.

**CHAPTER 1**

**INTRODUCTION**

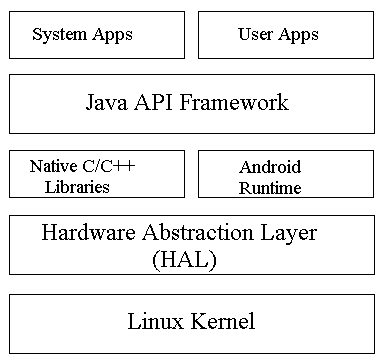
* 1. **Introduction to Android**

Android is an operating system and programming platform developed by Google for smartphones and other mobile devices (such as tablets). It can run on many different devices from many different manufacturers. Android includes a software development kit for writing original code and assembling software modules to create apps for Android users. It also provides a marketplace to distribute apps. All together, Android represents an ecosystem for mobile apps.

Apps are developed for a variety of reasons: addressing business requirements, building new services, creating new businesses, and providing games and other types of content for users. Developers choose to develop for Android in order to reach the majority of mobile device users.As the world's most popular mobile platform, Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It has the largest installed base of any mobile platform and is still growing fast. Every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital. Android provides a touch-screen user interface (UI) for interacting with apps. Android's user interface is mainly based on direct manipulation, using touch gestures such as swiping, tapping and pinching to manipulate on-screen objects. In addition to the keyboard, there’s a customizable virtual keyboard for text input. Android can also support game controllers and full-size physical keyboards connected by Bluetooth or USB.

The Android home screen can contain several pages of app icons, which launch the associated apps, and *widgets*, which display live, auto-updating content such as the weather, the user's email inbox or a news ticker. Android can also play multimedia content such as music, animation, and video. The figure above shows app icons on the home screen (left), playing music (center), and displaying widgets (right). Along the top of the screen is a status bar, showing information about the device and its connectivity. The Android home screen may be made up of several pages, between which the user can swipe back and forth. Android is designed to provide immediate response to user input. Besides a fluid touch interface, the vibration capabilities of an Android device can provide haptic feedback. Internal hardware such as accelerometers, gyroscopes and proximity sensors, are used by many apps to respond to additional user actions. These sensors can detect rotation of the screen from portrait to landscape for a wider view or it can allow the user to steer a virtual vehicle in a racing game by rotating the device as if it were a steering wheel.

The Android home screen can contain several pages of *app icons*, which launch the associated apps, and *widgets*, which display live, auto-updating content such as the weather, the user's email inbox or a news ticker. Android can also play multimedia content such as music, animation, and video. The figure above shows app icons on the home screen (left),playing music (center), and displaying widgets (right). Along the top of the screen is a status bar, showing information about the device and its connectivity. The Android home screen may be made up of several pages, between which the user can swipe back and forth. Android is designed to provide immediate response to user input. Besides a fluid touch interface, the vibration capabilities of an Android device can provide haptic feedback. Internal hardware such as accelerometers, gyroscopes and proximity sensors, are used by many apps to respond to additional user actions. These sensors can detect rotation of the screen from portrait to landscape for a wider view or it can allow the user to steer a virtual vehicle in a racing game by rotating the device as if it were a steering wheel. To develop apps using the SDK, use the Java programming language for developing the app and Extensible Markup Language (XML) files for describing data resources. By writing the code in Java and creating a single app binary, you will have an app that can run on both phone and tablet form factors. You can declare your UI in lightweight sets of XML resources, one set for parts of the UI that are common to all form factors, and other sets for features specific to phones or tablets. At runtime, Android applies the correct resource sets based on its screen size, density, locale, and so on. To help you develop your apps efficiently, Google offers a full Java Integrated Development Environment (IDE) called Android Studio, with advanced features for developing, debugging, and packaging Android apps. Using Android Studio, you can develop on any available Android device, or create virtual devices that emulate any hardware configuration. Android provides a rich development architecture. You don’t need to know much about the components of this architecture, but it is useful to know what is available in the system for your app to use. The following diagram shows the major components of the Android *stack* — the operating system and development architecture.



**Figure 1.1: Android stack — the operating system and development architecture.**

**1.2 Introduction to Fall Detection app**

Exponential increase in falls are recognized as a major factor causing physical, psychological, and economical concerns within the growing elderly population worldwide. Statistical reports from the World Health Organization (WHO) indicate that 28%-35% of seniors over 64 are subject to a fall event each year, which further elevates to 32%-42% for those over 70 years of age These falls are responsible for majorly 90% of hip and wrist fractures and 60% of head injuries Besides these injuries, long-lie situation (i.e. remaining on the ground for long time) is another outcome of fall that has serious consequences such as dehydration, hypothermia and even death. Moreover, frequent incidence of falls in the elderly may provoke fear of falling (FoF) which in turn, deteriorates their confidence in living independently and being socially active .Thus, the development of automated, reliable, and prompt fall detection systems is vital to guarantee immediate assistance in case of falls, especially those involving long lies, and minimize severe health complications .Contemporary techniques employed for automatic detection of imminent real-life falls can be broadly classified into two categories: (i) context-aware systems and (ii) wearable systems. The former category concerns the deployment of sensory gadgets such as cameras, microphones, infrared, and pressure sensors to track the movement of people in limited environments. The main strength of these systems lies in usability amongst the elderly as no dedicated device is needed to be worn. Nonetheless, such systems are vulnerable to issues such as limited coverage, high installation cost, high false alarms due to other mobile entities, and privacy (especially in video based systems). Fall detection methods based on wearable motion sensors that rely on kinematic signals, like tri-axial accelerometers and gyroscopes, fall under the latter category.

While such body-worn systems offer several advantages over video-based systems, the bearer is still required to carry at least one device which may be intrusive and raise usability concerns. Moreover, the cost incurred by wearable commercial/customized fall detection devices such as LifeCall and LPFD is another issue of fundamental importance. Momentum in the advancement of micro electro-mechanical systems (MEMS) technology has however, enabled the development of very light, compact, low power, and inexpensive wireless inertial sensors that eliminate concerns regarding their portability and user inconvenience . Consequently, smartphones (SPs) integrated with dedicated detection hardware serve as potential candidates that have widely been accepted as a daily life commodity . The remarkable penetration of smart phones (SPs) as a mobility and safety tool amongst the elderly has therefore, led the surge for cost-effective, efficient, and commercially viable fall detection applications. Standalone body-worn or SP-based fall detectors have been extensively surveyed . Earlier studies were predominantly grounded on threshold based methods (TBM) that rely on certain decision thresholds applied on features extracted from the inertial sensor signals. These detection algorithms are easy to implement, offer less computation cost, and are power efficient. The author compared four famous TBM algorithms with a commercial device and showed that the TBM techniques fail to avoid false negatives (falls that remain undetected) and false positives (activities of daily living (ADL) classified as falls) simultaneously. Since fine-tuning the thresholds invokes a trade-off between number of false negatives and false positives, it is difficult to attain optimal thresholds ensuring performance consistency for everybody. More recent studies reported in the literature use sophisticated machine learning (ML) techniques such as support vector machines (SVMs) , artificial neural networks (ANNs), and k-nearest neighbour (k-NN) to classify falls from ADL. A comprehensive Mat lab-based comparison of the offline classification performance of six ML approaches is given in , wherein 1,404 features extracted from accelerometer, gyroscope and magnetometer sensing were used by the authors to obtain an average accuracy of around 99%. In spite of the profound results, usage of multiple sensors, especially gyroscope, along with the high computational cost of ML techniques exploiting large feature sets demands high power consumption thus, making them unfeasible for SPs. With over 86% of the market share in the third quarter of 2016 and due to its open source approach, Android OS stands out as the most widely used programming environment for SP-based fall detection solutions . Though many SP based fall detection algorithms have been reported, yet only few have actually been tested in real-life. Moreover, to our best knowledge, Smart Fall Detector (SFD) and iFall are the only two applications that have been released for public use. Other fall detection Android apps available on Google Play Store include Fade: Fall detector, Emergency Fall detector, and T3 Lab . Nonetheless, information on neither the underlying algorithm nor their performance exists.

* 1. **Permission Induced Risks in Android**

As the official application (or app) market, Google’s Play store provides a platform of delivering apps for Android smartphones and mobile devices. There are many third-party app markets providing similar platforms. App developers publish their apps on the Google’s play or on the third-party app markets, where end users download and install their interested apps on their Android smartphones. Obviously, how to detect and keep the large number of malware out of the application (or app) markets is an emerging, crucial, but challenging issue. Previous work on the detection of malapps mainly focused on permissions ,static and dynamic analysis Permission control is one of the major Android security mechanisms. Android permissions provide fine-grained security features by enforcing restrictions on the specific operations that a particular process can perform.

However, it imparts a significant responsibility to the app developers with regard to declaring the least-privileged set of permissions needed by designed apps, and to the app users with regard to fully understanding the risk of granting certain combinations of permissions. Android provides developers documentation, but its permission information is limited. On the one hand, the lack of reliable permission information may let developers request unnecessary permissions, resulting in over privileged applications that users may cancel the installation. In addition, the unnecessarily risky permissions may be leaked to other malapps , leading to the permission re-delegation attacks . On the other hand, the lack of risk information of permissions confuses the users with regard to determining whether to install the app or not. Current Android permission warnings do not help most users make correct security decisions. It is feasible to identify malapps through analysing the permission usage patterns, as intuitively an app’s behaviour is characterized by the permissions it requests. We thus see that exploring the permission-induced risk is beneficial to three parties, the Android app developers, the users, as well as the malapp detectors. Curiosities are aroused on understanding the following questions: what is the ranking of the permissions with respect to (w.r.t.) the risk to the Android system; what is the subset of permissions that collaboratively cause security issues in malapps; to what degree the Android malapps can be detected based on the permissions they requests; and whether there exist fine-grained permission rules that can be used to identify unknown malapps (zero-day malapps), like the 9 detection rules with permissions called Kirin. We are motivated to answer the above questions aboutthe permission system of Android, in the vision of risk evaluation of Android permissions based on systematically quantitative analysis of Android apps on a very large scale(we consider 310,926 free apps from Google’s play and 4,868 real-world malapps). To fulfill the goal of exploration, our study is performed on the following three levels. First, we systematically analyze the risk of each individual permission and the risk of a group of collaborative permissions by employing machine learning techniques, such as feature ranking with mutual information, Correlation Coefficient(CorrCoef) and T-test, subset selection and transformation with Sequential Forward Selection (SFS) as well as Principal Component Analysis (PCA). Second, we evaluate the usefulness of risky permissions for malapp detection using classification algorithms, suck as Support Vector Machine(SVM), decision tree as well as Random Forest. Last but not least, we discuss and analyze in depth the feasibility as well as the limitations of malapp detection based on permissions requests.