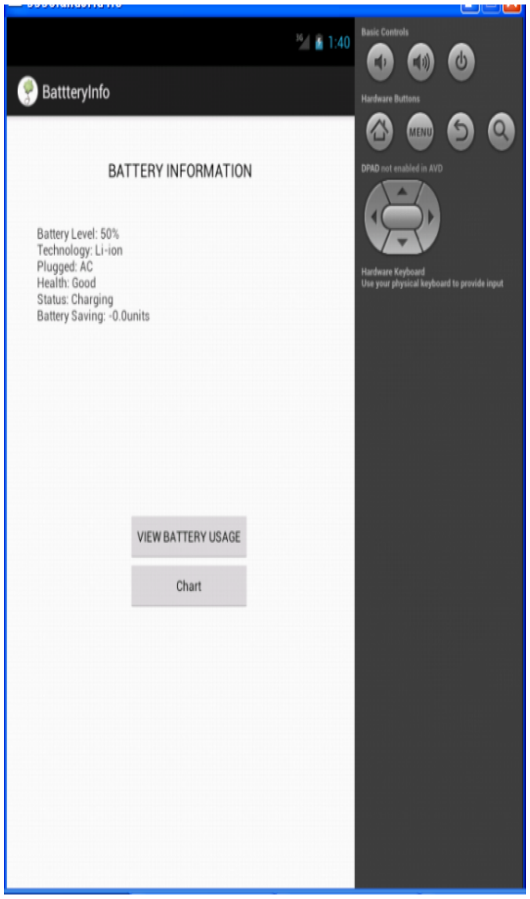
Mar 28 – Apr 4: about 0.00B used

Measured by your phone. Your carrier's data usage accounting may differ.

No apps used data during this period.

Aggressive Mode Activited

Fig. 4.Aggressive mode activation



Battteryinfo

BATTERY INFORMATION

Battery Level: 50%
Technology: Li-ion
Plugged: AC
Health: Good
Status: Charging
Battery Saving: -0.0units

Fig. 5. Battery information

VI. CONCLUSION

Mobile devices have become popular in our lives since they offer almost the same functionality as personal computers. Among them, Android-based mobile devices had appeared lately and, they were now an ideal target for attackers. Android-based smart phone users can get free applications from Android Application Market. But, these applications were not certified by legitimate organizations and they may contain malware applications that can steal privacy information for users. In this project, a framework that can detect android malware applications is proposed to help organizing Android Market. The proposed framework intends to develop a machine learning-based malware detection system on Android to detect malware applications and to enhance security and privacy of smart phone users. This system monitors various permission based features and events obtained from the android applications, and analyses these features by using machine learning classifiers to classify whether the application is good ware or malware. This project, describe state based malware detection system for android based smart phones users. This system exploits machine learning techniques to distinguish between normal and malware applications. 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.

REFERENCES

[1] S. Anand, M. Naik, M. J. Harrold, and H. Yang, "Automated concolic testing of smartphone apps," Proc. ACM SIGSOFT 20th Int'l Symp. Foundations of Soft. Engr. (FSE 12), ACM, 2012, pp. 59:1-59:11

[2] M. Arnold, M. Vechev, and E. Yahav, "QVM: an efficient runtime for detecting defects in deployed systems," ACM Trans. Software Engi-neering and Methodology, vol. 21, 2011, pp. 2:1-2:35

[3] Dillig, T. Dillig, E. Yahav, and S. Chandra, "The CLOSER: automating resource management in Java," Proc. Int'l Symp. Memory Manage-ment (ISMM 08), ACM, 2008, pp. 1-10.

[4] T. Azim and I. Neamtiu, "Targeted and depth-first exploration for systematic testing of android apps," Proc. ACM SIGPLAN Int'l Conf. Object-oriented Programming Systems Languages & Applica-tions (OOPSLA 13), ACM, pp. 641-660

[5] M. Dong and L. Zhong "Sesame: Self-constructive high-rate system energy modeling for battery-powered mobile systems," Proc. Int'l Conf. Mobile Systems, Applications, and Services (Mobisys 11), ACM, 2011, pp. 335-348

[6] Jack Sampson, Manish Arora, Nathan Goulding-Hotta, Ganesh Venkatesh, Jonathan Babb, Vikram Bhatt, Steven Swanson, and Michael Bedford Taylor "An Evaluation of Selective Depipelining for FPGA-based Energy-Reducing Irregular Code Coprocessors", ,2011 International Conference on Field Programmable Logic and Applications, September 2011.

[7] Manish Arora, Jack Sampson, Nathan Goulding-Hotta, Jonathan Babb, Ganesh Venkatesh,Michael Bedford Taylor, and Steven Swanson"Reducing the Energy Cost of Irregular Code Bases in Soft Processor Systems" - Programmable Custom Computing Machines, Annual IEEE Symposium on:210-213, 2011