A Methodology for Reliable Detection of Anomalous Behavior in Smartphones

by

Robin Joe Prabhahar Soundar Raja James

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Smartphones have become the most preferred computing device for both personal and business use. Different applications in smartphones result in different power consumption patterns. The fact that every application has been coded to perform different tasks leads to the claim that every action onboard (whether software or hardware) will consequently have a trace in the power consumption of the smartphone. When the same sequence of steps is repeated on it, it is observed that the power consumption patterns hold some degree of similarity. A device infected with malware can exhibit increased CPU usage, lower speeds, strange behavior such as e-mails or messages being sent automatically and without the user's knowledge; and programs or malware running intermittently or in cycles in the background. This deviation from the expected behavior of the device is termed an anomalous behavior and results in a reduction in the similarity of the power consumption. The anomalous behavior could also be due to gradual degradation of the device or change in the execution environment in addition to the presence of malware. The change in similarity can be used to detect the presence of anomalous behavior on smartphones.

This thesis focuses on the detection of anomalous behavior from the power signatures of the smartphone. We have conducted experiments to measure and analyze the power consumption pattern of various smartphone apps. The test bench used for the experiments has a Monsoon Power Meter, which supplies power to the smartphone, and an external laptop collects the power samples from the meter. To emulate the presence of anomalous behavior, we developed an app which runs in the background with varying activity windows. Based on our experiments and analysis, we have developed two separate models for reliable detection of anomalous behavior from power signatures of the smartphone. The first model is based on Independent Component Analysis (ICA) and the second model is based on a Similarity Matrix developed using an array of low pass filters. These models detect the presence of anomalies by comparing the current power consumption pattern of the device under test with that of its normal behavior.

Acknowledgements

Praise to the Lord, God Almighty, for all that I am and have is from him.

I would like to extend my sincere thanks to my supervisor Prof. Kshi-rasagar Naik, who has been a mentor and guide, and a constant source of support and encouragement. Without his guidance, patient feedback and valuable insights, completion of this thesis work would not have been possible.

I would like to thank Prof. Mohamed-Yahia Dabbagh who was my co-supervisor for his guidance and support. I would also like to thank Prof. Catherine H. Gebotys and Prof. Igor Ivkovic for being my readers.

I would also like to thank my family for their encouragement, love and support. I am very grateful to my colleagues and friends for motivating me and being supportive through the two years of my program.

Dedication

This is dedicated to Abba, father; my parents; and my sister.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Smartphones have become an indispensable part of modern life. Hand-held mobile devices can perform many online tasks, including web browsing, internet banking, multimedia streaming, and connecting people through social media, to name a few. When not connected to the internet, smartphones can be used as a music player, camera, document viewer, and a gaming console. The continued miniaturization, increased portability, user friendly interface, better connectivity, enhanced computational power, improved battery life and affordable costs have made them popular for both personal and business use. Studies have found that mobile usage has been outpacing desktops and laptops [31]. The number of smartphone users has increased at a rapid pace [6]. It is estimated that there are currently around 2.4 billion people using smartphones [45] and forecasts predict that the number will rise to 2.87 billion by 2020 [6].

Google's Android and Apple's iOS are the two most popular smartphone operating systems with Android accounting for nearly 80% and iOS accounting for about 15% of all smartphone sales. Their growth in popularity is driven by the wide range of applications (apps) now available in the market. Recent years have seen an increase in the usage of smartphones in healthcare [48] [14], psychotherapy [20] and health research [24]. Smartphone applications are used as learning aids in class rooms [10], in forensic data acquisition [18], banking [15], road and traffic monitoring [47], as interfaces to wearable devices, home security and automation and social media. They process large amounts of information and have become repositories of personal, financial and corporate data that need to be protected. Malware attacks can cause data loss or data theft, hardware failure, loss of connectivity and may affect the whole infrastructure. With companies and enterprises allowing employee-owned devices to connect to their networks, in a concept known as "bring your own device" (BYOD), security stakes are at the highest.

An anomaly is any deviation from the expected behavior of the device and the anomalous behavior could be due to gradual degradation of the device, change in the execution environment or due to the presence of an external agent such as a malware. The presence of an anomaly even in a single connected wireless device could affect the whole network and may lead to loss of confidential data, data theft or loss of connectivity and defeat the purpose of having a well connected network. Smartphones are fast becoming the focal point of most of the wireless networks and are pivotal in IoT; and hence the need for anomaly detection in smartphones is imperative.

1.1 Motivation

The subject of this thesis was motivated by the following facts:

1. ***The growing popularity of smartphones and their application in security critical applications.*** A recent survey reported that global mobile internet penetration is going to increase to 71% in 2019 from 28% in 2013 [54]. With this popularity, mobile devices have become the fastest growing consumer technology. The growing popularity of smartphones, wearables and other wireless connected devices; and their increasing application in security-critical applications, namely, banking, health monitoring, official transactions and smart grids to name a few, has made security in smartphones critical.

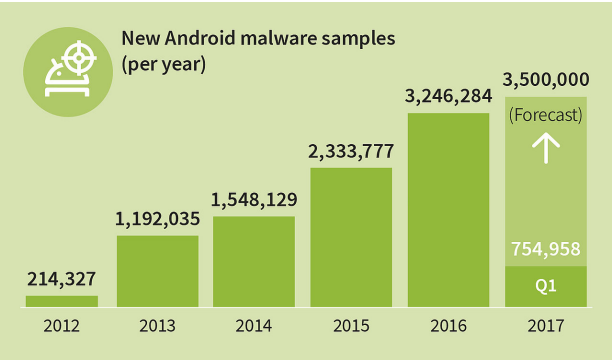


Figure 1.1: Number of new Android malware samples from 2012 to 2017 (per year) [13]

2. ***Increase in the number of malware attacks.*** The number of malware attacks have increased over the years. According to a report by McAfee Labs [44], around

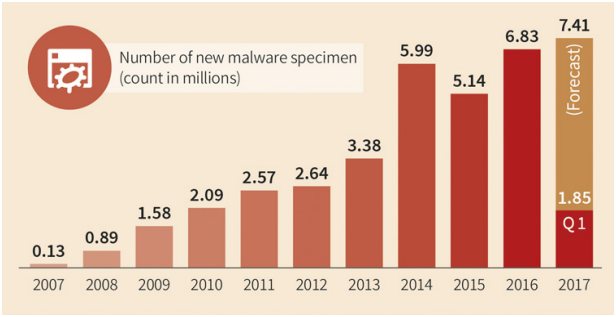


Figure 1.2: Number of new malware specimen (count in millions) from 2007 to 2017 [13]

13% of the mobile users have reported malware infection with users in Asia reporting as high as 21%. The attacks keep modifying the malware and this makes detection of malware a challenge. A report by G-Data security [13] has estimated that the number of new malware specimens have risen from 0.13 million in 2007, to 6.83 million in 2016 and could be around 7.41 million in 2017, with new android malware samples alone accounting for 3.5 million as seen in Fig. 1.1 and Fig. 1.2.

3. ***Presence of hacked versions of legitimate applications.*** Mobile Applications or Apps as they are popularly called, are now preferred over mobile websites. The latest data from Yahoo's Flurry analytics shows that 90% of consumer's mobile time is spent on apps, and with mobile usage fast outpacing desktop or laptop, it is essential to secure the apps available in the market. Third-party app stores are notorious for delivering hacked versions of legitimate applications that often contain malware. Even the official app stores can suffer problems. An article by Guardian [31], reports that as many as 4000 apps in Apple's official App Store, considered to have strong security, have been infected with malware. A study by Symantec in 2014 found that, of all the Android apps scanned, 33% were leaking SIM card information such as address book details, mobile PIN numbers and call history.

Therefore with these facts in mind, we were motivated to study the power consumption patterns in smartphone to attempt to delineate anomalous behavior from normal behavior. We conducted experiments to characterize the normal behavior of the smartphone and based on our findings we were motivated to develop two models for autonomous anomaly detection. The first model described in Chapter 4, uses Independent Component Analysis (ICA), and works well for malware which have a significant impact on the probability distribution of power consumption of the smartphone. Chapter 5 proposes a more generic model and makes use of a Similarity Matrix to detect anomalous behavior in smartphone.

1.2 Problem Statement

An anomaly is any deviation from the expected behavior of the device. The anomalous behavior could be due to gradual degradation of the device, change in the execution environment or due to the presence of an external agent such as malware. Security in smartphones has become critical with the increase in the number of malware attacks that target smartphones and other connected devices. Smartphones have become an indispensable part of modern life and contain large amounts of critical data and need to be protected. Further, smartphones are an interface to various wireless devices and are, also, a significant part of IoT. Hence, a compromise on the security of a smartphone can not only lead to loss/theft of data, but also, affect the whole network and infrastructure.

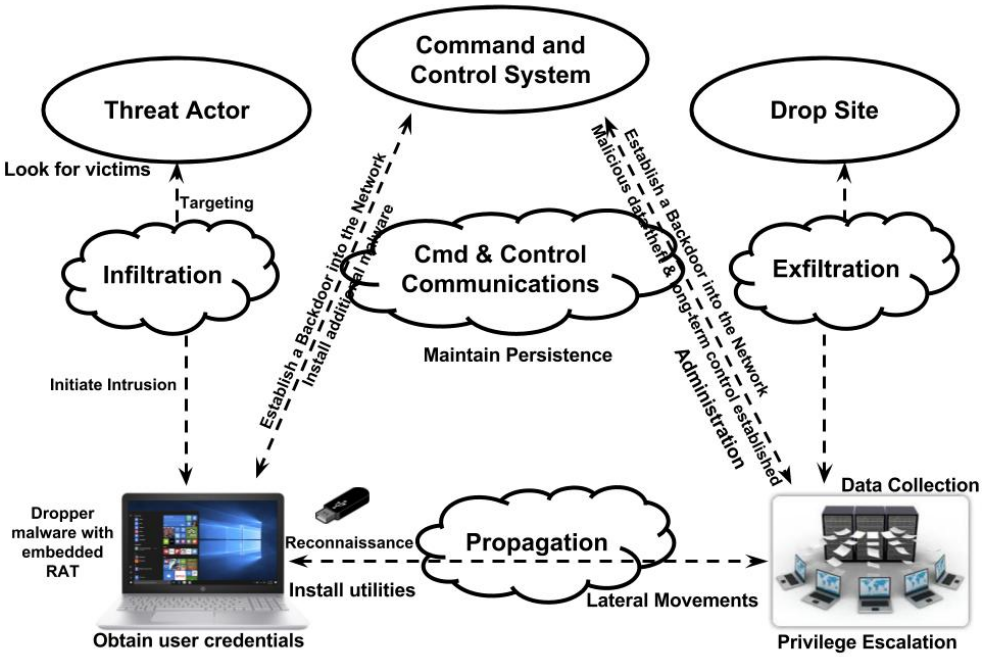


Figure 1.3: Lifecycle of a modern attack

The main problem explored in this thesis is as follows. The increase in the number of smartphone users has led to a tremendous increase in the number of mobile applications (apps) that can be used on the phones. Users prefer these apps over traditional websites as they are easy to access and use. Most of the smartphones can access a wide array of

data and interfaces in the smartphone. Recent years have seen an alarming increase in the number of fake illegitimate apps that can steal user information; and even take control or corrupt the legitimate apps. Some dubious apps download codes of malware as they are downloaded to the phone as shown in the malware threat model in Fig. 1.3. These malware codes can lie dormant in the phone with short intervals of activity, making them difficult to detect. This thesis focuses on developing a methodology for reliable detection of anomalous behavior from the power signatures of the device.

1.3 Solution Strategy and Contributions

The model proposed in this thesis detects anomaly in a smartphone based on inconsistency in its behavior. An inconsistency in the behavior of a smartphone can be observed through various parameters such as reduced speed, suspicious system calls, increased CPU utilization and battery drain. Since a smartphone or any wireless device requires a constant source of power, its power consumption pattern can serve as a reference for the behavior of the device and account for almost any kind of anomalous behavior. Anomalous behavior detection consists of three steps, shown in Fig. 1.4.

Figure 1.4: Anomalous Behavior Detection - General Approach

The first step in anomaly detection is monitoring, which involves collecting samples or data of the parameters such as API calls [62] [12], dataflow patterns [28], system logs [30] [64], CPU and resource utilization [56], and energy and power consumption [54] [40] [66] [17] [38]. The next step consists of analyzing the data collected, using algorithms or anomaly detection models and based on the results, the presence of anomaly is decided. The data can be collected internally using apps such as PowerTutor [69] [64], using smart batteries [40] [38]; or externally using a monitoring tool. Similarly, the algorithm or model used for analyzing the data and detecting malware, can be run on the device or off the device. The algorithms used for detecting anomalies are usually complex and can use lot of power and resource [62], [38]. Further, these algorithms themselves may be targeted by malware [54]. To overcome these problems, the anomaly detection model developed in this

thesis is based on off-device monitoring and off-device analysis and decision. The details of this model can be found in Chapter. 4 and Chapter. 5.

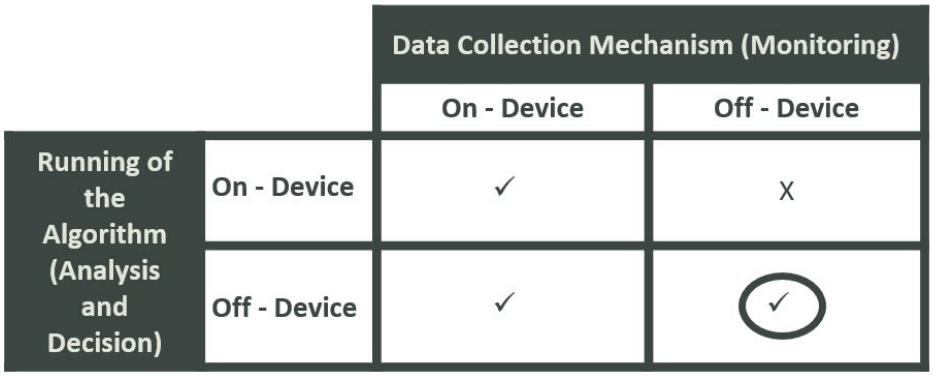


Figure 1.5: Monitoring, Analysis and Decision - Different Methods

This work makes the following contributions that are listed below.

- Proposes two approaches to detect anomalous behavior in smartphones. The first approach explores the use of Independent Component Analysis (ICA) in detecting anomalous behavior based on power consumption traces, and the second approach proposes a similarity matrix to detect the anomalous behavior.
- Analyses, evaluates and compares the two approaches based on the data collected through a background app developed to resemble an anomaly in the smartphone.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Related research and background are presented in Chapter 2. Chapter 3 describes the experimental setup used for acquiring data and the methodology for data acquisition. The proposed anomaly detection methodology based on ICA is explained in Chapter 4 and a evolution of this methodology is described in Chapter 5. Finally, Chapter 6 documents conclusions of this thesis and recommendations for future work.

Chapter 2

Background and Literature Survey

2.1 Introduction

This chapter provides a brief introduction on anomalies in smartphones and wireless devices and the need to build an effective anomaly detection framework/methodology. This is followed by a discussion on the challenges faced in detecting anomalies and the literature on the existing detection techniques. Finally, this chapter provides an overview on ICA and comparison of signals, which have been used in the model proposed in this thesis.

2.2 Anomaly in Wireless Devices

Anomalies are patterns in data that do not conform to a well defined notion of a normal behavior. An anomaly could be caused in a device for any of the following reasons.

- gradual degradation or ageing of the device,
- physical damage to the device,
- change in the execution environment, and
- malware attcks on the device.

2.2.1 Types of Anomalies and their Causes

It is important to study the types of anomalies since most of the detection methods are specific to certain kinds of anomalies [19].

Point Anomalies

This is the simplest type of anomaly. An anomaly is called a point anomaly if an individual data instance is considered as anomalous with respect to the rest of data. An example of a point anomaly is credit card fraud detection.

Contextual Anomalies

An anomaly is termed as a contextual or a conditional anomaly if a data instance is anomalous in a specific context, but not otherwise [58]. Notion of a context is induced by the structure in the data set. Each data instance is defined using the following two sets of attributes [19]:

- **Contextual attributes.** The contextual attributes are used to determine the context (or neighborhood) for that instance. For example, in spatial data sets, the longitude and latitude of a location are the contextual attributes. In time-series data, time is a contextual attribute that determines the position of an instance on the entire sequence
- **Behavioral attributes** The behavioral attributes define the noncontextual characteristics of an instance. For example, in a spatial data set describing the average rainfall of the entire world, the amount of rainfall at any location is a behavioral attribute.

Collective Anomalies

An anomaly is called collective anomaly when a collection of data instances is anomalous with respect to the entire data.

The anomalies considered in this thesis are an extension of a point and collective anomaly and are explained in Section 3.1.3.

2.2.2 Challenges in Anomaly Detection

An anomaly is any deviation from the expected behavior. A mechanism which monitors the behavior of a device and flags an aberration might seem a sufficient strategy. However, several constraints make this approach challenging [19]:

- Modeling or defining the normal behavior of a device is difficult since devices have diverse uses and are dynamic in their behavior. This makes it difficult to differentiate anomalous behavior from normal behavior.
- Anomalies resulting from malware attacks are difficult to detect since malwares keep adapting themselves trying to duplicate normal behavior in an attempt to go undetected.
- Device software updates often result in a change in the behavior of the devices and the rules of detection might flag them as an anomaly or malware. Hence, the rules or criteria used to detect malware need to be constantly updated.
- The definition of anomaly is different for different domains. For example, in the medical domain a small deviation from normal (e.g., fluctuations in body temperature) might be an anomaly, while similar deviation in the stock market domain (e.g., fluctuations in the value of a stock) might be considered as normal. Thus applying a technique developed in one domain to another, is not straightforward.
- It is difficult to obtain ground data for training/validating models used by anomaly detection techniques.
- Even the data available might often contain noise which might make legitimate data look malicious.

Due to these challenges, a methodology for reliable detection of anomaly is imperative. Any wireless device needs a constant supply to power. These devices are usually optimized and are consistent in their power consumption. Hence, an intrusion of any kind will result in an abnormal power consumption pattern. Hence, monitoring and analyzing the power consumption pattern can detect anomalies. However, most of the algorithms that monitor and analyze power consumption patterns are energy intensive and can cause battery drain and increased resource utilization.

2.3 Existing Literature on Anomaly Detection

In literature, there are mainly three methods to detect unknown or malicious activities in smartphones: statics analysis of app code; on-device monitoring; and external monitoring. *Static analysis* approaches analyze specific patterns in the app code to decide if the app is engaging in suspicious communication. William Enck [29] associates intents with certain APIs (Application Program Interface) and track the flow of those intents through the code to establish links between intents and UI (User Interface) elements. If an API can not be traced to a UI element, it is suspected as a malicious activity. Buennemeyer [16] identifies potential malware by performing static data-flow analysis of code to track dependence between the identification and use of data; if a function call cannot be traced to a user gesture, then it is detected as a suspicious behavior. Karim et a;. [27] identify covert communication by looking for code patterns where no response is presented to the user, neither on success nor on failure of the API. *static analysis* is effective in detecting malware, however, this approach has some drawbacks, which are :

- access to the source code is required, and
- the designers of malicious code soon learn about the hypotheses in static analysis and adapt their intrusion strategies.

In the *on-device monitoring* approaches, the key idea is to monitor the power consumed by a device via a software tools snd detect malicious code using power signals. Researchers in [34] have developed machine learning based techniques to classify apps according to their power consumption to facilitate detection of suspicious apps. They have used power histograms, Mel Frequency Cepstral Coefficients (MFCC) and Gaussian Models to classify apps. Researchers in [7] has proposed a rule-based engine to generate warnings for the user based on irregularities in power consumption. The effectiveness of the *on-device monitoring* approaches is limited by two factors: (i) low sampling rate of the power signal; and (ii) requirement to keep the processing techniques simple so as to put a low load on the mobile device. In the *off-device monitoring* approach, the key idea is to measure the power with external equipment and process the power signals to detect malicious activities. The authors in [53] have built a database of power profiles of legitimate applications and variants of mobile malware. The power profile of an unknown app is compared against the database using the χ^2-distance metric, to decide whether it is malware. Creating a database of power profiles for each smartphone poses a great challenge for this approach. Extraction of features in frequency domain by applying Fourier Transform on time series power consumption signals of different applications is proposed by Zhang et al.[68]; however

this idea is in a nascent state. Out of these three methods, *off-device* approaches offer more promising solutions in detecting unknown and malicious activities in smartphone, because this method imposes the least load on the device. Moreover, presence of malicious code doesn't affect this method unlike the case of *on-device monitoring* approach.

Battery based intrusion detection was proposed by Jacoby et al. [38] . Power consumption measurements were used by Kim et al. [41] and Guri et al.[33] for detecting energy-greedy anomalies and mobile malware variants, and sensitive activities respectively. Apart from power signature analysis, machine learning methodologies have been explored in detecting suspicious activities [52],[42]. The work by Xue et al. [63] focuses on anti-malware tools adaptability for changing malware intrusion pattern. Similar research has been reported in reference [39] to prove the effectiveness of a model to detect malware in different operation scenarios of mobile devices. Machine learning methods such as Support Vector Machine (SVM) have been applied in detecting malware [33]. A vast majority of the work for detecting malware using machine learning methodology focus on data from different applications to extract features. These methods can not be generalized for all mobile devices. A summary of the anomaly detection techniques based on power consumption is shown in Table 2.1.

Table 2.1: Summary of Anomaly Detection Techniques based on power consumption

Ref. #	Proposed Solution	Validation/Results	Observations
[41]	Energy consumption based malware detection by using power-aware malware-detection framework.	99% true-positive rate achieved in classifying mobile malware.	Implemented on HP iPAQ running a Windows Mobile OS. Untested on present day smartphones.
[21]	Proposes a model implemented in a kernel module, to build up an energy signature of both legal and malicious behaviors of WiFi hardware component in different Android devices.	The experiments conducted show that every activity has an immediate and noticeable impact on the Energy consumption of the device.	Results indicate that Energy monitoring can be used to detect security threats on Android based devices.

[38]	This technique correlates network attacks with their impact on device power consumption using a rules-based Host Intrusion Detection Engine (HIDE).	Effective Host based Intrusion detection technique.	Limited by resource utilization such as battery usage.
[63]	*Virus Meter*, a tool that uses energy consumption comparison between a clean system and when malicious activities have been performed on the device. This model is implemented using three different approaches: Linear Regression, Neural Networks (NN) and Decision Trees.	Results show a high detection rate, however linear regression has a high rate of false positives. Neural network is reported to achieve the best results among the three approaches.	This model can become ineffective if malware injects fake events in the OS of the device in which case the data collected by tracking the internal events becomes untrustworthy.
[9]	Proposes a smart anti-malware that can shift between different security levels according to the assets value and the battery status of the resource constrained device.	Switching between different levels of detection allows preservation of battery.	The anti-malware system can result in high false results with falling battery results.
[64]	A malicious software detection method based on power consumption. This method used Gaussian Mixture Model (GMM) built by using Mel frequency cepstral coefficients (MFCC).	Has a malware detection rate of 79%	The method has effective identification and detection of malicious software.
[23]	Exploits correlation between a user's location and power consumption pattern of his/her smartphone.	This method is capable of identifying some locations where location specific power consumption based detection technique can be used with high accuracy	There is need for further research into smartphone security primarily in the detection area

The anomaly detection techniques surveyed in the literature either have a high rate of false positives or use too much resources such as battery and CPU. This thesis proposes two approaches for anomaly detection that are based on the power consumption pattern of the smartphones and involve off device data collection and off device monitoring and hence are independent of the smartphone battery usage. The proposed model explained in Chapter 5 has a low false positive rate and hence is highly reliable.

2.4 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a statistical method for transforming an observed multidimensional random vector into components that are statistically as independent from each other as possible. One of the major challenges in anomaly detection using power measurements is the definition of features. This is overcome with the use of ICA.The code implemented in the software of a device and the code used by a malware are independent of each other. Therefore, we assume that the power consumed by the malware code is partially independent of the power consumed by the normal code of the device and hence can be used as a differentiator to provide contrast between a signal with no anomalous behavior and one with anomalous behavior. Further, ICA also aids in noise removal. The use of ICA is further explained in Section 4.1.2.

Chapter 3

Measurements and Analysis

This chapter describes the experiments conducted to acquire data for designing and validating the model proposed in this thesis. Section 3.1 details our experimental setup bench and data acquisition methodology respectively; and, Section 3.2 presents a brief analysis of the acquired data.

3.1 Experimental Setup

3.1.1 Monitoring Tool

The Monsoon Power Monitor, shown in Fig. 3.1, developed by Monsoon Solutions Inc., provides an effective solution for measuring power consumed by mobile devices. It can be used to measure the power consumption patterns of devices with removable batteries. PowerTool software is used to control the power supply provided by the hardware and is used by both, electrical engineers and software developers. A USB channel connects the monitor hardware to a computer and is used to transfer data. The power monitor can collect 5000 samples in a second and can provide a supply of up to 13.5V and 6A. The high rate of collection of data and a user-friendly GUI makes the monsoon power meter ideal for the model proposed in this thesis. Figure 3.2 shows the Graphical User Interface (GUI) of the PowerTool software.

Figure 3.1: Mobile Device Power Monitor

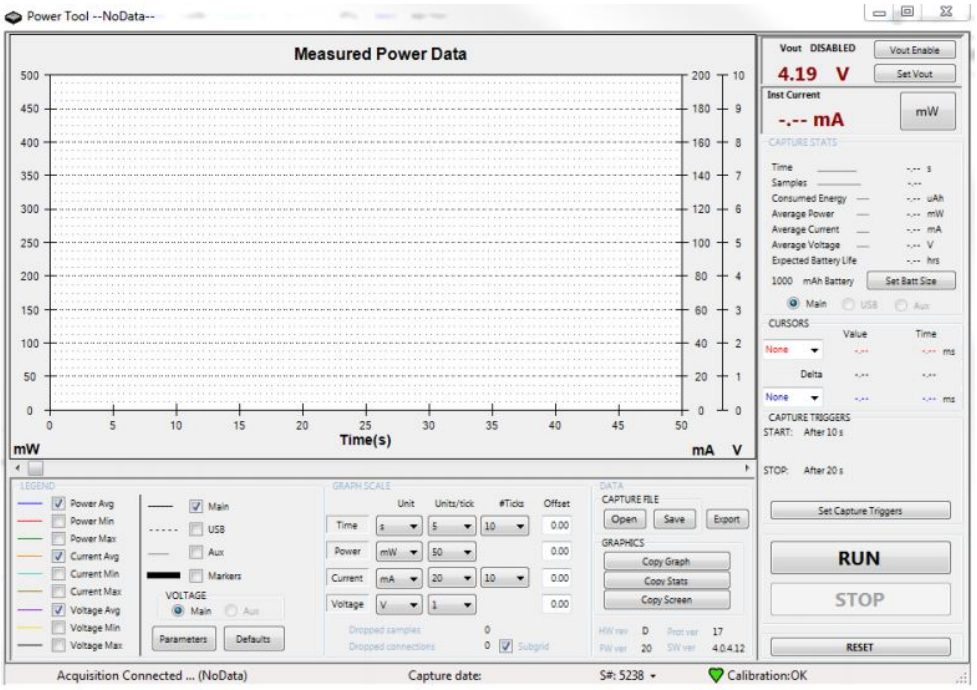


Figure 3.2: Graphical user interface (GUI) for the PowerTool software

3.1.2 Experimental Setup for detecting Malware using Power Signatures

The setup shown in Fig. 3.3 is used to measure the power consumed by a smartphone. The battery of the smartphone is bypassed as shown in Fig. 3.4 and the Monsoon Power Monitor is used to power the smartphone. The Monsoon Power Meter provides a constant power supply through the copper strips connected to the smartphone. The meter maintains a constant voltage as set by the user, and the resistance across the meter enables us to measure the current flowing through it and hence the power consumed by the smartphone is measured. An external laptop is connected to the meter through a USB cable for acquiring the power consumption data. The data acquisition is controlled using PowerTool, the GUI interface of the Monsoon Power Meter. The experiments were carried out with the smartphone, Samsung Galaxy S5 neo and the output voltage was set at 3.8V.

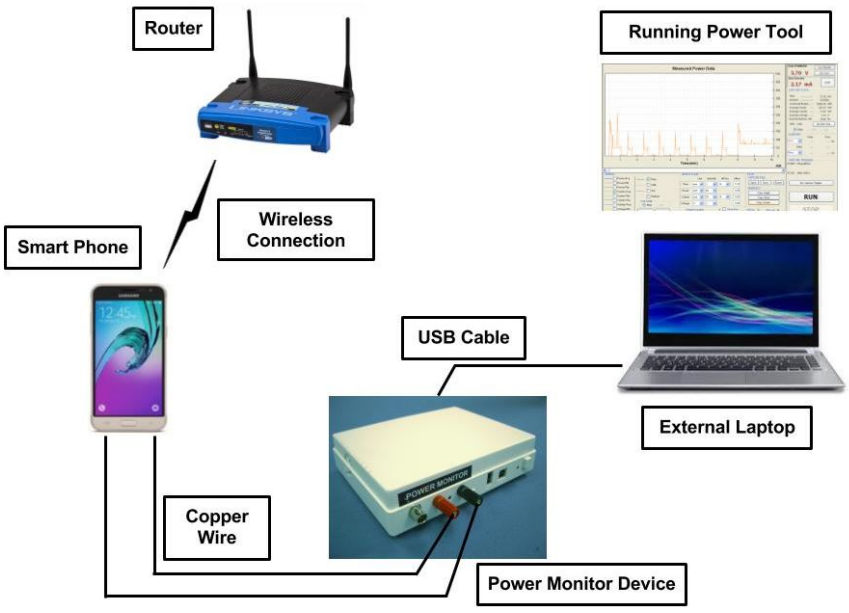


Figure 3.3: Experimental Setup for detecting Malware using Power Signatures

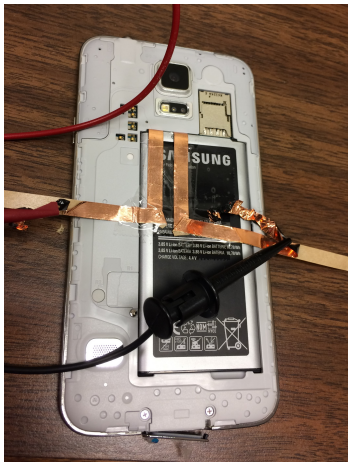


Figure 3.4: Bypassing the Battery

3.1.3 Background (Malware) App Emulating Anomalous Behavior

We have developed an Android app that runs in the background emulating anomalous behavior in smartphone. This background app alternates between actively downloading files and being dormant, based on a given duty cycle (Dut). The analogy is that an anomalous behavior is due to an undesired app in the smartphone that is dormant most of the time but becomes active at certain intervals. This behavior resembles the behavior of an adware. According to GDATA [13], adware frequently hides in fake apps that are installed from sources other than the official app markets and repeatedly launches advertisements that can cause energy drain. The GUI interface of the App is shown in Figures. 3.5, 3.6 and 3.7.

A duty cycle is the fraction of one period in which a signal or system is active. Duty cycle is usually expressed as a percentage or a ratio and is calculated using the formula shown in Equation 3.1. We have kept $T_{Cycle} = T_{ON} + T_{OFF}$, constant at 60 seconds, for all the duty cycles.

$$Dut = \frac{T_{ON}}{T_{ON} + T_{OFF}} \tag{3.1}$$

The Malware test bench app has been programmed as follows:

- When the button indicating the Duty Cycle is clicked, the app waits in a for loop for

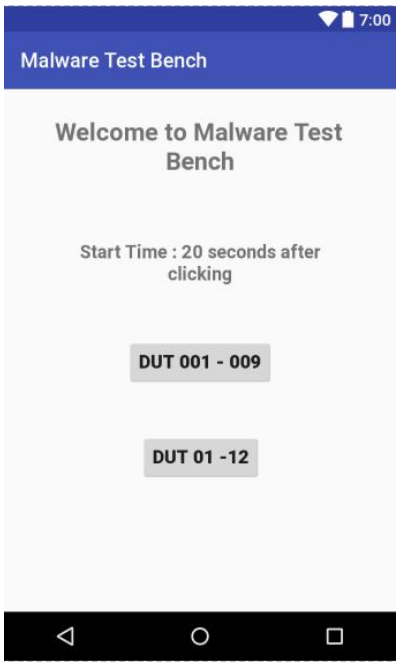


Figure 3.5: The Main Screen of the App

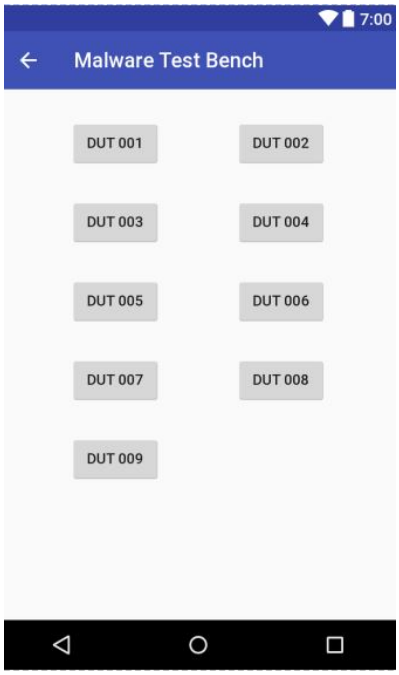


Figure 3.6: Duty Cycles 0.1% to 0.9%

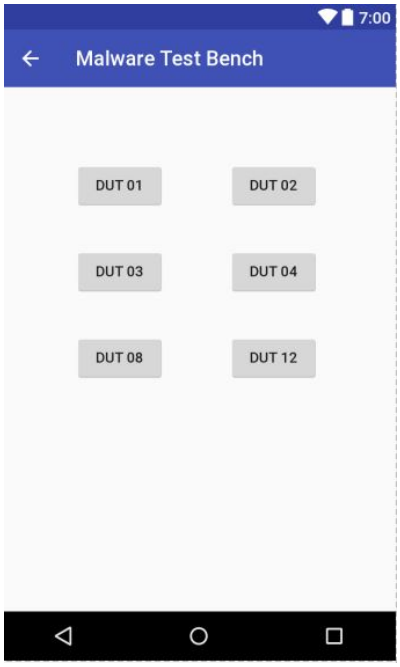


Figure 3.7: Duty Cycles 1% to 12%

a fixed duration of 20 seconds.

- After the wait time is over, the startService( ) method is called and the service remains active, downloading files for T_{ON} seconds.
- The stopService( ) method is called. and the service remains dormant for T_{OFF} seconds.
- After T_{OFF} seconds, the startService() method, the sequence continues in an infinite loop.

The power consumption pattern of the Malware Test Bench App with duty cycle 12%, shown in Fig. 3.8, clearly illustrates the cyclic behavior of the background app including the T_{ON} and T_{OFF} times. The first 20 seconds in Fig. 3.8 correspond to a for loop in the activity class of the app and the rest of the power trace is the power consumed by the service class. It is interesting to note that for the same piece of code the power consumption is fairly consistent. Therefore, to define the normal behavior of the device, thresholding is required which is explained in Chapter. 5 and Chapter. 6.

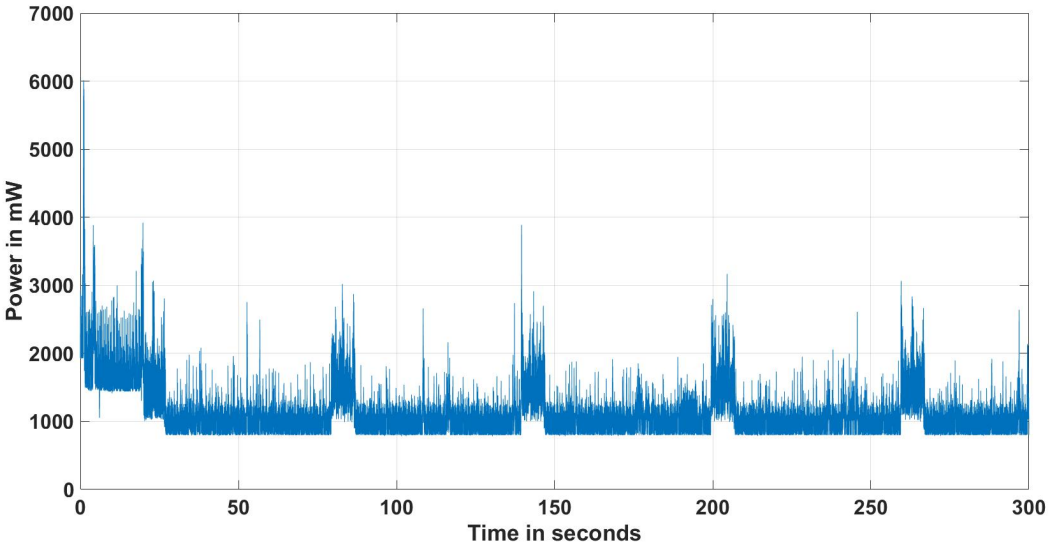


Figure 3.8: Power trace of the Malware Test Bench with Duty Cycle = 12%

The lifecycle of the background service is shown in Fig. 3.9.

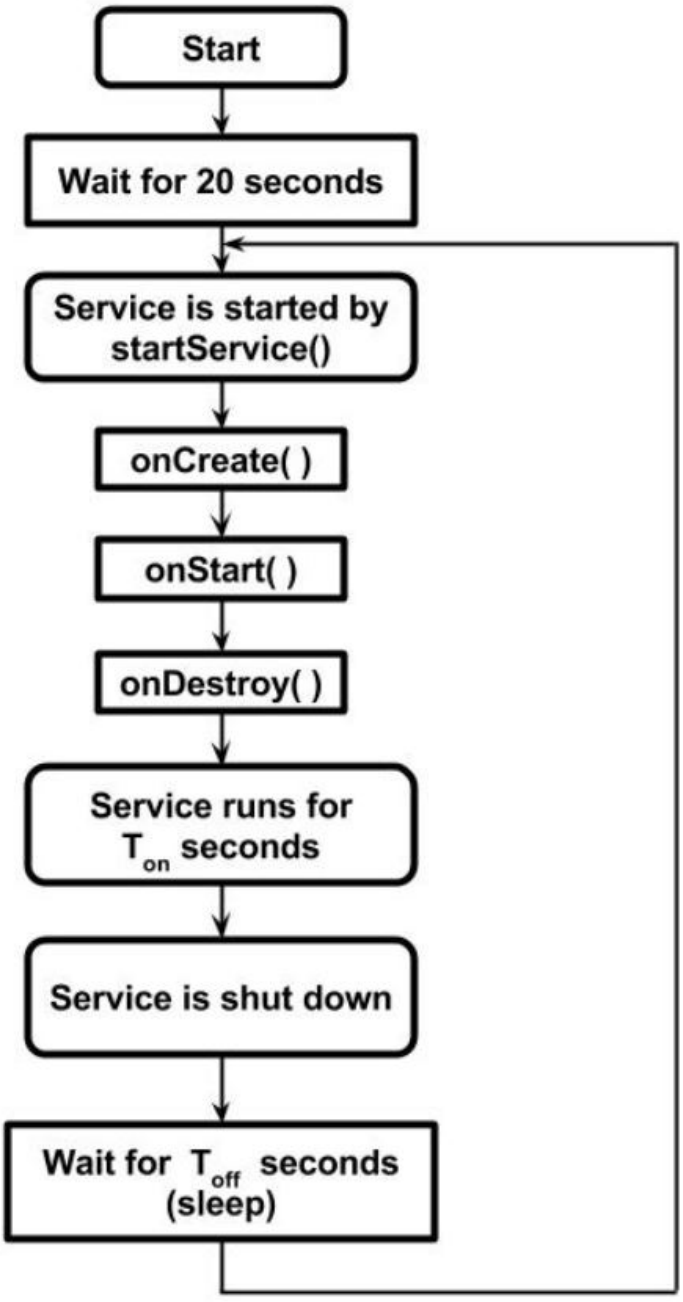


Figure 3.9: Life cycle diagram of the background service

3.2 Experiments and Preliminary Analysis

This section details the experiments conducted to acquire power consumption data and presents a preliminary analysis of the acquired data. We use the test bench shown in Fig. 3.3 to measure the power consumed by a smartphone in varying test conditions. All the experiments use YouTube as the foreground app.

3.2.1 Experiments in a Trusted Environment

Experiments were conducted to measure and analyze the power consumption pattern of a smartphone when Youtube App is used. These experiments were conducted in a trusted environment i,e., there was no anomalous behavior or malware in the phone. The settings of the smartphone were kept the same throughout the experiments to maintain consistency of data. The following sequence of steps are repeated on the YouTube App and the power consumed by the smartphone is measured.

- Launch the YouTube App
- Search for a particular video
- Select the video and let it play.

The duration of each run is 300 seconds. This sequence was repeated 45 times in sets of 15 iterations, spread over a duration of 2 months. A preliminary analysis of the data acquired in presented in 3.2.3

3.2.2 Experiments in the Presence of Anomalous Behavior

The next set of experiments were conducted in the presence of anomalous behavior i.e., with the malware test bench app running in the background. The sequence of steps followed are:

- Launch the malware test bench app.
- Press the button corresponding to the desired duty cycle.
- Press the home button so that the app keeps running in the background.

- The start time for the background service has been set to 20 seconds. Hence wait for 20 seconds from the time the button corresponding to the duty cycle was pressed.
- Launch the YouTube App. The launch should coincide with the start of the background service.
- Search for a particular video
- Select the video and let it play.

The duration of each run is again 300 seconds. The duration is kept the same to provide for accurate validation of the model. Experiments were conducted with the anomalous behavior having duty cycles 1%, 2%, 3%, 4%, 8% and 12%. The reason for choosing low duty cycles is that we are interested in detecting anomalous behavior that is active for short intervals of time, and which would otherwise be difficult to detect by analyzing power consumption of the smartphone. An example of the power consumption pattern of the smartphone with anomalous behavior of differing duty cycle are shown in Fig. 3.10. It is evident that detecting anomalous behavior with lower activity period from visual analysis alone is difficult.

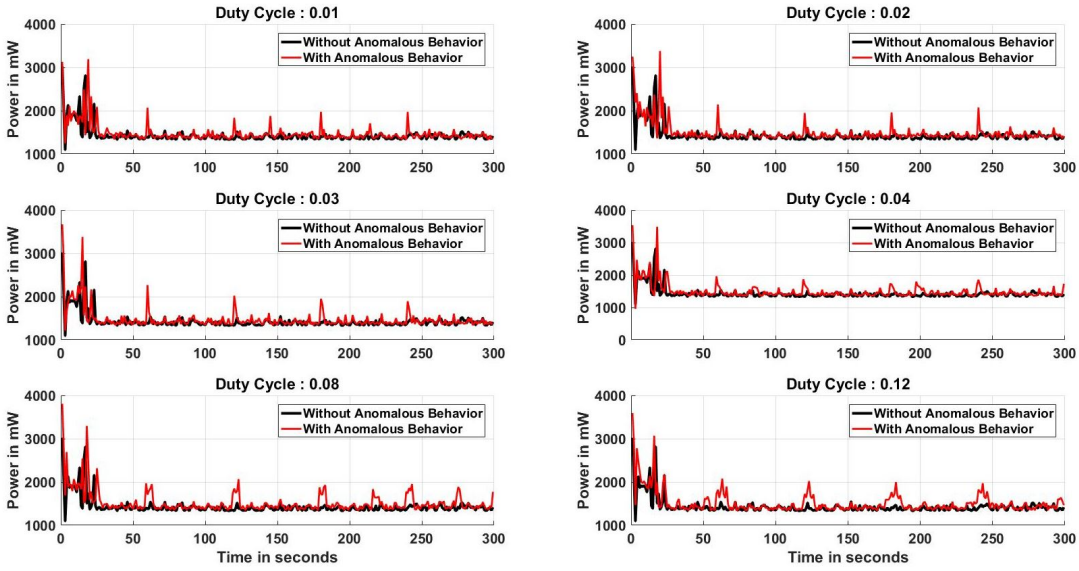


Figure 3.10: Power consumed by the smartphone in different anomaly conditions

3.2.3 Preliminary Analysis of the Data

This subsection presents a preliminary analysis of the power consumption data acquired through the experiments conducted.

Analysis of the Features of the Signals

The power consumed by the smartphone in the presence of anomalous behavior would be the sum of the power consumed by the smartphone in the absence of the background service, and the power consumed by the background service in isolation. Hence, theoretically, there should be a clear distinction between the two cases when common features such as minimum value, maximum value, mean value and standard deviation of the signals are analyzed. 15 readings of duration 300 seconds with 5000 samples per second, were taken for each duty cycle of malware and three sets of 15 signals each were taken for the no malware scenario, with each set being 1 month apart to show the variability in the behavior of the smartphone. Signal # denotes the signal number where # ranges from 1 to 15. However, as seen in Fig. 3.11, the power consumption is dynamic. Even though the standard deviation of the signals is not as dynamic as the other features, we observe that the readings in red in Fig. 3.11d, show more deviation than the signals in blue or red. The values in red are the standard deviation of the power consumed by the smartphone in the absence of anomalous behavior, measured 2 months after the initial experiments began. Hence, with ageing, the power consumption pattern of the smartphone shows more deviation. Therefore, standard deviation alone cannot be used to detect malware. Moreover, standard deviation cannot be used if the foreground app has high variance in its power consumption.

Fig. 3.12 shows a comparison of the minimum value in the signals, Fig. 3.13 shows the maximum values, Fig. 3.14 and Fig. 3.15. On observing the figures, it is evident that the energy consumed over a period of time in the presence of malware is sometime lower than the energy consumed in its absence. Analysis of the signal structure and spectral analysis might yield better results.

When we analyze the standard deviation of the signals, there appears a slight distinction between a signal exhibiting anomalous behavior and a signal that does not exhibit anomalous behavior as seen in Fig.3.15 However, this is attributed to the nature of power consumption of the Youtube app and might not hold for an app that is more dynamic in its power consumption. For this reason, we will use cross-correlation coefficient in the models developed in Chapter 4 and Chapter 5, which is similar to standard deviation in its formula but, it compares every point in the signal with every point of another signal, thus facilitating comparison with historical data.

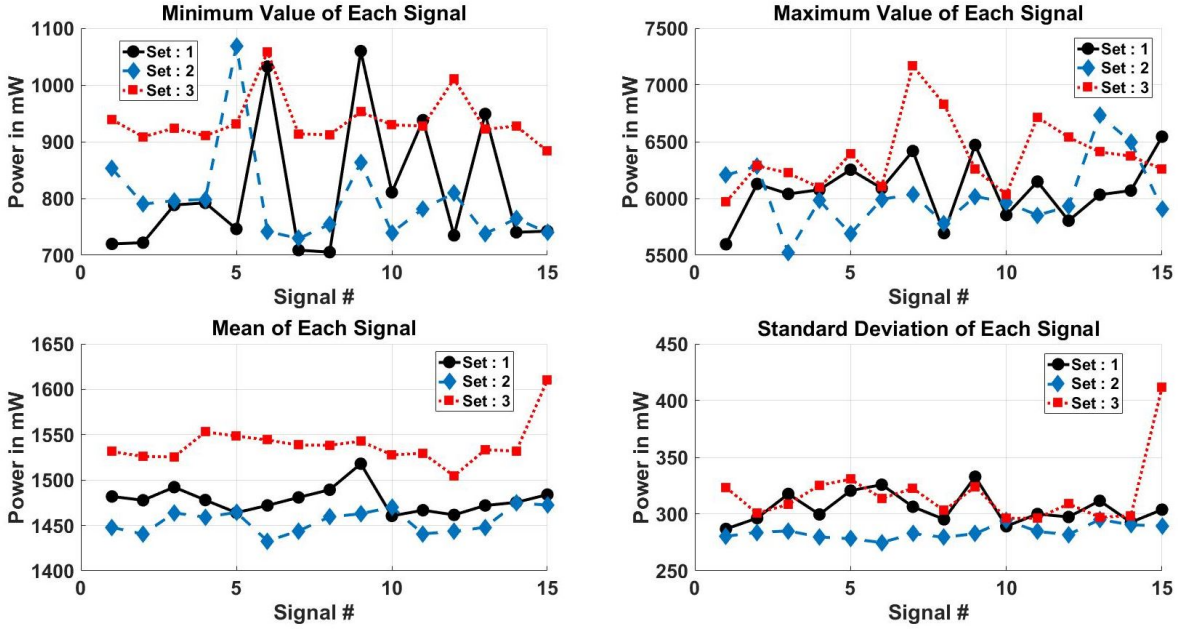


Figure 3.11: Features of No malware

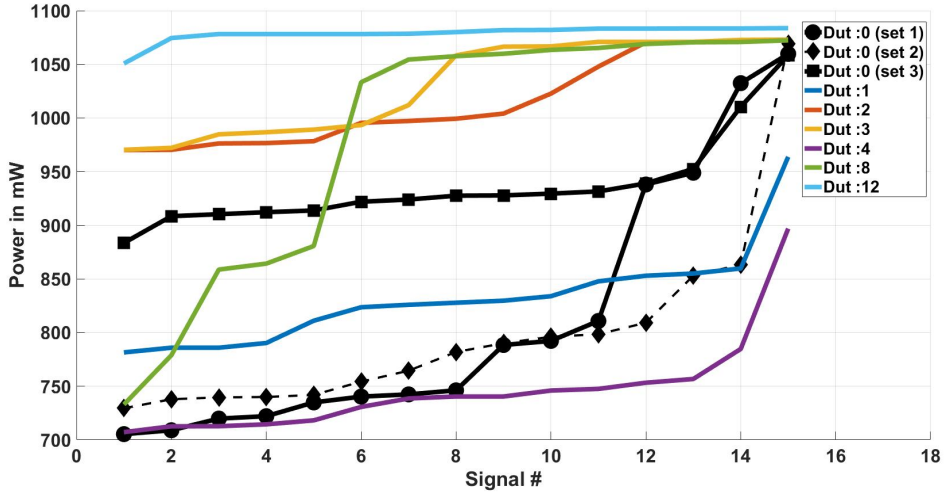


Figure 3.12: Minimum value of the Signals - sorted in ascending order

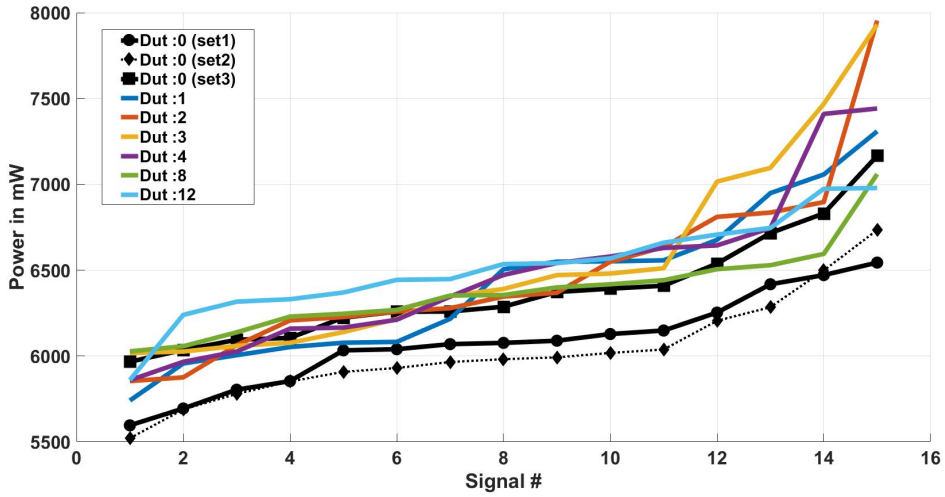


Figure 3.13: Maximum value of the signals - sorted in ascending order

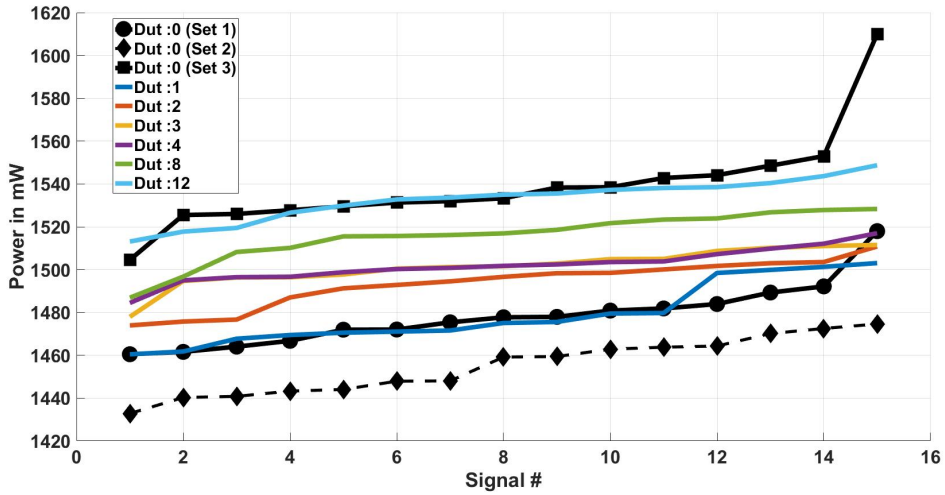


Figure 3.14: Mean value of the Signals - sorted in ascending order

3.2.4 Analysis of the Cyclic Behavior of the Signals

In this subsection we analyze the cyclic behavior of power consumption of the smartphone. Every application in a smartphone is a software program which has a fixed set of code unless there is an update, and this code runs in loops and functions. Hence, theoretically the

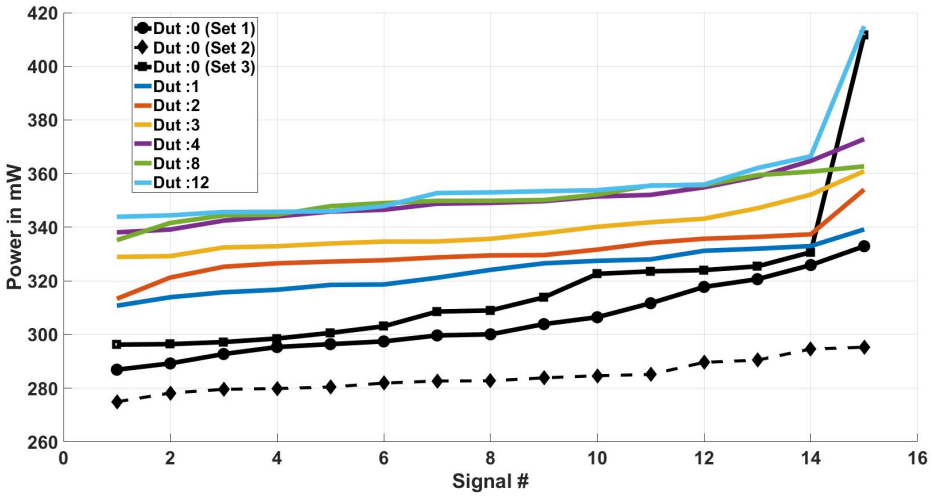


Figure 3.15: Standard Deviation of the Signals - sorted in ascending order

power consumption of the phones should be cyclic in nature. Each signal with a duration of 300 seconds contains 1499022 samples of data,as 5000 power samples are collected per second. The number of unique values and their respective values, in each signal was found. On analysis, it is observed that the number of unique values of power readings per signal is uniform for most of the signals both in the presence and absence of anomalous behavior and their number ranges from 20000 to 30000, which is significalty lower than 1499002, the number of samples per signal. This can be seen in Fig. 3.16.

Further, the unique values in each signal is also consistent and shows a distinct pattern. This leads us to the conclusion that the underlying power consumption of the smartphone is consistent over a period of time and the changes observed are due to the superposition of the power consumed by the anomalous behavior i.e., the background app. It is however interesting to note that the pattern for anomalous behavior and no anomalous behavior is not distinct as shown in Fig. 3.17. Hence, the power consumption of the YouTube app is dynamic to some extent inspite of having a comparatively lower standard deviation.

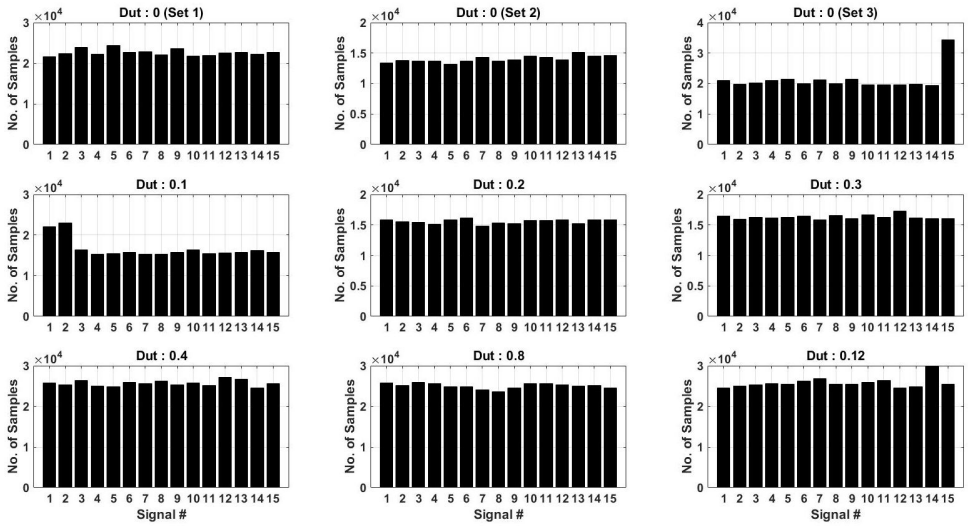


Figure 3.16: Number of Unique values in each signal

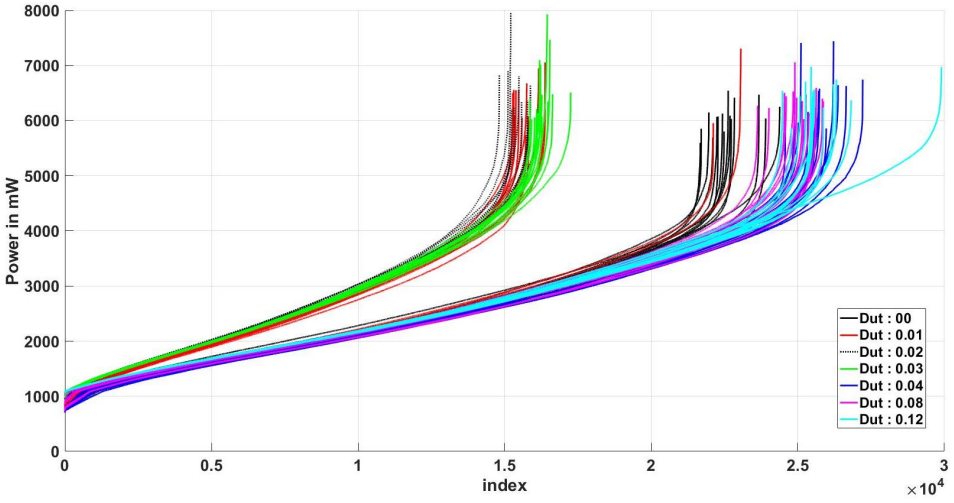


Figure 3.17: Unique values in each signal

Spectral Analysis

Histogram Analysis

We analyzed the distribution of power consumed under different scenarios. It was observed that the distribution remained consistent through all the scenarios, but an increase in the activity of anomalous did cause subtle variations in the distribution. The background app and the Youtube app are independent of each other and hence theoretically, the power consumed by them can be separated through Blind Source Separation. Blind source separation depends on the distributions of the independent components. Since the distributions are not very distinct as seen in Fig. 3.18, we will use a different approach to Blind Source Separation as explained in 4.1.2

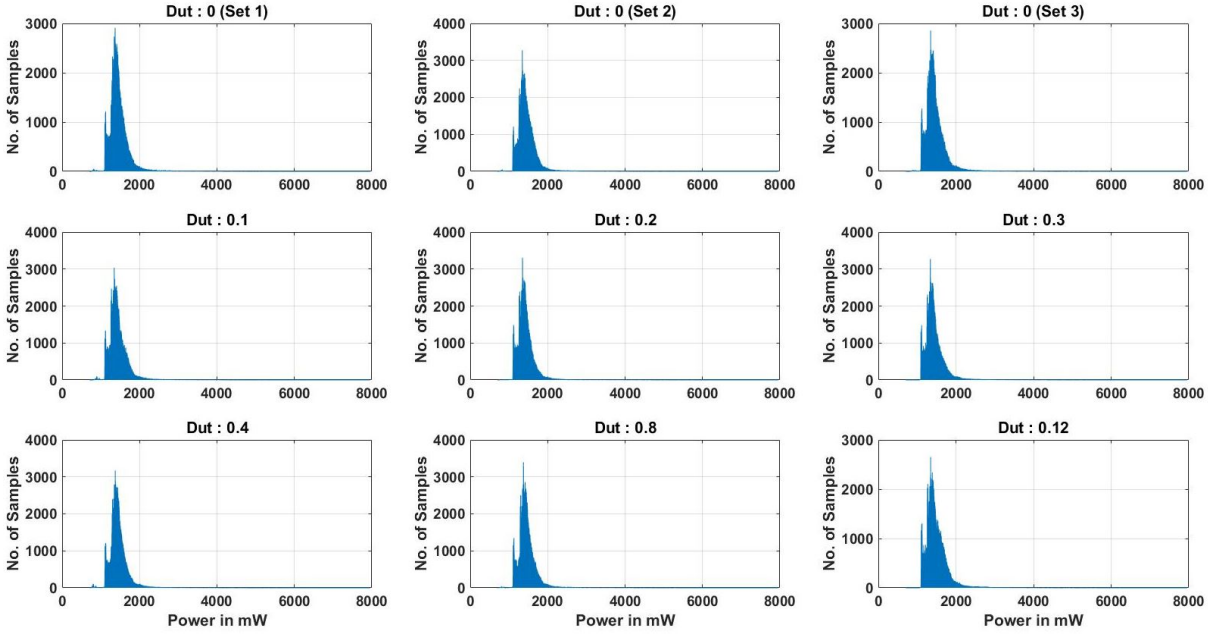


Figure 3.18: Histogram plot

Chapter 4

Detection of Anomalous Behavior of Smartphones Using Independent Component Analysis

This chapter describes the methodology we have developed for the detection of anomalous behavior in smartphones using a Blind Source separation technique known as Independent Component Analysis.

4.1 Methodology

We use the test bench shown in Fig. 3.3 to measure the power consumed by a smartphone in varying test configurations. The methodology used in detecting malware has been illustrated in Fig. 4.1. Power consumed by the smartphone when it is idle but with the interfaces switched ON is called no load power, and the signal derived during this condition is called Base Signal, $BS(t)$. The power consumed by the smartphone when an app is used would be the sum of the power consumed by the app and the Base Signal $BS(t)$ as seen in Fig. 4.2 and this signal is denoted by $RS_r(t)$ $(1 \leq r \leq n)$, where n is the total number of scenarios for a particular app. The power consumed by the smartphone is measured for different apps across various scenarios and are stored as $RS_1(t)$, $RS_2(t)$, ..., $RS_n(t)$. These signals will form the reference for the detection of malware when the same apps are used in future. Both $RS_r(t)$ and $BS(t)$ are recorded in a closed, controlled and trusted environment. In this thesis, we have used the methodology explained in Section 3.2.1 to

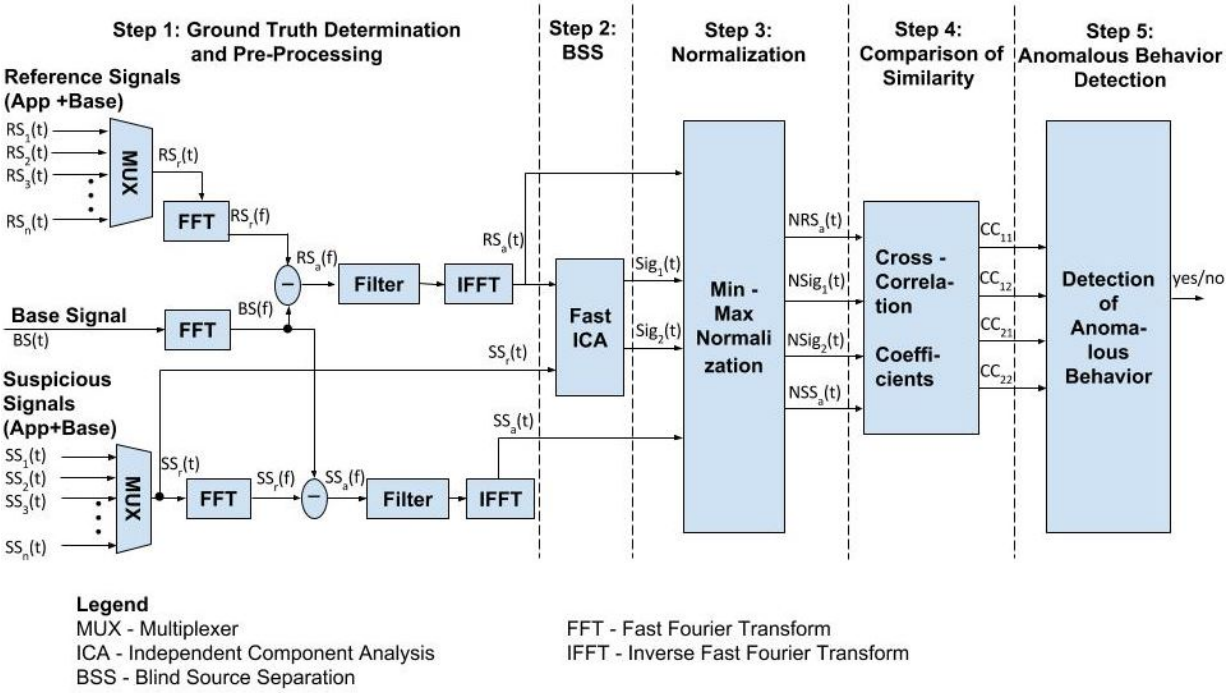


Figure 4.1: The five steps of the methodology

measure $RS(t)$. When the smartphone is tested for presence of malware as explained in Section 3.2.2, the power consumed by the smartphone is measured and the signal thus obtained is called suspicious signal $SS_r(t)$. This signal is provided as input to the model, and based on the results obtained it is established if the smartphone has a malware. The five steps of the methodology are as follows.

Step 1: Ground Truth Determination and Pre-Processing

Ground truth determination is done with the help of the power signals measured in the trusted environment. Fast Fourier Transform (FFT), a low pass Infinite impulse response (IIR) filter and Inverse Fast Fourier Transform ($IFFT$) are used for this. The inputs for this step are $RS_r(t)$, $SS_r(t)$ and $BS(t)$, and the outputs are $RS_a(t)$, $SS_a(t)$ and $SS_r(t)$ as shown in Fig. 4.1.

Step 2: Blind Source Separation (BSS)

Signals $RS_a(t)$ and $SS_r(t)$ are given as inputs to the FastICA algorithm [35], which separates two independent components from input signals and acts as a differentiator. Outputs after Independent Component Analysis (ICA), $Sig_1(t)$ and $Sig_2(t)$, are

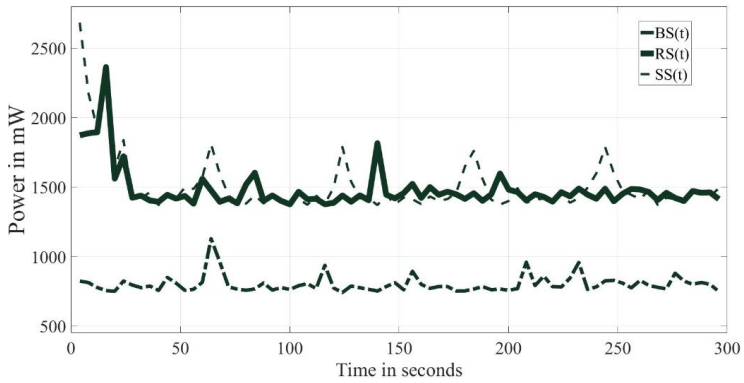


Figure 4.2: Power consumption pattern

power signatures of the individual apps that are part of $RS_a(t)$ and $SS_r(t)$ as shown in Equation. 4.1.

$$\begin{bmatrix} RS_a(t) \\ SS_r(t) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} Sig_1(t) \\ Sig_2(t) \end{bmatrix} \tag{4.1}$$

where, a_{11}, a_{12}, a_{21}, and, a_{22} depend on the intensity of the two signals.

Step 3: Normalization
Independent components, $Sig_1(t)$ and $Sig_2(t)$, obtained as outputs of FastICA, the reference signal $RS_a(t)$ and the suspicious signal $SS_a(t)$ are normalized to a scale of [0,1] based on Eq. 4.2, to remove error due to scaling.

$$NRS_a(t) = \frac{(RS_a(t) - Min(RS_a(t)))}{(Max(RS_a(t)) - Min(RS_a(t)))} \tag{4.2}$$

Step 4: Comparison of Similarity
The normalized signals $NSig_1(t)$, $NSig_2(t)$, $NRS_a(t)$ and $NSS_a(t)$ are then compared for similarity by calculating the cross correlation for each combination.

Step 5: Detection of Anomalous Behavior
Thresholds for each combination of reference signals $RS_r(t)$, have been established through prior tests and are used to determine if the smartphone has anomalous behavior.

4.1.1 Ground Truth Determination and Pre-Processing

The most important step for any signal extraction process is the determination of ground truth. In this experiment, power consumed by the smartphone, when a given app is active, is recorded in a closed, controlled and trusted environment. This is repeated for all the apps under consideration and the signals are stored as $RS_1(t)$, $RS_2(t)$, ..., $RS_n(t)$, where RS stands for Reference Signal. We assume that the requirements of all apps in terms of the background processes and the interface are under similar conditions. Power consumed by the smartphone when no app is active constitutes the power consumed under no load and is called the Base Signal $BS(t)$. When the smart phone is exposed to the external network, unknown background applications might get triggered which result in an increase in the power consumption of the smartphone. The known apps are run in the same sequence as before and the power consumed are recorded as $SS_1(t)$, $SS_2(t)$, ..., $SS_n(t)$, where SS is an abbreviation for Suspicious Signal. Our objective is to determine if these signals contain power consumed by an unknown application. The trusted power signal for one of the apps, referred to as $RS_r(t)$ $(1 \leq r \leq n)$ in Fig. 4.1, is chosen using a multiplexer and the corresponding suspicious signal, $SS_r(t)$, is also chosen using another multiplexer. The signals $RS_r(t)$, $SS_r(t)$, and $BS(t)$ are in the time domain and hence are converted to the frequency domain using Fast Fourier Transform as per Equation (4.3). The resulting signals $RS_r(f)$, $SS_r(f)$ and $BS(f)$ contain the mean at 0 Hz and the variances at the respective frequencies.

$$X(f) = \sum_{t=0}^{l-1} x(t)\, e^{-\frac{2\pi i}{l} tf} \tag{4.3}$$

where l represents the length of the signal. The reference signal and the suspicious signal both have an additive component in addition to the base signal. Hence $BS(f)$ is subtracted from $RS_r(f)$ and $SS_r(f)$ to obtain $RS_a(f)$ and $SS_a(f)$, respectively. $RS_a(f)$ is the frequency domain signal of the power consumed by just the app in the trusted environment; similarly, $SS_a(f)$ is the frequency domain signal of the power consumed by just the app in the suspicious environment. These signals are converted back to the time domain using Inverse Fast Fourier Transform (IFFT) as per Equation (4.4) to obtain the power consumed by the app alone in the trusted environment, $RS_a(t)$ and the power consumed by the app alone in the suspicious environment, $SS_a(t)$. The signal $RS_a(t)$ serves as the ground truth.

$$x(n) = \sum_{n=0}^{l-1} X(f) e^{\frac{2\pi}{l} nf} \tag{4.4}$$

4.1.2 Blind Source Separation (BSS)

This step involves the separation of the two signals, the reference signal $RS_r(t)$ for the app and suspicious signal $SS_r(t)$ for the app, into the independent components $Sig_1(t)$ and $Sig_2(t)$. The signals $RS_r(t)$ and $SS_r(t)$ are the power consumed by the smart phone when the app is active in a trusted environment and a suspicious environment, respectively. However, these signals contain the base signal plus the power consumed by the app and can be considered as a linear combination as shown in Equations (4.5), (4.6) and (4.7). This can also seen in Fig. 4.3. The removal of $BS(t)$ from $RS(t)$ provides enough contrast to implement BSS as a differentiator.

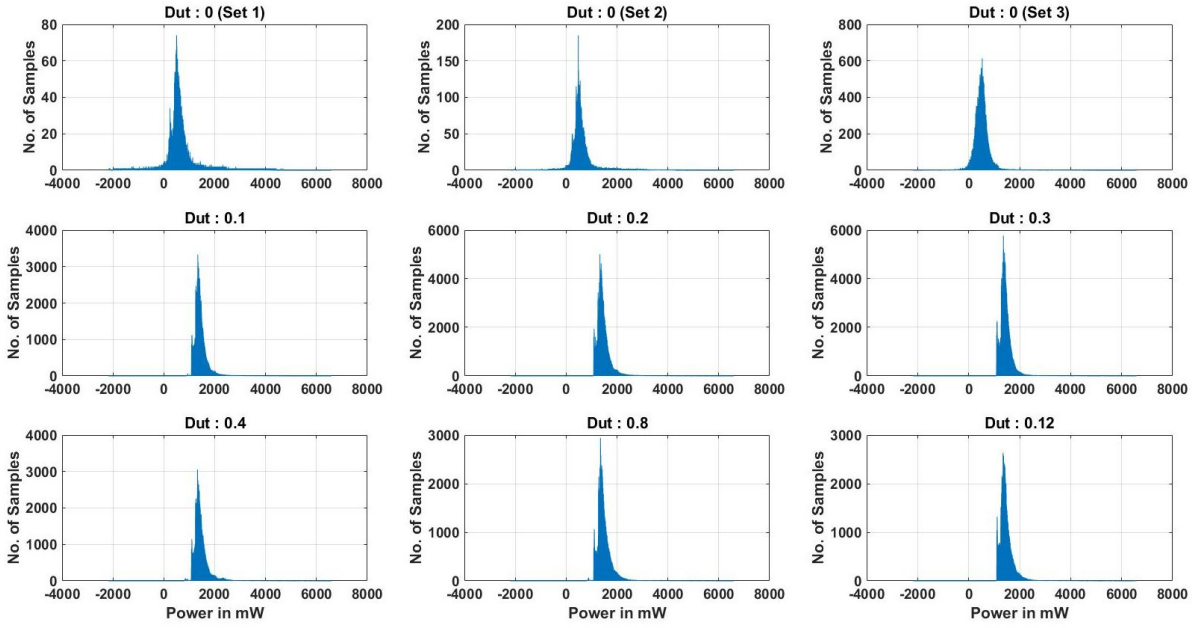


Figure 4.3: Histogram plot after removing the Base Signal from Reference Signal

$$RS_r(t) = a_{11} \times BS(t) + a_{12} \times RS_a(t) \tag{4.5}$$

$$SS_r(t) = a_{21} \times BS(t) + a_{22} \times SS_a(t) \tag{4.6}$$

$$\begin{bmatrix} RS_r(t) \\ SS_r(t) \end{bmatrix} = A \times \begin{bmatrix} BS(t) & RS_a(t) \\ BS(t) & SS_a(t) \end{bmatrix} \tag{4.7}$$

$$where, A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

The power consumed by the application in any given environment is independent of the power consumed by the smartphone under no load, since the two functionalities are independent of each other. FastICA, a Blind Source Separation (BSS) technique [35] used to find the independent power signals, $Sig_1(t)$ and $Sig_2(t)$, which are the power signatures of the same app alone in the trusted environment and in the suspicious environment, respectively. FastICA maximizes the mutual information within the samples of an independent component while minimizing the mutual information between the independent components. We use FastICA to estimate A, the mixing matrix and hence W, the demixing matrix to separate the input into a set of source signals from a set of mixed signals, without the aid of information (or with very little information) about the source signals or the mixing process.

4.1.3 Normalization

The Reference Signal $RS_a(t)$, Suspicious Signal $SS_a(t)$, and the independent components obtained from FastICA $Sig_1(t)$ and $Sig_2(t)$ are normalized to a scale of [0,1] based on Equation 4.8, to remove ambiguity due to scaling and provide for more accurate comparison. The outputs $NRS_a(t)$, $NSS_a(t)$, $NSig_1(t)$ and $Sig_2(t)$ are the respective similarly computed normalized signals.

$$NRS_a(t) = \frac{(RS_a(t) - Min(RS_a(t)))}{(Max(RS_a(t)) - Min(RS_a(t)))} \tag{4.8}$$

4.1.4 Comparison of Similarity

The independent signals obtained from FastICA need to be compared to the reference signal to check for similarity. If the signals are similar, then we can safely assume that there is no anomalous behavior on the smartphone. The signals are compared in pairs as shown in Table 4.1, and the measure of the similarity is noted. The inputs are the normalized Reference Signal $NRS_a(t)$, the normalized Suspicious Signal $NSS_a(t)$, and the normalized independent components $NSig_1(t)$ and $NSig_2(t)$. We have four pairs and thus four outputs CC_{11}, CC_{12}, CC_{21}, CC_{22}, where CC stands for Cross-Correlation coefficient. The formula used for calculating the Cross-Correlation coefficient is:

Table 4.1: Comparison of Similarity

Cross-Correlation Coefficient	$NRS_a(t)$	$NSS_a(t)$
$NSig_1(t)$	CC_{11}	CC_{12}
$NSig_2(t)$	CC_{21}	CC_{22}

$$CC_{ij} = \frac{\sum_{t=1}^{l} \left(X_i\left(t\right) - \overline{X_i}\right)\left(X_j\left(t\right) - \overline{X_j}\right)}{\sqrt{\sum_{t=1}^{l} \left(X_i\left(t\right) - \overline{X_i}\right)^2 \sum_{t=1}^{l} \left(X_j\left(t\right) - \overline{X_j}\right)^2}} \tag{4.9}$$

where, X_i is one of $NSig_1(t)$ and $NSig_2(t)$, and $\overline{X_i}$ is their respective mean; X_j is one of $NRS_a(t)$ and $NSS_a(t)$, and $\overline{X_j}$ is their respective mean; and l is the length of the signals.

4.1.5 Detection of Anomalous Behavior

The reference signals are used for the detection of anomalous behavior when the same app is run in the future. The set of reference signals, $RS_1(t)$, $RS_2(t)$, ..., $RS_n(t)$ contain measurements of the power consumed by the smartphone when a particular app is run. These measurements are taken for different scenarios or functions for which the app can be used. The outputs of Independent Component Analysis (ICA), $Sig_1(t)$ and $Sig_2(t)$, are mutually independent. We need to determine if these signals are different because, the power consumed for similar functions of the smartphone are different for different points in time or because there is an anomalous behavior on the smartphone. Hence, $Sig_1(t)$ and $Sig_2(t)$ are compared with $SS_a(t)$ and $RS_a(t)$ after normalization, as shown in Table 4.1.

Different combinations of the reference signals, $RS_r(t)$ $(1 \leq r \leq n)$, were provided as inputs to the proposed model and the obtained values of CC_{ij} were analyzed. We observed that when neither of the inputs showed anomalous behavior, at least two of the values of CC_{ij} were greater than δ_{CC}, where δ_{CC} is the minimum extraction threshold value. This value will be used for detecting anomalous behavior. Every app has some dynamic component involved with regard to the power consumed for the same functions. Hence, for a particular reference signal which is not very similar to the suspicious signal, the value of CC_{ij} could be less then δ_{CC} even though the suspicious signal does not contain anomalous behavior. Sum_kCC is the number of values of CC_{ij} for a pair of signals, that are less δ_{CC}. θ_{CC} is the minimum number of values of CC_{ij} required to flag the suspicious signal. S is the number of combinations that have been flagged; and n is the total number of reference signals used for a particular suspicious signal. η is a threshold that is set depending on the app being considered and is the percentage of the combinations that have been flagged as suspicious. The value will be lower for an app that is more dynamic in its power consumption pattern. The detection procedure has been depicted as flowchart in Fig. 4.4

The steps to be followed for testing a smartphone for malware are as follows.

Step 1: Selection of the foreground app
The app used by the user when the smartphone is tested for malware is termed as the foreground app. When the app is selected, the proposed model is run against the corresponding reference signals, $RS_1(t)$, ..., $RS_n(t)$.

Step 2: Operations on the foreground app
The foreground app is used by the user for a fixed duration and the power consumed by the smartphone is measured and this will be the suspicious signal, $SS_r(t)$.

Step 3: Iterative evaluation
The proposed model is run for n iterations with one input, $SS_r(t)$ kept constant and the other input varying from $RS_1(t)$ to $RS_n(t)$. The corresponding values of CC_{ij} and Sum_kCC are calculated and stored.

Step 4: Anomalous Behavior detection
The number of CC_{ij} values greater than δ_{CC} are calculated for each run. If this value is less than θ_{CC}, then the value of S is incremented by 1.

If $\frac{S}{n} > \eta$, then, Potential anomalous behavior is detected

The detection procedure depends on the thresholds, δ_{CC}, θ_{CC} and η. The extracted independent components of an app with a more dynamic power consumption pattern, might

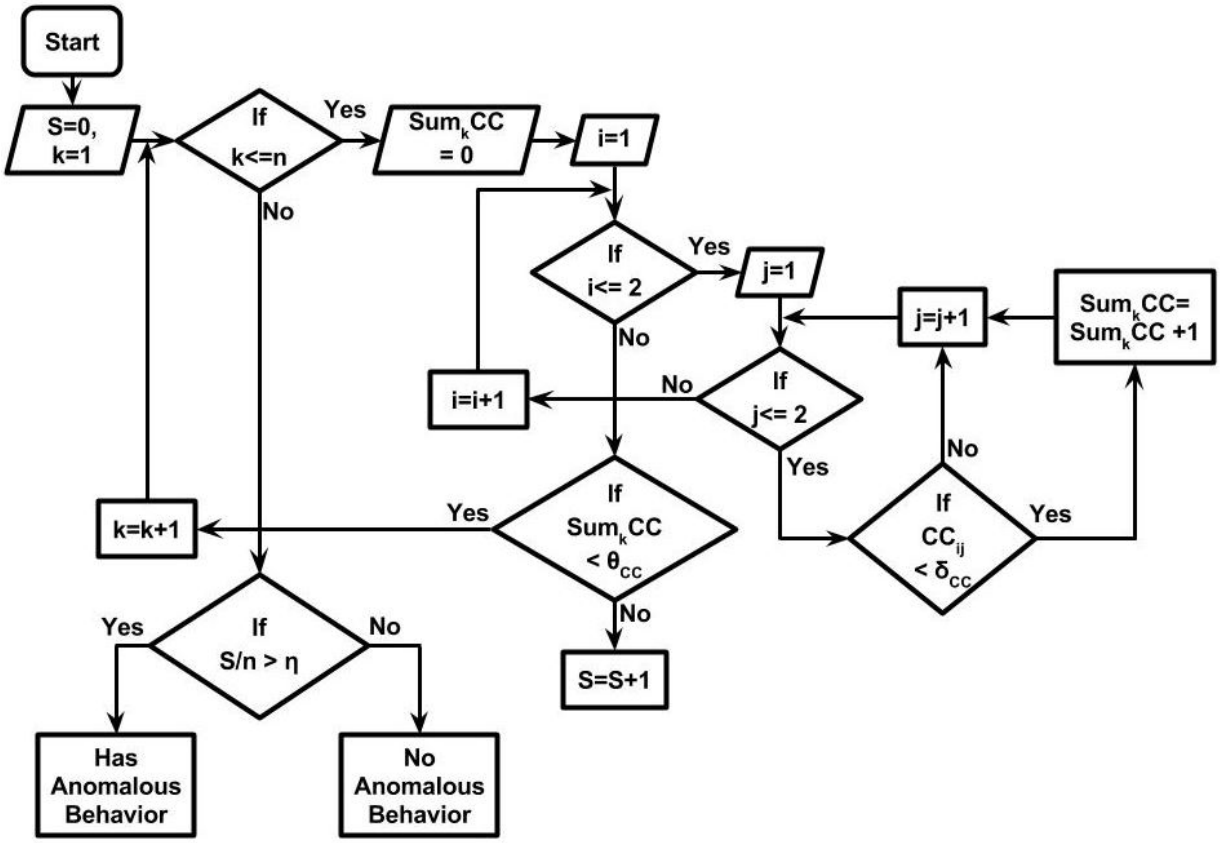


Figure 4.4: Detection of Anomalous Behavior

not be very similar to the Reference signals RS_n and the value of CC_{ij} is low even when the smartphone has no anomalous behavior. Hence, the values of δ_{CC} and θ_{CC} is lower for apps with dynamic power consumption pattern than the apps with a more predictable power consumption pattern. The value of η is the final decision boundary and is also selected based on the probability of the power consumption pattern being similar or dissimilar.

We conducted experiments with YouTube app. YouTube is not very dynamic in its power consumption pattern, for the same sequence of steps. Hence the value of δ_{CC} and θ_{CC} used were 0.89 and 2. The values were selected by training the model with a separate set of Reference Signals. The value of η was set at 0.2 because it was observed that 80% of Reference Signals had a similarity greater than δ_{CC} and θ_{CC}. Similarly, the values of δ_{CC} and θ_{CC} were selected when low pass filters were used in the model. The values of the thresholds used are shown in Table 4.2.

Table 4.2: Values of the Thresholds

Approach	δ_{CC}	θ_{CC}	η
No Filter	0.89	2	0.200
With $f_c = 625Hz$	0.89	2	0.133
With $f_c = 1250Hz$	0.90	2	0.266

4.2 Illustration of the Methodology

Let us look at an example that illustrates how the proposed detection methodology works. The length of the signals $RS_r(t)$, $SS_r(t)$ and $BS(t)$ is 5. Approximations of the signals are used in this illustration.

4.2.1 Ground Truth Determination and Pre-Processing

We obtain the following $RS_r(t)$, $SS_r(t)$ and $BS(t)$ vectors from measurements using Fig. 3.3.

$$RS_r(t) = [910.02 \quad 982.63 \quad 872.69 \quad 857.17 \quad 878.84]$$

$$SS_r(t) = [722.09 \quad 646.45 \quad 2039.63 \quad 2035.32 \quad 594.49]$$

$$BS(t) = [9.10 \quad 9.88 \quad 17.34 \quad 10.19 \quad 13.48]$$

The reference signal $RS_r(t)$, suspicious signal, $SS_r(t)$ and base signal $BS(t)$ are converted to the frequency domain using Fast Fourier Transform as shown in Equation (4.10).

$$\begin{aligned} RS_r(f) &= \sum_{t=0}^{4} RS_r(t)\, e^{-\frac{2\pi i}{l} tf} \\ SS_r(f) &= \sum_{t=0}^{4} SS_r(t)\, e^{-\frac{2\pi i}{l} tf} \\ BS(f) &= \sum_{t=0}^{4} BS(t)\, e^{-\frac{2\pi i}{l} tf} \end{aligned} \tag{4.10}$$

The spectral components of the Base Signal $BS(t)$ are subtracted from $RS_r(t)$ and $SS_r(t)$ as shown in Equations (4.11), to obtain the power consumed by just the app. A direct subtraction is possible because the signals are in the frequency domain and of the same length and hence have the same frequency index.

$$\begin{aligned} RS_a(f) &= RS_r(f) - BS(f) \\ SS_a(f) &= SS_r(f) - BS(f) \end{aligned} \tag{4.11}$$

The signals are now converted back to the time domain using Inverse Fast Fourier Transform as shown in Equations (4.12).

$$\begin{aligned} RS_a(t) &= \sum_{n=0}^{4} RS_a(f)\, e^{\frac{2\pi}{l} tf} \\ SS_a(t) &= \sum_{n=0}^{4} SS_a(f)\, e^{\frac{2\pi}{l} tf} \end{aligned} \tag{4.12}$$

$$\begin{aligned} RS_a(t) &= [900.92 \quad 972.75 \quad 855.35 \quad 846.98 \quad 865.36] \\ SS_a(t) &= [713.01 \quad 636.62 \quad 2022.31 \quad 2025.12 \quad 581.02] \end{aligned}$$

4.2.2 Blind Source Separation (BSS)

The signal $RS_r(t)$ is a linear combination of $BS(t)$ and $RS_a(t)$; similarly, $SS_r(t)$ is a linear combination of $BS(t)$ and $SS_a(t)$, as shown in Equation (4.13). The signals $BS(t)$ and $RS_a(t)$, and $BS(t)$ and $SS_a(t)$ are mixed according to a mixing matrix A to obtain the signals $RS_r(t)$ and $SS_r(t)$, respectively. The objective of using Blind Source Separation is to determine the mixing matrix A and hence calculate the de-mixing matrix W which is the pseudo-inverse of A.

$$\begin{bmatrix} RS_a(1) & \dots & RS_a(5) \\ SS_r(1) & \dots & SS_r(5) \end{bmatrix} = A \times \begin{bmatrix} Sig_1(1) & \dots & Sig_1(5) \\ Sig_2(1) & \dots & Sig_2(5) \end{bmatrix} \tag{4.13}$$

The independent components $Sig_1(t)$ and $Sig_2(t)$ are obtained by multiplying the de-mixing matrix W with the input mixed signals as shown in Equation (4.14).

$$\begin{bmatrix} Sig_1(1) & \dots & Sig_1(5) \\ Sig_2(1) & \dots & Sig_2(5) \end{bmatrix} = W \times \begin{bmatrix} RS_a(1) & \dots & RS_a(5) \\ SS_r(1) & \dots & SS_r(5) \end{bmatrix} \tag{4.14}$$

$$= \begin{bmatrix} -0.5999 & -0.4542 & -2.5318 & -2.5331 & -0.4299 \\ 27.3408 & 29.3569 & 27.8915 & 27.4351 & 26.2763 \end{bmatrix}$$

4.2.3 Normalization

$NRS_a(t)$ is computed from Equation (4.8). Similarly, $NSS_a(t)$, $NSig_1(t)$ and $NSig_2(t)$ are computed in a similar fashion by applying Equation (4.8). The values of $NRS_a(t)$, $NSS_a(t)$, $NSig_1(t)$ and $NSig_2(t)$ are given below.

$$\begin{aligned} NRS_a(t) &= [0.4289 \quad 1.0000 \quad 0.0666 \quad 0 \quad 0.1461] \\ NSS_r(t) &= [.0914 \quad 0.0385 \quad 0.9980 \quad 1.0000 \quad 0] \\ NSig_1(t) &= [0.9192 \quad 0.9885 \quad 0.0007 \quad 0 \quad 1.0000] \\ NSig_2(t) &= [0.3455 \quad 1.0000 \quad 0.5243 \quad 0.3762 \quad 0] \end{aligned}$$

4.2.4 Comparison of Similarity

The normalized signals are compared with the apriori data for similarity as shown in Table 4.3, which is an instance of Table 4.1. The entries of 4.3 are computed using Equation (4.9).

4.2.5 Detection of Anomalous Behavior

To detect an unknown app, one needs to evaluate the conditions explained in subsection 4.1.5.

4.3 Validation of the model, analysis and results

We ran the same sequence of steps on the YouTube app in a trusted environment and measured the power consumed by the smartphone. This was repeated 15 ($i.e., n = 15$) times

Table 4.3: Comparison of Similarity: An Example

Cross-Correlation Coefficient	$NRS_a(t)$	$NSS_a(t)$
$NSig_1(t)$	0.8275	0.0516
$NSig_2(t)$	0.8680	0.5524

and the duration of each run was 300 *seconds*. This will be considered as the reference signals $RS_1(t)$, ..., $RS_{15}(t)$. We developed an app that runs in the background emulating anomalous behavior. This background app alternates between actively downloading files and being dormant, based on a given duty cycle (Dut). The analogy is that an anomalous behavior implies an undesired app in the smartphone that is dormant most of the time but becomes active at certain intervals. The background app remains dormant and starts executing when the YouTube app is launched. The ON and OFF periods of the background app depends on the duty cycle we specify. The power consumed by this background app, considered as the power consumed by an anomalous behavior results in a change in the power consumption pattern of the smartphone as seen in Fig. 4.2. The proposed model makes use of this change in power consumption to detect the anaomalous behavior. This technique makes the assumption that the malware code does not monitor the tasks being performed by the smartphone and is independent of the nature of the device.
We conducted experiments with the anomalous behavior having duty cycles 0.1%, 0.2%, 0.3%, ..., 0.8%, 0.9%, 1%, 2%, 3%, 4%, 8% and 12%. We also consider two different cases of no anomalous behavior. The second set of 15 readings were taken two months after the first set to account for change in the behavior of the app over time. The reason for choosing low duty cycles is that we are interested in detecting anomalous behavior that is active for short intervals of time, and which would otherwise be difficult to detect by analyzing power consumption of the smartphone. The same sequence of steps as done to measure RS_n, were repeated on the YouTube app, but, with the anomalous behavior present, and the power was measured. These signals, denoted as $SS_n(t)$ and were provided as inputs to the model, and the accuracy of model was evaluated. Accuracy here is the number of correct identifications divided by the total number of signals. The results of the validation of the model are shown in Fig. **??**.

4.3.1 Analysis and Results

The experiments were done with the YouTube app and a custom designed app (Anomalous behavior) that we have developed for testing. The same sequence of steps were repeated and the power was measured. The YouTube app was run for a duration of 5 minutes and the measurements were taken at 5000 samples/sec. First, only the YouTube app was run and the same video played. This was repeated 15 times and the measurements were stored as $RS_1(t)$, ..., $RS_{15}(t)$, the reference signals.

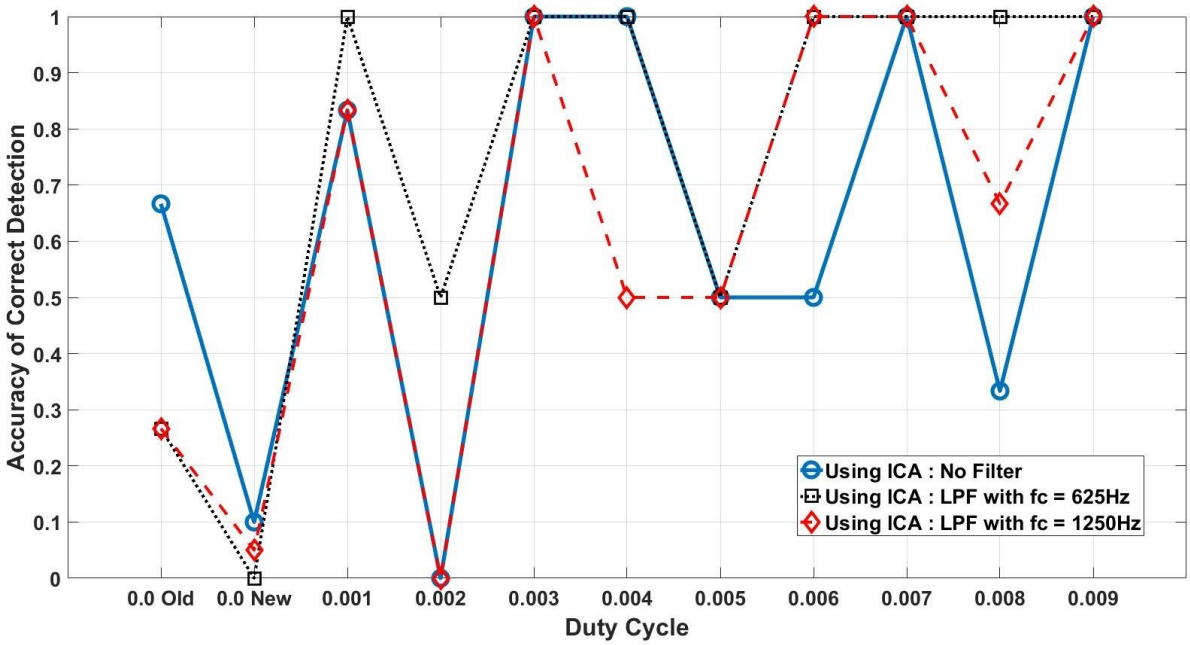


Figure 4.5: Accuracy in detecting malware with Lower Duty Cycles

Next, the same sequence of steps were repeated as before, but, with anomalous behavior of different duty cycles in the background. This was also repeated 15 times for each of the duty cycles. The measurements were stored as $SS_1(t)$,..., $SS_{15}(t)$, the suspicious signals.

Since we have 17 different duty cycles, we have a total of 225 suspicious signals. The proposed model was validated as described in Sections 4.1 and 4.3 without using a filter, with a low pass butterworth filter with cut-off frequency, $f_c = 625Hz$ and then with a low pass butterworth filter with cut-off frequency, $f_c = 1250Hz$. The accuracy of the proposed model was calculated for each of the duty cycles and for each of the validations. As seen in Fig. 4.5, the accuracy of the model for 0% Duty Cycle is 66.67% but reduces to 10%

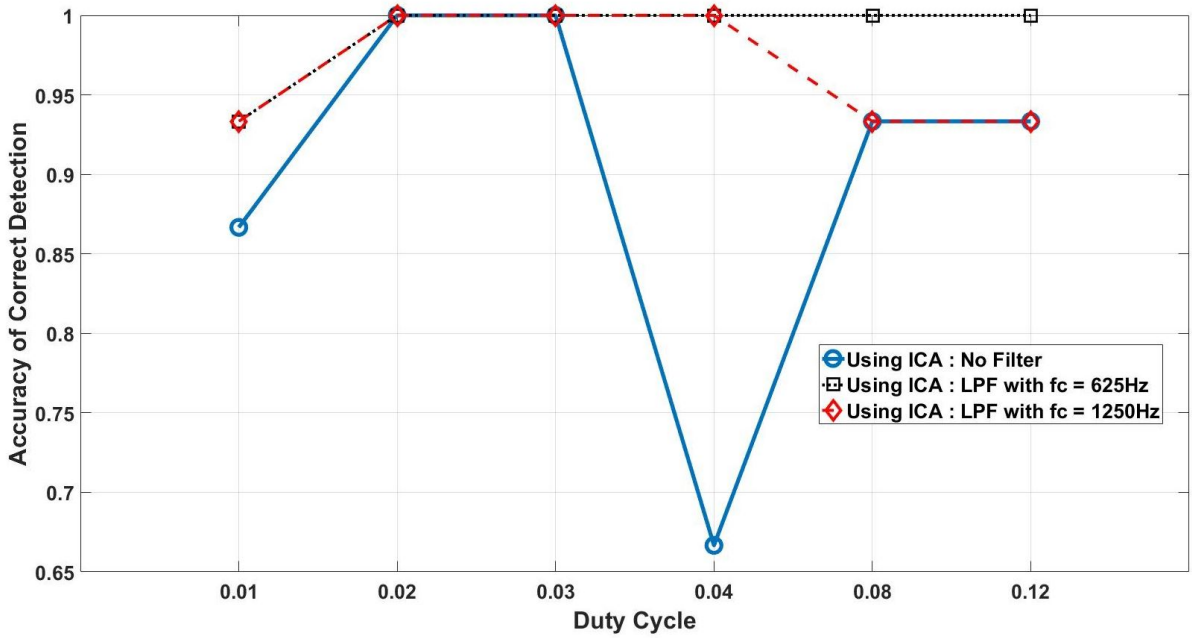


Figure 4.6: Accuracy in detecting malware with Higher Duty Cycles

for the new set of readings. Thus with time, the number of false positives increase. The model performs best for malware with duty cycle 2% and 3% and has an accuracy of 100%. The lowest accuracy is for the signals with malware with a duty cycle of 4%. This might be because the malware becomes active long enough to resemble the power consumption pattern of the YouTube app and hence ICA is not able to differentiate the signals properly. We observed that the use of a low pass filter resulted in better accuracy as seen in Fig. 4.5 and Fig. 4.6, however, it resulted in an increase in the rate of false positive. This could be because the high frequency components for the ground truths are eliminated by the low pass filter and only the low frequency components are compared.

Chapter 5

Detection of Anomalous Behavior based on Similarity Matrix

This chapter describes a new approach to detect anomalous behavior in smartphones based on the experiments conducted in Chapter 3 and validates the approach while presenting a comparision to the results obtained in Chapter 4

5.1 Introduction

The power consumption pattern of a smartphone is not deterministic and varies between certain ranges for the same sequence of steps. This makes it difficult to compare the current power consumption to one particular historical data and hence the current data has to be compared with a larger historical dataset. Further, the power consumed by a malware or an anomalous behavior would superimpose wit the general power consumption of the smartphone. Since the probability distributions cannot be accurately predicted, Independent component analysis is fairly limited as a differentiator as discussed in 4.1. In the analysis of the validation of the model proposed in Chapter 4, it was was observed that the use of a Low pass filter improved the accuracy for most of the anomalies. However, there was in increase in the number of false positives. When the detection procedure described in Section 4.1 was run with Cross-Correlation coefficient, we observed that we could achieve similar results even without the use of Independent Component Analysis. The signals were compared with the power signals measured in the trusted environment using the cross - correlation formula and as we can see in Fig. 5.1, there is some degree of

differentiation between the signals containing malware with differing duty cylcles, however there is no clear distinction between the no malware signals and those containing malware. When the lower frequencies of the signals are compared, we oberserve that the similarity of signals varies for different bands of frequencies.

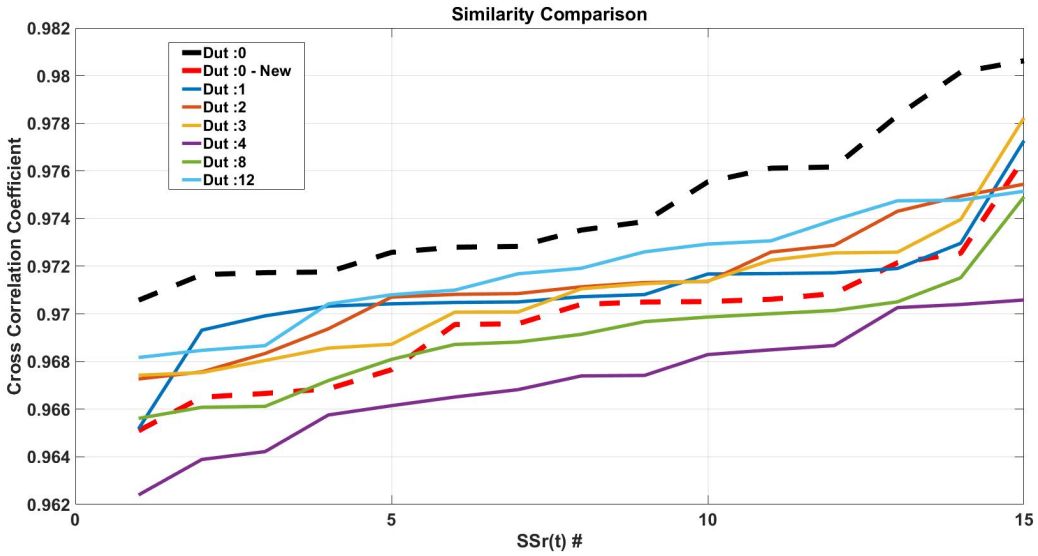
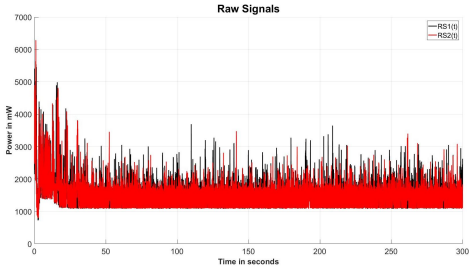
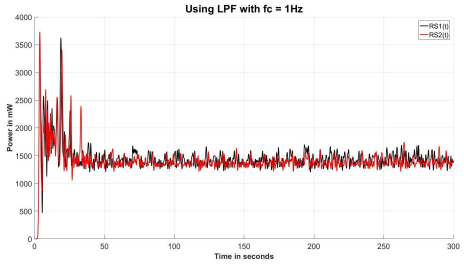


Figure 5.1: Similarity of signals using a single comparison

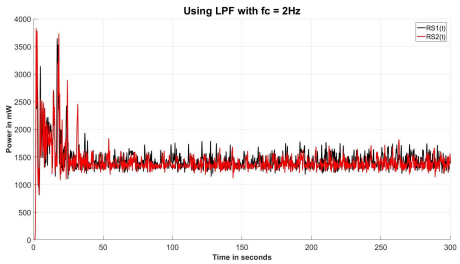
Based on these observations we proposed a new modified model that uses a similarity matrix to detect anomalous behavior. The similarity matrix consists of a matrix of thresholds obtaining through an array of low pass filters.



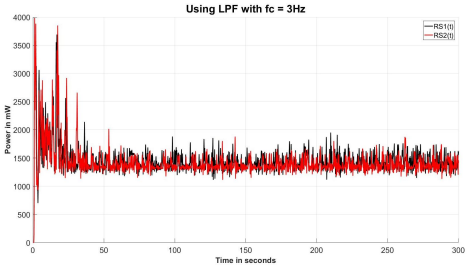
(a) CC = 0.9746



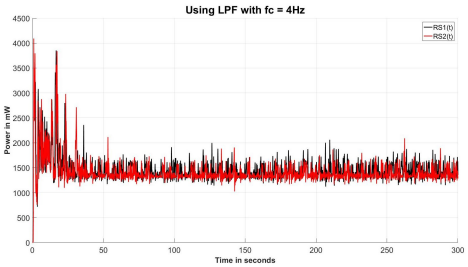
(b) CC = 0.9678



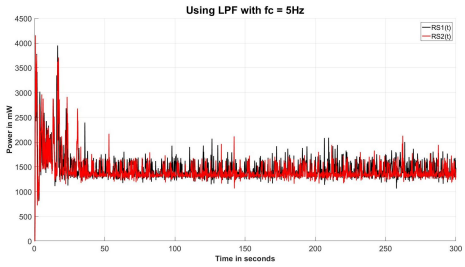
(c) CC = 0.9681



(d) CC = 0.9763



(e) CC = 0.9788



(f) CC = 0.9793

Figure 5.2: Illustration of similarity of signal in Lower Frequencies

5.2 Methodology

The proposed model consists of three steps: Thresholding, comparison of similarity and detection of anomlaous behavior as shown in Fig. 5.3. Cross Correlation coefficient is a good measure for comparing similarity and hence is used for both thresholding and signal comparison. The formula used for calculating the Cross-Correlation coefficient is:

$$CC_{ij} = \frac{\sum_{t=1}^{l} \left(sig_i\left(t\right) - \overline{sig_i}\right)\left(sig_j\left(t\right)\overline{sig_j}\right)}{\sqrt{\sum_{t=1}^{l}\left(sig_i\left(t\right) - \overline{sig_i}\right)^2 \sum_{t=1}^{l}\left(sig_j\left(t\right) - \overline{sig_j}\right)^2}} \tag{5.1}$$

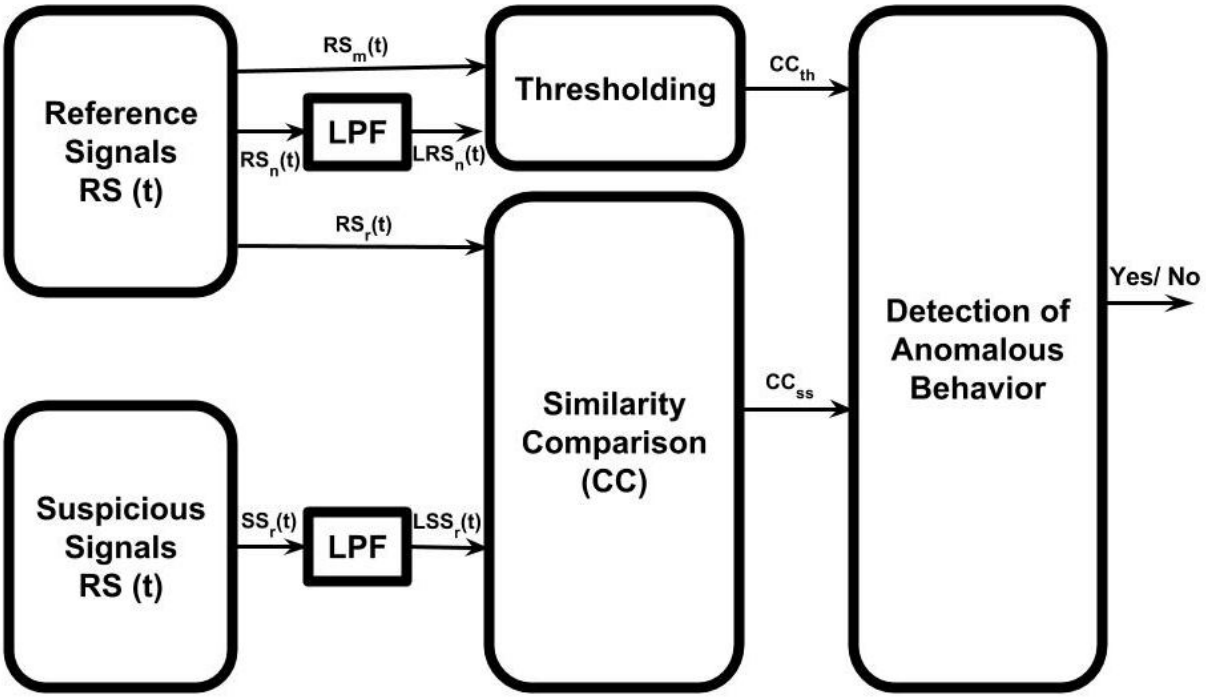


Figure 5.3: Anomaly Detection using Similarity Matrix

5.2.1 Thresholding

In this step we compare the degree of similarity of the reference signals $RS_r(t)$, $1 \leq r \leq N$, where N is the number of reference signals. As we have N reference signals, $N \times (N-1)$ different combinations are possible. The similarity of each of these combinations are calculated using Equation 5.1 and an array of Low Pass IIR filters to establish a matrix of thresholds, $CC_(th)$, that will used to detect anomalous behavior. The input to this step is the set of reference signals, $RS_t(t), RS_2(t), \ldots, RS_N(t)$ and the output is a $(N \times (N-1)) \times L$ matrix, where L is the number of Low Pass IIR filters used. The steps followed to determine the threshold matrix are as follows:

Step 1: We select one reference signal, say, $RS_1(t)$ and pass the rest of the reference signals, $RS_2(t), RS_3(t), \ldots, RS_N(t)$ through a Low Pass filter. The Low Pass filter is an IIR filter of order 20, with a Pass Band Frequency of $1Hz$ and Pass Band Ripple 0.1.

Step 2: In this step, we calculate the cross correlation coefficients of $RS_1(t)$ with each of the filtered outputs $LRS_2(t), LRS_3(t), \ldots, LRS_N(t)$ and the resultant values are stored in the column 1 of the $CC_(th)$.

Step 3: We repeat steps 1 and 2 for the remaining reference signals and the subsequent cross correlation coeffients are appended to column 1 of matrix $CC_(th)$.

Step 4: The Low Pass IIR filter is replaced with another Low Pass IIR filter with Pass Band Frequency of $2Hz$ and steps 1,2 and 3 are repeated. The resultant cross correlation values are stored in the second column of the matrix CC_th. The same sequence of steps are repeated with filters of Pass Band Frequencies $3Hz, 4Hz, \ldots, KHz$ and the cross correlation coefficients are stored in the corresponding columns, to obtain the $(N \times (N-1)) \times L$ threshold matrix.

A flow chart representing the steps followed for thresholding is shown in Fig. 5.4.

5.2.2 Comparison of Similarity

The power consumed by the device is measured and stored as suspicious signal, $SS(t)$. We need to determine if this signal is indicative of a device exhibiting anomalous behavior or normal behavior. To determine the behavior of the device, we determine the similarity of the lower spectral components of the suspicious signal $SS(t)$ and the reference signals. The inputs of this step are the reference signals $RS_1(t), RS_2(t), \ldots, RS_N(t)$ and the suspicious signal, $SS(t)$ and the output is a $N \times L$ similarity matrix, CC_{SS}. The steps followed are explained bellow and have also been represented as a flow chart in Fig. 5.5.

Step 1: The suspicious signal, $SS(t)$ is passed through a Low Pass IIR filter of order 20, with a Pass Band Frequency of $1Hz$ and Pass Band Ripple 0.1 and the filtered signal is called $LSS(t)$. The similarity of the filtered signal $LSS(t)$ and the reference signals is determined by calculating of the cross correlation coefficient of $LSS(t)$ and each of the reference signals, $RS_1(t), RS_2(t), \ldots, RS_N(t)$, and the calculated values are stored in column 1 of the similarity matrix, CC_{SS}.

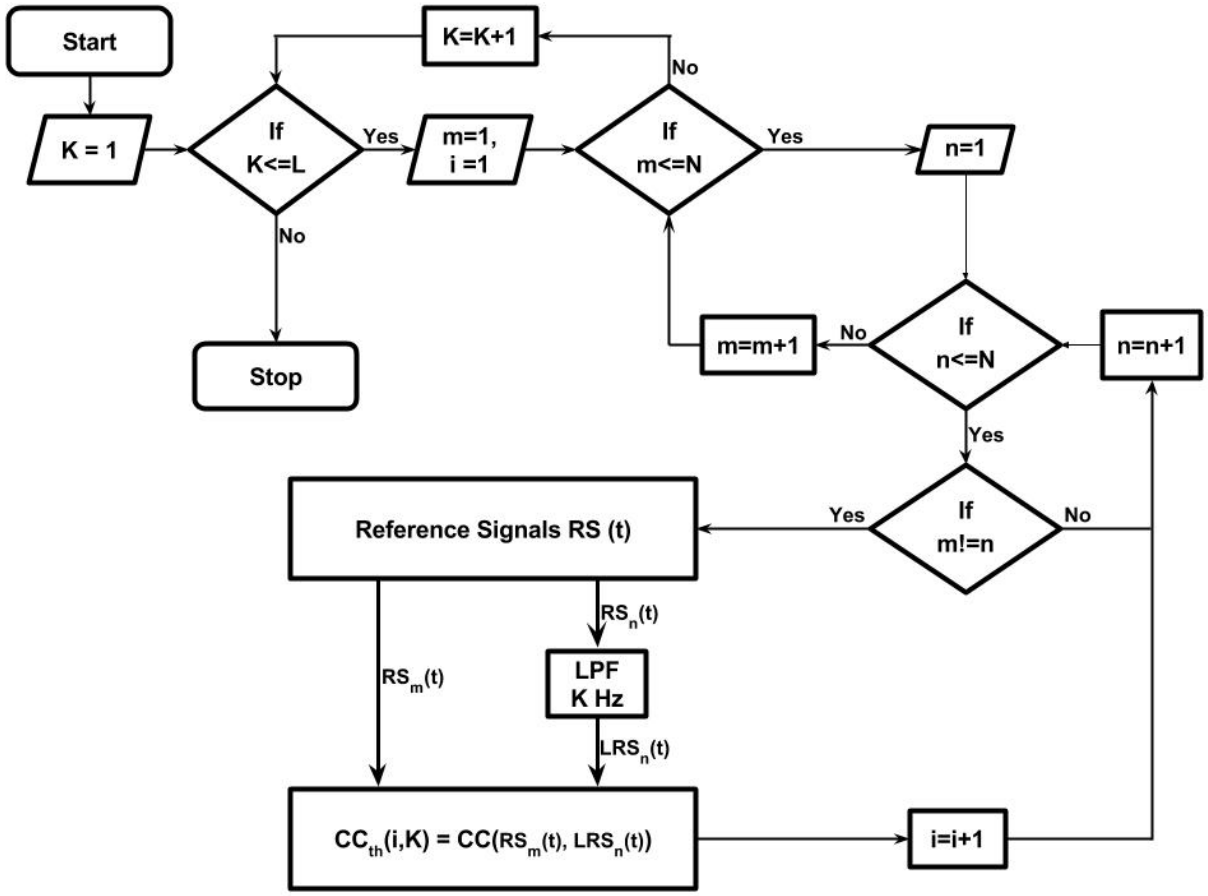


Figure 5.4: Thresholding Methodology

Step 2: We repeat step 1 with Low Pass IIR filters of pass band frequencies of $2Hz, 3Hz, \ldots, KHz$ and the calculated cross correlated coefficients are stored in the corresponding columns of the similarity matrix, $CC_(SS)$.

5.2.3 Detection of Anomalous Behavior

The step involves an iterative comparison of the degree of similarity between the suspicious signal, $SS(t)$ and the reference signals, $RS_1(t), \ldots, RS_N(t)$; and the similarity or consistency of the reference signals. The inputs to this step are the threshold matrix CC_{th} and

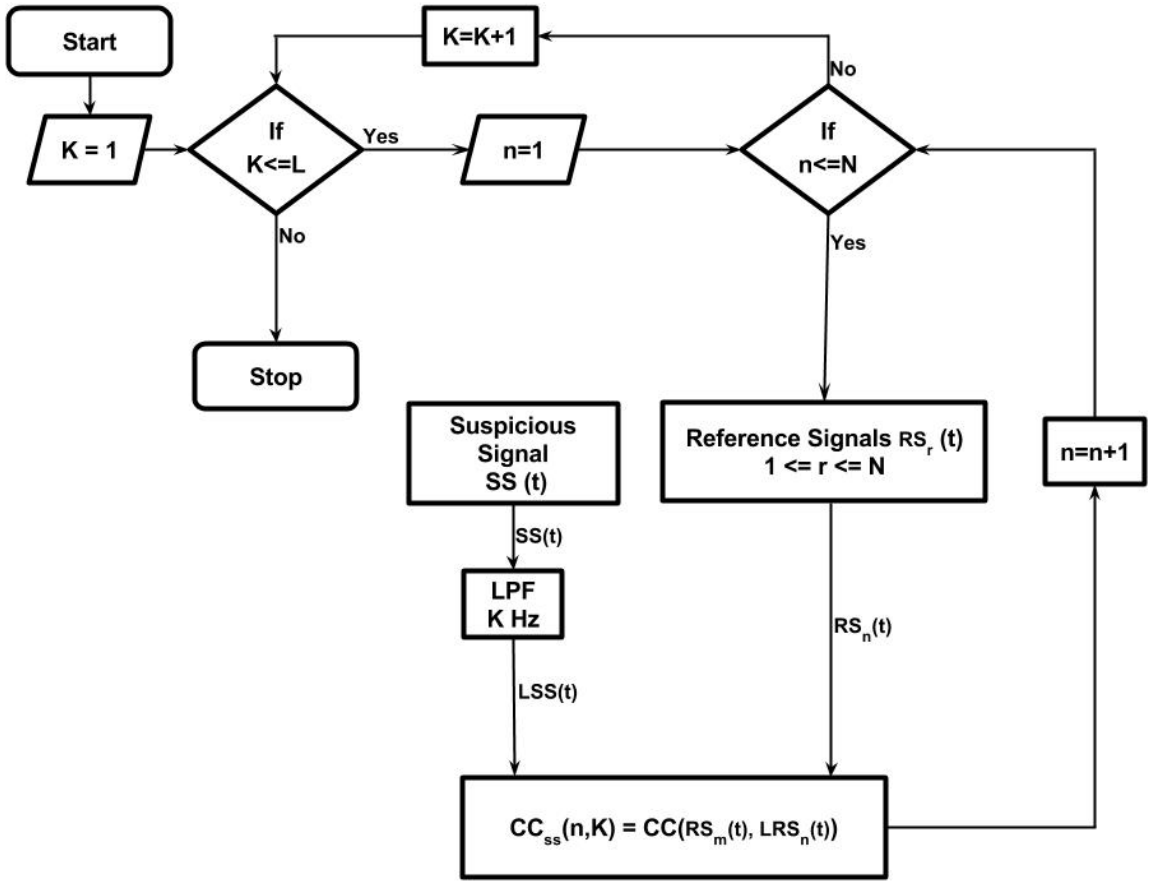


Figure 5.5: Steps followed for similarity comparison

similarity matrix CC_{SS}; and the outputs are a similarity count $Count_{CC}$ and a decision indicating the presence or absence of anomalous behavior in the device. A flowchart shown in Fig. 5.6 depicts the control flow for detecting anomalous behavior. From the experiments conducted, we understand that the similarity of power consumption in the absence of anomalous behavior lies largely within a certain range, with a few signals that show increased variance. The presence of anomalous behavior is enough to shift the number of values beyond the range mentioned earlier and hence calculating the number of these occurrences has help us in detecting anomalous behavior in the smartphone.

Step 1: Every value in column 1 CC_{SS} is compared with each of the values in column 1 of CC_{th}. If the value in CC_{SS} is lower than the value of CC_{th}, it means that the

suspicious signal is less similar and hence may be anomalous. Hence, the counter, $Count_{CC}$ is incremented by 1.

Steps 2: Step 1 is repeated for all the N columns of CC_{SS} and the average $count_{CC}$ is calculated.

Step 3: If the average is greater than a threshold η, then the smartphone has anomalous behavior.

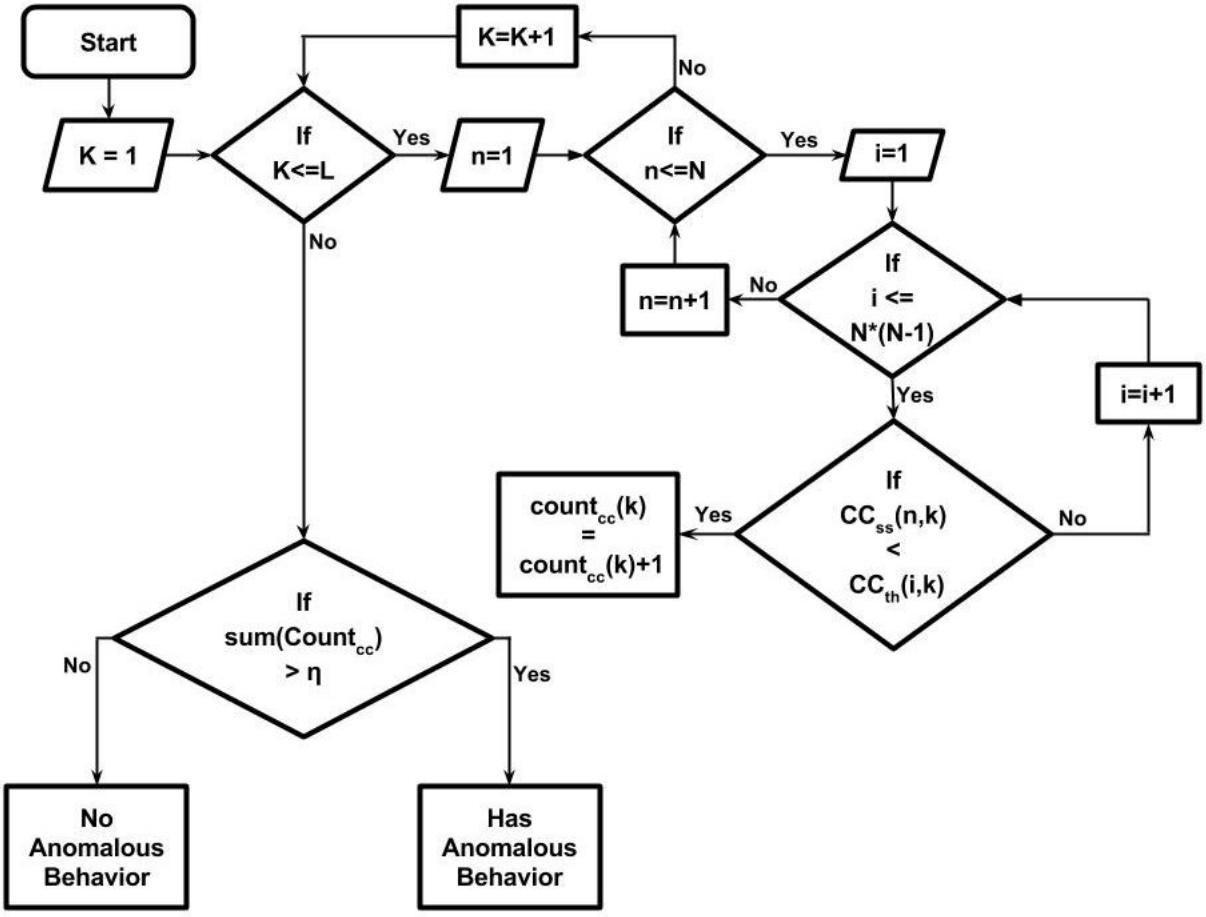


Figure 5.6: step 3: Detection of Anomalous Behavior

5.3 Validation of the model, analysis and results

Experiments were conducted to measure the power consumption of the smartphone in a trusted environment, thus exhibiting no anomalous behavior; and in the presence of a background app with varying duty cycles. The duty cycle of the background app was varied from 0.1% to 12% and it was observed that the accuracy of detection of anomalous behavior was 100% for all the duty cycles except 1% duty cycle, for which it was 93%. The occurrence of false positives was found to be at 20%, with 80% of no anomalous behavior accurately detected. The results are shown in Fig. 5.7.

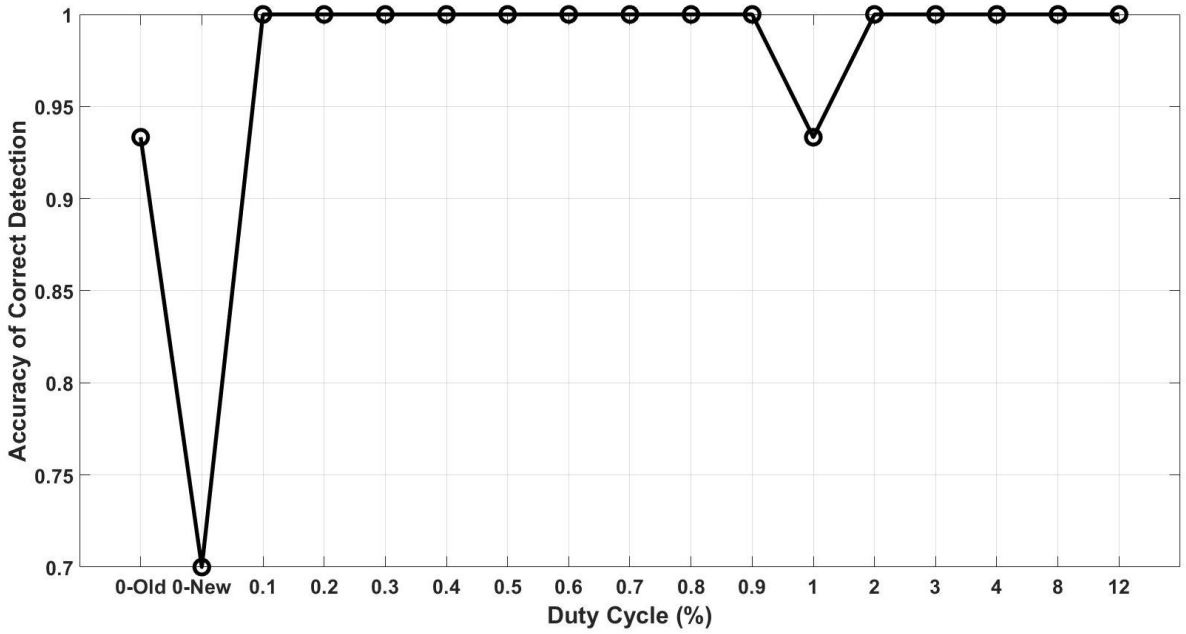


Figure 5.7: Accuracy of the Proposed Model

5.4 Comparision and Discussion

To compare the ICA approach with Similarity Matrix approach, Fig. **??** shows the F-measure results of both the approaches. The F-measure is a measure of the model's accuracy and is calculated from the precision and recall as shown in Eq. 5.2 which in turn are calculated by first forming the confusion matrix shown in Table 5.1. Therefore, we believe it fits the domain of anomalous behavior detection better than accuracy.

$$
\begin{aligned}
Precision &= \frac{TP}{TP + FP} \\
Recall &= \frac{TP}{TP + FN} \\
F\text{-}measure &= \frac{2 \times precision \times recall}{precision + recall}
\end{aligned} \tag{5.2}
$$

Table 5.1: Confusion Matrix

		Predicted Output	
		No Malware	With Malware
True Label	No Malware	True Positive (TP)	False Negative (FN)
	With Malware	False Positive (FP)	True Negative (TN)

The two methods were validated and compared by randomly choosing 10 signals for each duty cycle, where 5 had malware and the other 5 had no malware. As seen in Fig. 5.8 and Fig. 5.9, the Similarity Matrix approach performs better for anomalous behavior for all the duty cycles. It is interesting to note that both the approaches had a dip in accuracy for 1% duty cycle, but detected all the malware below 1% duty cycle. This is because, malware with such a short activity window, results in small bursts in power consumption which alter the signal and hence can be detected.

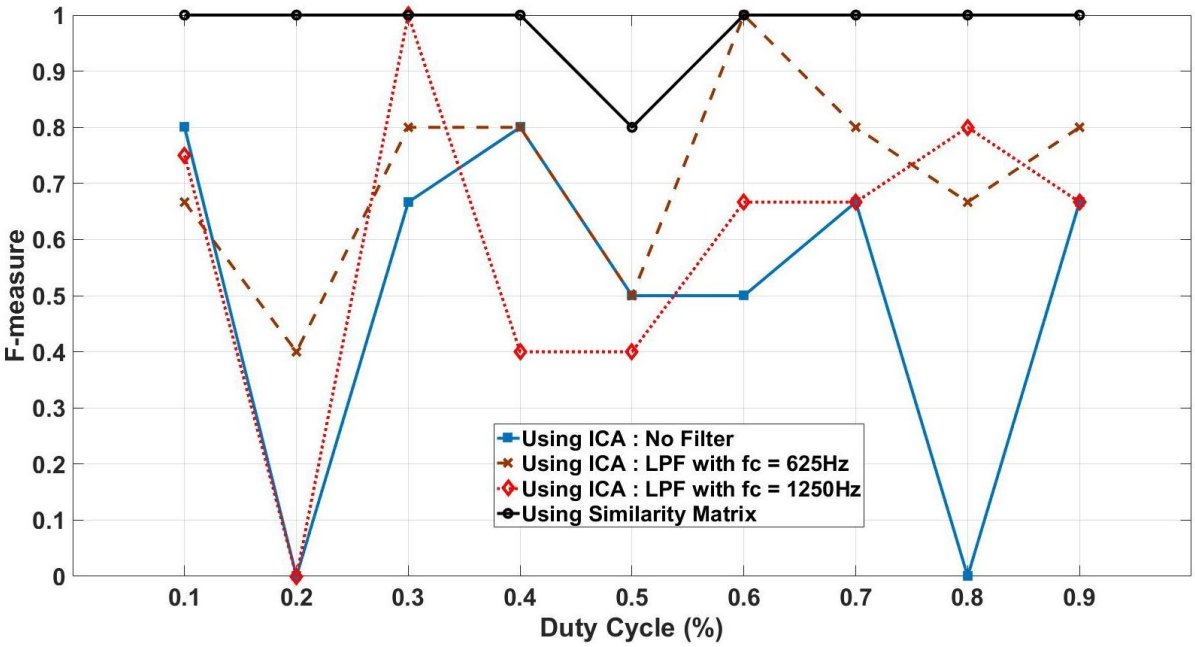


Figure 5.8: F-measure for malware with lower Duty Cycle

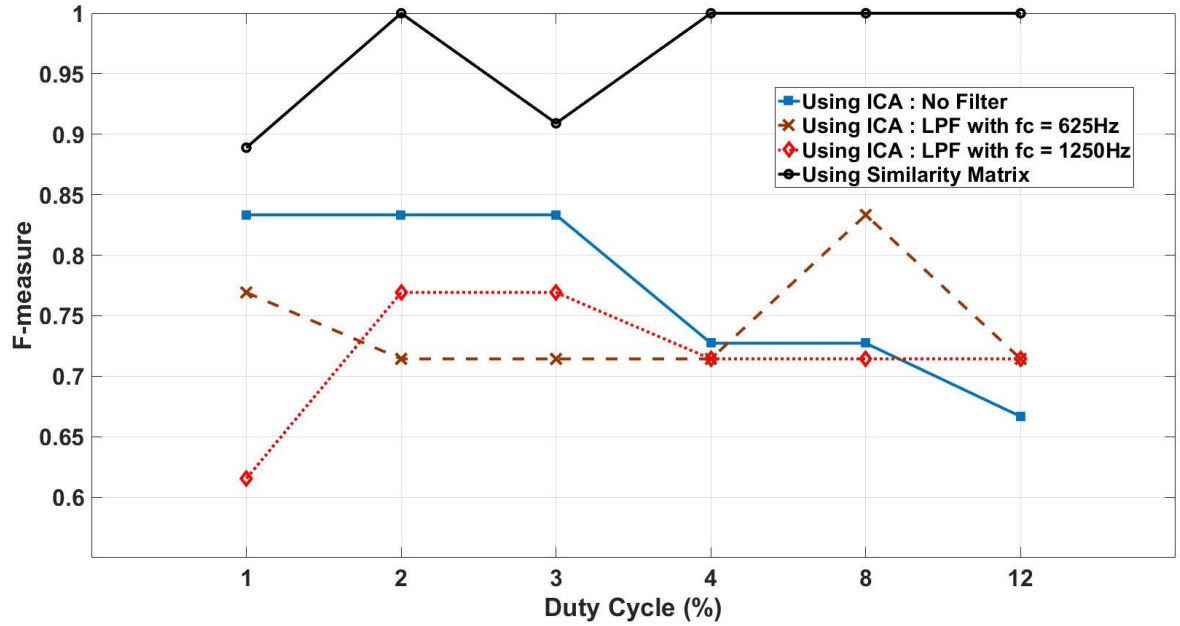


Figure 5.9: F-measure for malware with higher Duty Cycle

The values of precision, recall and F-measure calculated through Equations 5.2 for ICA with no filter are shown in Table. 5.2

Table 5.2: Validation of Anomaly Detection based on ICA (No Filter)

Duty Cycle	**Time in seconds**			**Detection - ICA**		
	T_{ON}	T_{OFF}	T_{Cycle}	**Precision**	**Recall**	**F-measure**
0.0 %	0.00	60.00	60	0.83	0.67	0.74
0.1 %	0.06	59.94	60	0.50	0.67	0.57
0.2 %	0.12	59.88	60	0.00	0.00	NA
0.3 %	0.18	59.82	60	0.67	1.00	0.80
0.4 %	0.24	59.76	60	0.67	1.00	0.80
0.5 %	0.30	59.70	60	1.00	0.50	0.67
0.6 %	0.36	59.64	60	1.00	0.50	0.67
0.7 %	0.42	59.58	60	0.50	1.00	0.67
0.8 %	0.48	59.52	60	0.50	0.50	0.50
0.9 %	0.54	59.46	60	0.67	1.00	0.80
1 %	0.60	59.40	60	0.67	0.80	0.73
2 %	1.20	58.80	60	0.71	1.00	0.83
3 %	1.80	58.20	60	0.56	1.00	0.71
4 %	2.40	57.60	60	0.60	0.60	0.60
8 %	4.80	55.20	60	0.77	0.60	0.67
12 %	7.20	52.80	60	0.57	0.80	0.67

The values of precision, recall and F-measure calculated through Equations 5.2 for ICA with LFP with cut-off frequency, $f_c = 625Hz$ are shown in Table. 5.3

Table 5.3: Validation of Anomaly Detection based on ICA with LFP, $f_c = 625Hz$

Duty Cycle	**Time in seconds**			**Detection - ICA with LFP,** $f_c = 625Hz$		
	T_{ON}	T_{OFF}	T_{Cycle}	**Precision**	**Recall**	**F-measure**
0.0 %	0.00	60.00	60	0.83	0.67	0.74
0.1 %	0.06	59.94	60	0.50	1.00	0.67
0.2 %	0.12	59.88	60	0.33	0.50	0.40
0.3 %	0.18	59.82	60	0.67	1.00	0.80
0.4 %	0.24	59.76	60	0.67	1.00	0.80
0.5 %	0.30	59.70	60	0.50	0.50	0.50
0.6 %	0.36	59.64	60	0.50	1.00	0.67
0.7 %	0.42	59.58	60	0.67	1.00	0.80
0.8 %	0.48	59.52	60	0.50	1.00	0.67
0.9 %	0.54	59.46	60	0.67	1.00	0.80
1 %	0.60	59.40	60	0.50	0.80	0.70
2 %	1.20	58.80	60	0.63	1.00	0.86
3 %	1.80	58.20	60	0.63	1.00	0.86
4 %	2.40	57.60	60	0.63	1.00	0.75
8 %	4.80	55.20	60	0.63	1.00	0.82
12 %	7.20	52.80	60	0.63	1.00	0.77

The values of precision, recall and F-measure calculated through Equations 5.2 for ICA with LFP with cut-off frequency, $f_c = 1250Hz$ are shown in Table. 5.4

Table 5.4: Validation of Anomaly Detection based on ICA with LFP, $f_c = 1250Hz$

Duty Cycle	**Time in seconds**			**Detection - ICA with LFP,** $f_c = 1250Hz$		
	T_{ON}	T_{OFF}	T_{Cycle}	**Precision**	**Recall**	**F-measure**
0.0 %	0.00	60.00	60	0.83	0.67	0.72
0.1 %	0.06	59.94	60	0.60	1.00	0.75
0.2 %	0.12	59.88	60	0.00	0.00	NA
0.3 %	0.18	59.82	60	0.50	1.00	0.67
0.4 %	0.24	59.76	60	0.33	0.50	0.40
0.5 %	0.30	59.70	60	0.33	0.50	0.40
0.6 %	0.36	59.64	60	1.00	1.00	1.00
0.7 %	0.42	59.58	60	0.67	1.00	0.80
0.8 %	0.48	59.52	60	0.50	1.00	0.67
0.9 %	0.54	59.46	60	0.67	1.00	0.80
1 %	0.60	59.40	60	0.44	0.80	0.57
2 %	1.20	58.80	60	0.63	1.00	0.77
3 %	1.80	58.20	60	0.56	1.00	0.71
4 %	2.40	57.60	60	0.56	1.00	0.71
8 %	4.80	55.20	60	0.50	0.80	0.61
12 %	7.20	52.80	60	0.56	1.00	0.71

The values of precision, recall and F-measure calculated through Equations 5.2 for anomaly detection using Similarity Matrix (SM) are shown in Table. 5.5

Table 5.5: Validation of Anomaly Detection based on Similarity Matrix

Duty Cycle	**Time in seconds**			**Detection - SM**		
	T_{ON}	T_{OFF}	T_{Cycle}	**Precision**	**Recall**	**F-measure**
0.0 %	0.00	60.00	60	0.83	1.00	0.91
0.1 %	0.06	59.94	60	0.83	1.00	0.91
0.2 %	0.12	59.88	60	0.83	1.00	0.91
0.3 %	0.18	59.82	60	0.83	1.00	0.91
0.4 %	0.24	59.76	60	0.83	1.00	0.91
0.5 %	0.30	59.70	60	0.83	1.00	0.91
0.6 %	0.36	59.64	60	0.83	1.00	0.91
0.7 %	0.42	59.58	60	0.83	1.00	0.91
0.8 %	0.48	59.52	60	0.83	1.00	0.91
0.9 %	0.54	59.46	60	0.83	1.00	0.91
1 %	0.60	59.40	60	0.82	0.93	0.87
2 %	1.20	58.80	60	0.83	1.00	0.91
3 %	1.80	58.20	60	0.83	1.00	0.91
4 %	2.40	57.60	60	0.83	1.00	0.91
8 %	4.80	55.20	60	0.83	1.00	0.91
12 %	7.20	52.80	60	0.83	1.00	0.91

Chapter 6

Conclusion and Future Work

In this thesis, we have proposed a model for detection of anomalous behavior using signal processing techniques. We developed a customized app to emulate an anomalous behavior and measured the power consumed by the smartphone with and without anomalous behavior. The intensity of the anomalous behavior was varied based on duty cycle. The measured power readings were used to detect anomalous behavior. The detection was done using two different approaches and the results were compared. It has been observed that detection model with ICA had an accuracy of 76.22% without the use of filters, but increased to 92.89% when a low pass filter with cut-off frequency $625Hz$ was used. The rate of false positives was 40%. The Similarity Matrix approach performed better in detecting anomalous behavior and had a detection rate of 99.56% and a false positive rate of 82% which is significantly better than any of the methods surveyed in the literature.

It was observed that both the methods are complementary, since both of them have the same approach with the only difference being the use of ICA in the first model. The detection model based on Similarity Matrix was more resource intensive than the model based on ICA. For future work, we aim to combine both the methods and increase the accuracy of detection whilst using less of the resources. The proposed models used low pass filters with cut-off frequencies $625Hz$ and $1250Hz$ for ICA and $1-5Hz$ for Similarity Matrix. For future work, the effectiveness of using low pass and high pass filters will be investigated. The validations and experiments done in this thesis used only the Youtube app. In future, We will be validating the model for other types of foreground apps and also include real time malware for detection.

References

[1] F-secure mobile threat report q1 2014, 204.

[2] Malware life cycle. http://slideplayer.com/slide/1467916/4/images/4/Life+cycle+of+a+modern+attack.jpg.

[3] Monsoon power monitor. http://msoon.github.io/powermonitor/.

[4] Monsoon power solutions. https://www.msoon.com/LabEquipment/PowerMonitor/.

[5] Samsung galaxy s5. http://www.gsmarena.com/samsung_galaxy_s5_neo-6506.php.

[6] Staistica, the statistics portal. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[7] *Battery-based intrusion detection a first line of defense*, June 2005.

[8] It threat evolution. https://securelist.com/it-threat-evolution-q2-2017-statistics/79432/, 2017.

[9] Hajer Al Housani, Hadi Otrok, Rabeb Mizouni, Jean-Marc Robert, and Azzam Mourad. Towards smart anti-malwares for battery-powered devices. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–4. IEEE, 2012.

[10] Muhammad Anshari, Mohammad Nabil Almunawar, Masitah Shahrill, Danang Kuncoro Wicaksono, and Miftachul Huda. Smartphones usage in the classrooms: Learning aid or interference? *Education and Information Technologies*, pages 1–17, 2017.

[11] L. Apvrille and A. Apvrille. Identifying unknown android malware with feature extractions and classification techniques. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 182–189, Aug 2015.

[12] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*, 2014.

[13] GDATA Security Blog. Malware trends. https://www.gdatasoftware.com/blog/2017/04/29666-malware-trends-2017, 2017.

[14] Maged N Kamel Boulos, Steve Wheeler, Carlos Tavares, and Ray Jones. How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from ecaalyx. *Biomedical engineering online*, 10(1):24, 2011.

[15] Jozef Bucko. Security of smart banking applications in slovakia. *Journal of theoretical and applied electronic commerce research*, 12(1):42–52, 2017.

[16] T. K. Buennemeyer. Mobile device profiling and intrusion detection using smart batteries. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pages 296–296, Jan 2008.

[17] Timothy K Buennemeyer, Theresa M Nelson, Lee M Clagett, John P Dunning, Randy C Marchany, and Joseph G Tront. Mobile device profiling and intrusion detection using smart batteries. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 296–296. IEEE, 2008.

[18] Niken Dwi Wahyu Cahyani, Ben Martini, Kim-Kwang Raymond Choo, and AKBP Al-Azhar. Forensic data acquisition from cloud-of-things devices: windows smartphones as a case study. *Concurrency and Computation: Practice and Experience*, 29(14), 2017.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[20] Bonnie A Clough and Leanne M Casey. The smart therapist: A look to the future of smartphones and mhealth technologies in psychotherapy. *Professional Psychology: Research and Practice*, 46(3):147, 2015.

[21] Monica Curti, Alessio Merlo, Mauro Migliardi, and Simone Schiappacasse. Towards energy-aware intrusion detection systems on mobile devices. In *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, pages 289–296. IEEE, 2013.

[22] Pasquale Daponte, L De Vito, F Picariello, and M Riccio. State of the art and future developments of measurement applications on smartphones. *Measurement*, 46(9):3291–3307, 2013.

[23] Bryan Dixon, Yifei Jiang, Abhishek Jaiantilal, and Shivakant Mishra. Location based power analysis to detect malicious code in smartphones. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 27–32. ACM, 2011.

[24] E Ray Dorsey, Michael V McConnell, Stanley Y Shaw, Andrew D Trister, Stephen H Friend, et al. The use of smartphones for health research. *Academic Medicine*, 92(2):157–160, 2017.

[25] J ea Qadri. A review of significance of energy-consumption anomaly in malware detection in mobile devices. *International Journal on Cyber Situational Awareness*, 1(1), 2016.

[26] Marwa M. A. Elfattah, Aliaa A. A. Youssif, and Ebada Sarhan Ahmed. Handsets malware threats and facing techniques. *CoRR*, abs/1204.1601, 2012.

[27] Karim O. Elish, Danfeng (daphne Yao, and Barbara G. Ryder. User-centric dependence analysis for identifying malicious mobile apps.

[28] Karim O Elish, Danfeng Yao, and Barbara G Ryder. User-centric dependence analysis for identifying malicious mobile apps. In *Workshop on Mobile Security Technologies*, 2012.

[29] William Enck. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 393–407, 2010.

[30] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[31] The Guardian. How secure is your smartphone. https://www.theguardian.com/media-network/2015/sep/29/how-secure-is-your-smartphone, 2015.

[32] M. Guri, G. Kedma, B. Zadov, and Y. Elovici. Trusted detection of sensitive activities on mobile phones using power consumption measurements. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, pages 145–151, 2014.

[33] Mordechai Guri, Yuri Poliak, Bracha Shapira, and Yuval Elovici. Joker: Trusted detection of kernel rootkits in android devices via jtag interface. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 65–73. IEEE, 2015.

[34] Jianjun Huang and Xiangyu Zhang. Asdroid: Detecting stealthy behaviors in android applications by user interface and program behavior contradiction. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 1036–1046, 2014.

[35] Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634, 1999.

[36] Nayeem Islam and Roy Want. Smartphones: Past, present, and future. *IEEE Pervasive Computing*, 13(4):89–92, 2014.

[37] International Telecommunication Union (ITU). Measuring the information society report volume 1. https://www.itu.int/en/ITU-/Statistics/Documents/publications/misr2017/MISR2017_Volume1.pdf, 2017.

[38] Grant A Jacoby and NathanielJ Davis. Battery-based intrusion detection. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 4, pages 2250–2255. IEEE, 2004.

[39] H. Kim, K. G. Shin, and P. Pillai. Modelz: Monitoring, detection, and analysis of energy-greedy anomalies in mobile handsets. *IEEE Trans. on Mobile Computing*, 10(7):968–981, July 2011.

[40] Hahnsang Kim, Kang G Shin, and Padmanabhan Pillai. Modelz: monitoring, detection, and analysis of energy-greedy anomalies in mobile handsets. *IEEE Transactions on Mobile Computing*, 10(7):968–981, 2011.

[41] Hahnsang Kim, Joshua Smith, and Kang G. Shin. Detecting energy-greedy anomalies and mobile malware variants. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 239–252, 2008.

[42] S. Kumar, A. Viinikainen, and T. Hamalainen. Machine learning classification model for network based intrusion detection system. In *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 242–249, Dec 2016.

[43] Malwarebytes Labs. State of malware report. https://www.malwarebytes.com/pdf/white-papers/stateofmalware.pdf, 2017.

[44] McAfee Labs. Threats report. https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf, June 2017.

[45] Mobile Marketing. http://mobilemarketingmagazine.com/24bn-smartphone-users-in-2017-says-emarketer, 2017.

[46] Yisroel Mirsky, Asaf Shabtai, Bracha Shapira, Yuval Elovici, and Lior Rokach. Anomaly detection for smartphone data streams. *Pervasive and Mobile Computing*, 35:83–107, 2017.

[47] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[48] Abu Saleh Mohammad Mosa, Illhoi Yoo, and Lincoln Sheets. A systematic review of healthcare applications for smartphones. *BMC medical informatics and decision making*, 12(1):67, 2012.

[49] Alexios Mylonas, Stelios Dritsas, Bill Tsoumas, and Dimitris Gritzalis. Smartphone security evaluation the malware attack case. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 25–36. IEEE, 2011.

[50] Ebrahim Nemati, Christina Batteate, and Michael Jerrett. Opportunistic environmental sensing with smartphones: a critical review of current literature and applications. *Current Environmental Health Reports*, 4(3):306–318, 2017.

[51] P. O'Kane, S. Sezer, K. McLaughlin, and E. G. Im. Svm training phase reduction using dataset feature filtering for malware detection. *IEEE Transactions on Information Forensics and Security*, 8(3):500–509, March 2013.

[52] S. Papadopoulos, A. Drosou, and D. Tzovaras. A novel graph-based descriptor for the detection of billing-related anomalies in cellular mobile networks. *IEEE Transactions on Mobile Computing*, 15(11):2655–2668, Nov 2016.

[53] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies. *Security in Computing (5th Edition)*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2015.

[54] Jameel et al Qadri. A review of significance of energy-consumption anomaly in malware detection in mobile devices. *International Journal on Cyber Situational Awareness*, 1(1), 2016.

[55] Julia Rubin, Michael I Gordon, Nguyen Nguyen, and Martin Rinard. Covert communication in mobile applications (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 647–657. IEEE, 2015.

[56] Aubrey-Derrick Schmidt, Frank Peters, Florian Lamour, Christian Scheel, Seyit Ahmet Çamtepe, and Şahin Albayrak. Monitoring smartphones for anomaly detection. *Mobile Networks and Applications*, 14(1):92–106, 2009.

[57] Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Leonid Batyuk, Jan Hendrik Clausen, Seyit Ahmet Camtepe, Sahin Albayrak, and Can Yildizli. Smartphone malware evolution revisited: Android next target? In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 1–7. IEEE, 2009.

[58] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(4):440–448, 2002.

[59] Symantec. Internet security threat report (istr). https://www.symantec.com/security-center/threat-report, April 2017.

[60] Yong Wang, Kevin Streff, and Sonell Raman. Smartphone security challenges. *Computer*, 45(12):52–58, 2012.

[61] M. R. Watson, N. u. h. Shirazi, A. K. Marnerides, A. Mauthe, and D. Hutchison. Malware detection in cloud computing infrastructures. *IEEE Transactions on Dependable and Secure Computing*, 13(2):192–205, March 2016.

[62] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu. Droidmat: Android malware detection through manifest and api calls tracing. In *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*, pages 62–69. IEEE, 2012.

[63] Y. Xue, G. Meng, Y. Liu, T. H. Tan, H. Chen, J. Sun, and J. Zhang. Auditing anti-malware tools by evolving android malware and dynamic loading technique. *IEEE Transactions on Information Forensics and Security*, 12(7):1529–1544, July 2017.

[64] Hongyu Yang and Ruiwen Tang. Power consumption based android malware detection. *Journal of Electrical and Computer Engineering*, 2016, 2016.

[65] Hongyu Yang and Ruiwen Tang. Power consumption based android malware detection. *J. Electrical and Computer Engineering*, 2016:6860217:1–6860217:6, 2016.

[66] Thomas Zefferer, Peter Teufl, David Derler, Klaus Potzmader, Alexander Oprisnik, Hubert Gasparitz, and Andrea Höller. Towards secure mobile computing: Employing power-consumption information to detect malware on mobile devices.

[67] Thomas Zefferer, Peter Teufl, David Derler, Klaus Potzmader, Alexander Oprisnik, Hubert Gasparitz, and Andrea Höller. Towards secure mobile computing: Employing power-consumption information to detect malware on mobile devices, 2014.

[68] Lide Zhang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference*, CODES/ISSS '10, pages 105–114, 2010.

[69] Lide Zhang, Birjodh Tiwana, Robert P Dick, Zhiyun Qian, Z Morley Mao, Zhaoguang Wang, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2010 IEEE/ACM/IFIP International Conference on*, pages 105–114. IEEE, 2010.