

MALWARE DETECTION AND CLASSIFICATION USING MACHINE LEARNING

A Project Report Phase – II

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

by

ROHITH MODEM (39130380)

A ROHAN KUMAR (39130378)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICAITON ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of ROHITH MODEM (39130380) and AMBADI ROHAN KUMAR (39130378) who carried out the project entitled "**MALWARE DETECTION AND CLASSIFICATION USING MACHINE LEARNING**" under our supervision from November 2022 to April 2023.

Internal Guide

Dr. T. VINO, M.E., Ph.D.,

Head of the Department

Dr. T. RAVI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 23.4.2023

Internal Examiner

External Examiner

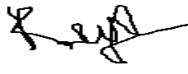
DECLARATION

We, ROHITH MODEM (39130380) and AMBADI ROHAN KUMAR (39130378) hereby declare that the Project Report entitled “**MALWARE DETECTION AND CLASSIFICATION USING MACHINE LEARNING**” done by us under the guidance of Dr.T.Vino, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 23.4.2023

PLACE: Chennai

SIGNATURE OF THE CANDIDATE(S)

1. 

2. 

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D., Professor & Dean, School of Electrical and Electronics** and **Dr. T. RAVI, Ph.D., Professor & Head, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our **Project Guide Dr.T.VINO, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

The interface allows the user to search for an arbitrary application on the Play Store the permissions list and the privacy policy are then automatically retrieved, whenever The user has then the ability of selecting a specific permission, and a list of relevant sentences are extracted by the privacy policy and presented to them, along with an accurate description of the permission itself. Such an interface allows the user to quickly evaluate the privacy-related risks of an Android application, by highlighting the relevant sections of the privacy policy and by providing useful information about sensible permissions. We presented a novel approach to the analysis of privacy policies in the context of Android applications. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. The tool is developed with expandability in mind, and further developments in the approach can easily be integrated in order to increase the reliability and effectiveness. In addition, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws. We proposed, A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene.

TABLE OF CONTENTS

Chapter No	TITLE	Page No
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	Domain Introduction	2
	Over view of the Project	3
2	LITERATURE SURVEY	4
	2.1 Analysis of Bayesian classification	4
	2.2 Android permissions	5
	2.3 Android platform-based individual privacy	8
3	MACHINE LEARNING	10
	3.1 Machine Learning Algorithms	13
	3.1.1 Supervised Learning	13
	3.1.2 Classification	14
	3.1.3 Regression	14
	3.1.4 Unsupervised Learning	14
	3.2 How to choose Machine learning Algorithm	15
	3.2.1 Challenges and limitations of ML	15
	3.3 Applications of Machine learning	15
	3.3.1 Advantages of Machine learning	16
	3.4 Data Mining	17
	3.4.1 Optimization	18
	3.4.2 Generalization	18
	3.5 Statistics	18
	3.6 Approaches	19
	3.6.1 Types of Machine learning algorithms	19
	3.7 Machine learning Models	23
4	EXISTING AND PROPOSED METHODOLOGY	26
	4.1 Existing System	26
	4.1.1 Disadvantages of Existing system	26
	4.2 Proposed system	26

	4.2.1 Advantages of Proposed system	26
	4.3 How Naïve bayes algorithm works	27
5	SYSTEM ARCHITECTURE	28
	System Architecture	28
	System Requirements	29
	Pentium Dual Core	29
	Java	30
	Windows	30
	Android Studio	31
	Modules	32
	Applications of Java	35
	Features of java	35
	Collection frame work	37
	System Design and Testing Plan	38
	Input design	38
	Output Design	39
	Modules	42
	System Study	44
	Non-Functional Requirements	47
	System Testing	47
	5.10Types of Tests	47
6	CONCLUSION	51
	REFERENCES	52

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
3.1	Traditional Programming	11
3.2	Machine learning	11
3.3	Learning Phase of ML	12
3.4	Interface model of ML	12
3.5	Machine learning Algorithms	13
5.1	Block Diagram for System Architecture	28
5.2	Pentium Dual Core	29
5.3	Java	30
5.4	Windows	30
5.5	Android Studio	31
5.6	Block Diagram of Platform Independent	36
5.7	Collection of Frame Work	38
5.8	Block Diagram of Use Case Diagram	39
5.9	Activity Diagram	40
5.10	Sequence Diagram	41
5.11	Collaboration Diagram	42
6.1	Initial Process	51
6.2	Scanning Process	52
6.3	Malware Detection Process	52
6.4	Existing vs Proposed Methodology	53

CHAPTER-1

INTRODUCTION

Mobile devices of various operating systems have exhibited a steep increase in the past decade, thus leading to an increase in the number and variety of applications that run on mobile devices. Smartphones are recently used either in a complementary manner to a desktop computer or even to replace it. A closer look at the purpose of using mobile phones reveal that they are mostly used for web browsing, social networking, and online banking. In addition, they are used for mobile-specific functions such as SMS messaging, real-time broadcasting of location, and ubiquitous access. As the capabilities in mobile phone functionality increase (e.g., applications for personal health), mobile phones become more and more attractive for a wider population. Market data surveys reveal that the number of smartphone sales worldwide reached 208 million in 2012. This was a 38.3% increase compared to the sales in 2011. The development of smartphones and mobile applications has resulted in a major change in people's way of doing tasks in various aspects of daily life, such as making business and conducting social communication. Mobile phone applications exhibit a rich variety not only in common daily life activities but also for users with more specific needs. From games to multimedia applications, navigation systems, and health-related applications, recently available mobile application markets, such as Google's Play Market, Apple's App Store, or Microsoft's Windows Store offer a wide variety of applications to users with different needs. The major application markets have been growing steadily both in terms of the applications offered to the users and the downloads performed by the users. The fast increase in the popularity of smartphones has led to an increase in their potential as a target for malicious activities. This is mainly because users provide access for various types of private information by means of mobile applications. As a result, some applications in the market have been identified as performing malicious activities. The spread of malicious software is also influenced by the policy of the market providers. For instance, Apple's App Store recently applies a policy that is subject to strict registration and company-issued digital certification before the release of any application, thus providing a security check for the applications listed in their application platform regularly. Others, such as Google's Play Market, introduce more freedom to developers in uploading their own software to the market.

Applications are then removed from the market in case of reported malicious activity. In particular, the Android operating system (in its recent form) allows removal of the malicious application from the device remotely. A similar mechanism of removal is also employed by Apple's App Store. The policy of post-detection removal of malicious software brings the need for early detection of malware applications.

DOMAIN INTRODUCTION

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

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 9 "Pie", released in August 2018. Google released the first Android Q beta on all Pixel phones on March 13, 2019. The core Android source code is known as Android Open Source Project (AOSP), and is primarily licensed under the Apache License.

At Google I/O 2014, the company revealed that there were over one billion active monthly Android users, up from 538 million in June 2013. Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software developed and licensed by Google. Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices.

OVERVIEW OF THE PROJECT

The GP-PP (Generic Permissions-Privacy invasive permissions) model is useful to classify the permissions into Generic and Privacy invasive permissions. The model proposes a simplistic way for users to decide which apps are dangerous to install. Based on the permission set that a particular app requests, the GP-PP model classifies an app as privacy invasive if majority of the permissions requested are privacy invasive. So, users can decide which set of permissions can be harmful. Therefore, by looking at the set of permissions requested, a user can determine whether an app is privacy invasive or not. In other words, a user can determine whether it is safe to install the said app or not. Although the proposed GP-PP model seems valuable, its efficacy has not been fully assessed.

To address this gap, we test the effectiveness of the GP-PP model using Naïve Bayes Classifier in this paper. In particular, we validate the GP-PP model in order to verify whether the model classifies an app on the basis of permission sets that the app requests.

Our study confirms that greater the number of Privacy-invasive permissions that an app requests, more dangerous it is to install the app. In summary, the major contribution of the paper is the validation of the proposed GP-PP model using Naïve Bayes Classifier and decision tree algorithm for popularity.

CHAPTER-2

LITERATURE REVIEW

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

Analysis of Bayesian classification-based approaches for Android malware detection

Mobile malware has been growing in scale and complexity spurred by the unabated uptake of smartphones worldwide. Android is fast becoming the most popular mobile platform resulting in sharp increase in malware targeting the platform. Additionally, Android malware is evolving rapidly to evade detection by traditional signature-based scanning. Despite current detection measures in place, timely discovery of new malware is still a critical issue. This calls for novel approaches to mitigate the growing threat of zero-day Android malware. Hence, the authors develop and analyse proactive machine-learning approaches based on Bayesian classification aimed at uncovering unknown Android malware via static analysis. The study, which is based on a large malware sample set of majority of the existing families, demonstrates detection capabilities with high accuracy. Empirical results

and comparative analysis are presented offering useful insight towards development of effective static-analytic Bayesian classification-based solutions for detecting unknown Android malware.

Android permissions: a perspective combining risks and benefits

The phenomenal growth of the Android platform in the past few years has made it a lucrative target of malicious application (app) developers. There are numerous instances of malware apps that send premium rate SMS messages, track users' private data, or apps that, even if not characterized as malware, conduct questionable actions affecting the user's privacy or costing them money. In this paper, we investigate the feasibility of using both the permissions an app requests, the category of the app, and what permissions are requested by other apps in the same category to better inform users whether the risks of installing an app is commensurate with its expected benefit. Existing approaches consider only the risks of the permissions requested by an app and ignore both the benefits and what permissions are requested by other apps, thus having a limited effect. We propose several risk signals that and evaluate them using two datasets, one consists of 158,062 Android apps from the Android Market, and another consists of 121 malicious apps. We demonstrate the effectiveness of our proposal through extensive data analysis.

Detecting repackaged smartphone applications in third-party android marketplaces.

Recent years have witnessed incredible popularity and adoption of smartphones and mobile devices, which is accompanied by large amount and wide variety of feature-rich smartphone applications. These smartphone applications (or apps), typically organized in different application marketplaces, can be conveniently browsed by mobile users and then simply clicked to install on a variety of mobile devices. In practice, besides the official marketplaces from platform vendors (e.g., Google and Apple), a number of third-party alternative marketplaces have also been created to host thousands of apps (e.g., to meet regional or localization needs). To maintain and foster a hygienic smartphone app ecosystem, there is a need for each third-party marketplace to offer quality apps to mobile users. In this paper, we perform a systematic study on six popular Android-based third-party marketplaces. Among them, we find a common "in-the-wild" practice of repackaging legitimate apps (from

the official Android Market) and distributing repackaged ones via third-party marketplaces. To better understand the extent of such practice, we implement an app similarity measurement system called DroidMOSS that applies a fuzzy hashing technique to effectively localize and detect the changes from app-repackaging behavior. The experiments with DroidMOSS show a worrisome fact that 5% to 13% of apps hosted on these studied marketplaces are repackaged. Further manual investigation indicates that these repackaged apps are mainly used to replace existing in-app advertisements or embed new ones to “steal” or re-route ad revenues. We also identify a few cases with planted backdoors or malicious payloads among repackaged apps. The results call for the need of a rigorous vetting process for better regulation of third-party smartphone application marketplaces.

A large scale study on application permissions and risk signals.

Third-party applications (apps) drive the attractiveness of web and mobile application platforms. Many of these platforms adopt a decentralized control strategy, relying on explicit user consent for granting permissions that the apps request. Users have to rely primarily on community ratings as the signals to identify the potentially harmful and inappropriate apps even though community ratings typically reflect opinions about perceived functionality or performance rather than about risks. With the arrival of HTML5 web apps, such user-consent permission systems will become more widespread. We study the effectiveness of user-consent permission systems through a large scale data collection of Facebook apps, Chrome extensions and Android apps. Our analysis confirms that the current forms of community ratings used in app markets today are not reliable indicators of privacy risks of an app. We find some evidence indicating attempts to mislead or entice users into granting permissions: free applications and applications with mature content request more permissions than is typical; 'look-alike' applications which have names similar to popular applications also request more permissions than is typical. We also find that across all three platforms popular applications request more permissions than average.

Analyzing sensitive data transmission in android for privacy leakage detection

Android phones often carry personal information, attracting malicious developers to embed code in Android applications to steal sensitive data. With known techniques in the literature, one may easily determine if sensitive data is being transmitted out of an Android phone. However, transmission of sensitive data in itself does not necessarily indicate privacy leakage; a better indicator may be whether the transmission is by user intention or not. When transmission is not intended by the user, it is more likely a privacy leakage. The problem is how to determine if transmission is user intended. As a first solution in this space, we present a new analysis framework called AppIntent. For each data transmission, AppIntent can efficiently provide a sequence of GUI manipulations corresponding to the sequence of events that lead to the data transmission, thus helping an analyst to determine if the data transmission is user intended or not. The basic idea is to use symbolic execution to generate the aforementioned event sequence, but straightforward symbolic execution proves to be too time-consuming to be practical. A major innovation in AppIntent is to leverage the unique Android execution model to reduce the search space without sacrificing code coverage. We also present an evaluation of AppIntent with a set of 750 malicious apps, as well as 1,000 top free apps from Google Play. The results show that AppIntent can effectively help separate the apps that truly leak user privacy from those that do not.

Android permissions demystified

Android provides third-party applications with an extensive API that includes access to phone hardware, settings, and user data. Access to privacy- and security-relevant parts of the API is controlled with an install-time application permission system. We study Android applications to determine whether Android developers follow least privilege with their permission requests. We built Stowaway, a tool that detects overprivilege in compiled Android applications. Stowaway determines the set of API calls that an application uses and then maps those API calls to permissions. We used automated testing tools on the Android API in order to build the permission map that is necessary for detecting overprivilege. We apply Stowaway to a set of 940 applications and find that about one-third are overprivileged. We investigate the

causes of over privilege and find evidence that developers are trying to follow least privilege but sometimes fail due to insufficient API documentation.

Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing

Smartphone security research has produced many useful tools to analyze the privacy-related behaviors of mobile apps. However, these automated tools cannot assess people's perceptions of whether a given action is legitimate, or how that action makes them feel with respect to privacy. For example, automated tools might detect that a blackjack game and a map app both use one's location information, but people would likely view the map's use of that data as more legitimate than the game. Our work introduces a new model for privacy, namely privacy as expectations. We report on the results of using crowdsourcing to capture users' expectations of what sensitive resources mobile apps use. We also report on a new privacy summary interface that prioritizes and highlights places where mobile apps break people's expectations. We conclude with a discussion of implications for employing crowdsourcing as a privacy evaluation technique.

Android platform-based individual privacy information protection system

With the popularity of mobile phones with Android platform, Android platform-based individual privacy information protection has been paid more attention to. In consideration of individual privacy information problem after mobile phones are lost, this paper tried to use SMS for remote control of mobile phones and providing comprehensive individual information protection method for users and completed a mobile terminal system with self-protection characteristics. This system is free from the support of the server and it can provide individual information protection for users by the most basic SMS function, which is an innovation of the system. Moreover, the protection mechanism of the redundancy process, trusted number mechanism and SIM card detection mechanism are the innovations of this system. Through functional tests and performance tests, the system could satisfy user functional and non-functional requirements, with stable operation and high task execution efficiency.

A framework for static detection of privacy leaks in android

applications We report on applying techniques for static information flow analysis to identify privacy leaks in Android applications. We have crafted a framework which checks

with the help of a security type system whether the Dalvik bytecode implementation of an Android app conforms to a given privacy policy. We have carefully analyzed the Android API for possible sources and sinks of private data and identified exemplary privacy policies based on this. We demonstrate the applicability of our framework on two case studies showing detection of privacy leaks.

CHAPTER 3

MACHINE LEARNING

Machine Learning is a system of computer algorithms that can learn from example through self-improvement without being explicitly coded by a programmer. Machine learning is a part of artificial Intelligence which combines data with statistical tools to predict an output which can be used to make actionable insights.

The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input and uses an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.



Fig:3.1 Traditional Programming

Machine learning is supposed to overcome this issue. The machine learns how the input and output data are correlated and it writes a rule. The programmers do not need to write new rules each time there is new data. The algorithms adapt in response to new data and experiences to improve efficacy over time.



Fig:3.2 Machine Learning

Working of Machine Learning

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.

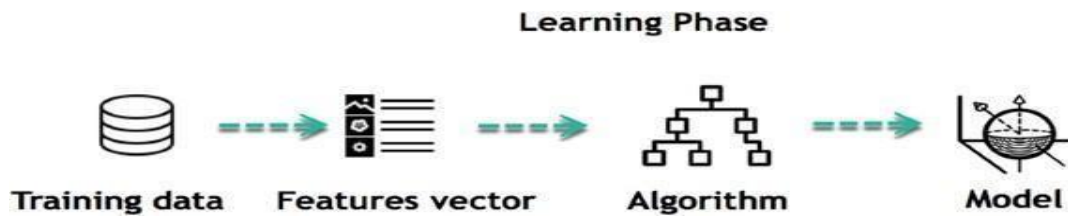


Fig: 3.3 Learning Phase of ML

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring

When the model is built, it is possible to test how powerful it is on never-seen- before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

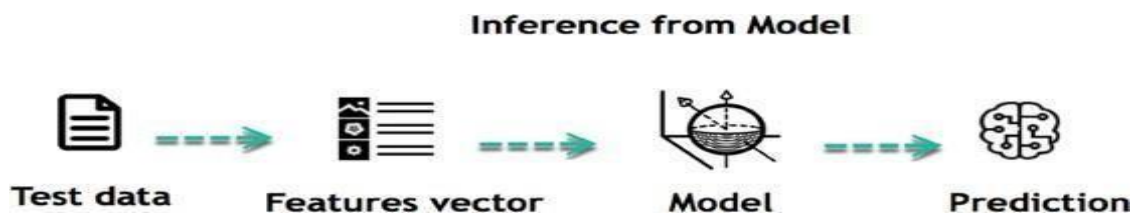


Fig: 3.4 Inference Model of ML

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm

8. Loop 4-7 until the results are satisfying

9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

Machine Learning Algorithms

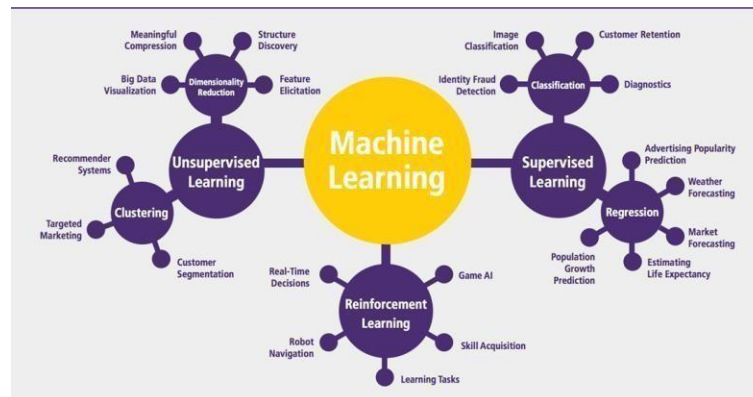


Fig: 3.5 Machine Learning Algorithm

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above Machine learning example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

How to Choose Machine Learning Algorithm

There are plenty of machine learning algorithms. The choice of the algorithm is based on the objective. In the Machine learning example below, the task is to predict the type of flower among the three varieties. The predictions are based on the length and the width of the petal. The picture depicts the results of ten different algorithms. The picture on the top left is the dataset. The data is classified into three categories: red, light blue and dark blue. There are some groupings. For instance, from the second image, everything in the upper left belongs to the red category, in the middle part, there is a mixture of uncertainty and light blue while the bottom corresponds to the dark category. The other images show different algorithms and how they try to classify the data.

Challenges and Limitations of Machine Learning

The primary challenge of machine learning is the lack of data or the diversity in the dataset. A machine cannot learn if there is no data available. Besides, a dataset with a lack of diversity gives the machine a hard time. A machine needs to have heterogeneity to learn meaningful insight. It is rare that an algorithm can extract information when there are no or few variations. It is recommended to have at least 20 observations per group to help the machine learn. This constraint leads to poor evaluation and prediction.

Application of Machine Learning

Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

- Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

Government organization

- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

Advantages of Machine learning

Machine learning is the best tool so far to analyze, understand and identify a pattern in the data. One of the main ideas behind machine learning is that the computer can be trained to automate tasks that would be exhaustive or impossible for a human being. The clear breach from the traditional analysis is that machine learning can take decisions with minimal human intervention.

Take the following example for this ML tutorial; a retail agent can estimate the price of a house based on his own experience and his knowledge of the market.

A machine can be trained to translate the knowledge of an expert into features. The features are all the characteristics of a house, neighborhood, economic environment, etc. that make the price difference. For the expert, it took him probably some years to master the art of estimate the price of a house. His expertise is getting better and better after each sale.

For the machine, it takes millions of data, (i.e., example) to master this art. At the very beginning of its learning, the machine makes a mistake, somehow like the junior salesman. Once the machine sees all the example, it got enough knowledge to make its estimation. At the same time, with incredible accuracy. The machine is also able to adjust its mistake accordingly.

Most of the big company have understood the value of machine learning and holding data. McKinsey have estimated that the value of analytics ranges from \$9.5 trillion to \$15.4 trillion while \$5 to 7 trillion can be attributed to the most advanced AI techniques.

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Data mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner

accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Optimization

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples.

Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples).

Generalization

The difference between optimization and machine learning arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples. Characterizing the generalization of various learning algorithms is an active topic of current research, especially for deep learning algorithms.

Statistics

Machine learning and statistics are closely related fields in terms of methods, but distinct in their principal goal: statistics draws population inferences from a sample, while machine learning finds generalizable predictive patterns. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics. He also suggested the term data science as a placeholder to call the overall field.

Leo Breiman distinguished two statistical modeling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine

learning algorithms like Random Forest.

Some statisticians have adopted methods from machine learning, leading to a combined field that they call statistical learning.

Approaches

Types of learning algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised learning

A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white.

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Types of supervised learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms,

therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

Semi-supervised learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically

represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

Self learning

Self-learning as a machine learning paradigm was introduced in 1982 along with a neural network capable of self-learning named crossbar adaptive array (CAA). It is a learning with no external rewards and no external teacher advice. The CAA self-learning algorithm computes, in a crossbar fashion, both decisions about actions and emotions (feelings) about consequence situations. The system is driven by the interaction between cognition and emotion. The self-learning algorithm updates a memory matrix $W = ||w(a,s)||$ such that in each iteration executes the following machine learning routine:

In situation s perform an action a ;

Receive consequence situation s' ;

Compute emotion of being in consequence situation $v(s')$;

Update crossbar memory $w'(a,s) = w(a,s) + v(s')$.

It is a system with only one input, situation s , and only one output, action (or behavior) a . There is neither a separate reinforcement input nor an advice input from the environment. The backpropagated value (secondary reinforcement) is the emotion toward the consequence situation. The CAA exists in two environments, one is the behavioral environment where it behaves, and the other is the genetic environment, wherefrom it initially and only once receives initial emotions about situations to be encountered in the behavioral environment. After receiving the genome (species) vector from the genetic environment, the CAA learns a goal-seeking behavior, in an environment that contains both desirable and undesirable situations.

Feature learning

Several learning algorithms aim at discovering better representations of the inputs provided during training. Classic examples include principal components analysis and cluster analysis. Feature learning algorithms, also called representation learning

algorithms, often attempt to preserve the information in their input but also transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions. This technique allows reconstruction of the inputs coming from the unknown data-generating distribution, while not being necessarily faithful to configurations that are implausible under that distribution. This replaces manual feature engineering, and allows a machine to both learn the features and use them to perform a specific task.

Feature learning can be either supervised or unsupervised. In supervised feature learning, features are learned using labeled input data. Examples include artificial neural networks, multilayer perceptrons, and supervised dictionary learning. In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization and various forms of clustering.

Anomaly detection

In data mining, anomaly detection, also known as outlier detection, is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items represent an issue such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are referred to as outliers, novelties, noise, deviations and exceptions.

In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts of inactivity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular, unsupervised algorithms) will fail on such data unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro-clusters formed by these patterns.

Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal, by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherently unbalanced nature of outlier

detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set and then test the likelihood of a test instance to be generated by the model.

Machine learning Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

Artificial neural networks

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

Decision trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.

Regression analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization (mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel, logistic regression (often used in statistical classification) or even

kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

Training models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from biased data can result in skewed or undesired predictions. Algorithmic bias is a potential result from data not fully prepared for training.

CHAPTER-4

EXISTING AND PROPOSED SYSTEM

EXISTING SYSTEM

The GP-PP is a Nonexclusive Consents Security intrusive authorizations model is useful to describe the assents into Ordinary and Insurance prominent assents. The existing method a foolish system- clients toward pick. Applications hazardous toward present. The GP-PP model requests an application as security prominent expecting bigger piece of the approvals referenced are insurance meddlesome. Hence, clients can finish up which set of approvals can be horrendous. We support the GP-PP model to check whether the model requests an application in view of approval sets that the application requests.

DISADVANTAGES OF EXISTING SYSTEM

Malicious application is introduced by means of a connection, Applications request that authorizations get to superfluous data like contacts, display, video, Redirect the program into another site, Security is low.

PROPOSED SYSTEM

In the proposed methodology first we need to identify the list of third party installed on the applications. After that for all each application extract the complete list of permissions followed by identify android protection level of each permission, i.e Normal or Dangerous of each app. Take android app permissions dataset. Need to apply classification algorithm to identify the harmful applications apply. Monitor the accuracy details using spam classification given by it also time required for execution. Final output we can see the accuracy among various harmful in addition to normal apps Classifiers are analyzed.

ADVANTAGES OF PROPOSED SYSTEM

Security is high.

Working of Naive Bayes algorithm

We should take a model. Underneath I have a rundown of climate preparing full intent on playing (which offers you the chance to play). Presently we want to list assuming the players will play contingent upon the climate. To do this, follow the means underneath. Stage 1: Change the data put on the table Step 2: Make a table that is just about as accessible as could be expected = 0.29, playable 0.64, etc. Stage 3: Now utilize the Naive Bayes correlation with ascertain the probabilities of every class. The most probable stage is the answer for the theory.

CHAPTER- 5

SYSTEM ARCHITECTURE

5.1 SYSTEM ARCHITECTURE

In the mobile phones, the applications will be listed. There will be two process steps they are, the applications will be extracted using XML code and the applications will be decompiled using Java code. In XML after data analyzing there will be many permissions will be generated and then in java after data analyzing there will be API function calls. The android apps will be scanned using decision tree algorithm, After scanning process the apps will be classified into malicious and non malicious. Using decision tree algorithm if it is yes then app is malicious and if it is no then the app is not malicious. By this architecture we are representing our project model

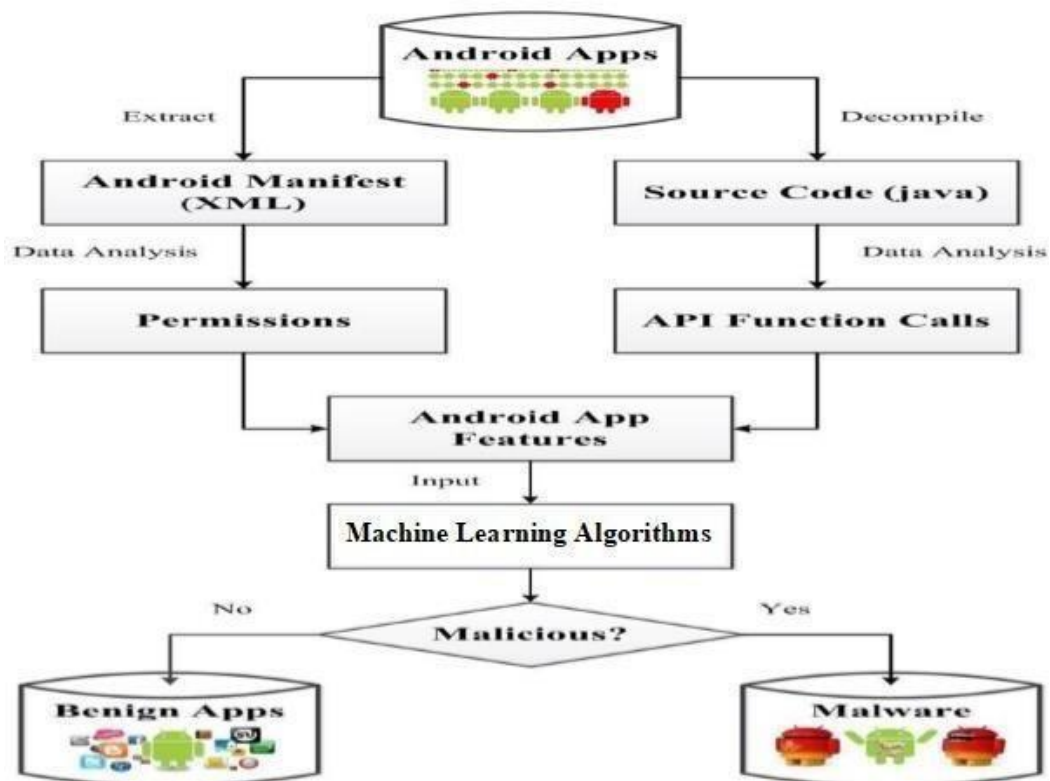


Figure 5.1 Block Diagram for System Architecture

SYSTEM

REQUIREMENTS

HARDWARE

REQUIREMENTS:

System : Pentium Dual Core.
Hard Disk : 120 GB.
Monitor : 15"LED
Input Devices : Keyboard, Mouse
Ram : 1GB.

SOFTWARE REQUIREMENTS:

Operating system : Windows 7.
Coding Language : Android, Java
Toolkit : Android 2.3 Above
IDE : Eclipse/Android Studio

PENTIUM DUAL CORE

The Pentium Dual-Core brand was used for mainstream x86-architecture microprocessors from Intel from 2006 to 2009 when it was renamed to Pentium. The processors are based on either the 32-bit *Yonah* or (with quite different microarchitectures) 64-bit *Merom-2M*, *Allendale*, and *Wolfdale-3M* core, targeted at mobile or desktop computers.



Fig 5.2 Pentium Dual Core

JAVA

The Java language has undergone several changes since JDK 1.0 as well as numerous additions of classes and packages to the standard library. Since J2SE 1.4, the evolution of the Java language has been governed by the Java Community Process (JCP), which uses Java Specification Requests (JSRs) to



Fig 5.3 Java

WINDOWS

Microsoft Windows is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft. Each family caters to a certain sector of the computing industry. Active Windows families include Windows NT and Windows Embedded; these may encompass subfamilies, e.g. Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x, Windows Mobile and Windows Phone.



Fig 5.4 Windows

ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.



Fig 5.5 Android Studio

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touch screen mobile devices such as smart phones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear). The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touch screen input, it has also been used in game consoles, digital cameras, regular PCs, and other electronics. As of 2015, Android has the largest installed base of all operating systems. As of July 2013, the Google Play store has had over one million Android applications ("apps") published, and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of them create applications for Android; another 2015

survey found that 40% of full-time professional developers see Android as the "priority" target platform, which is more than iOS (37%) or other platforms.

At Google I/O 2014, the company revealed that there were over one billion active monthly Android users, up from 538 million in June 2013. Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software developed and licensed by Google. Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices. Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially released running other operating systems. The operating system's success has made it a target for patent litigation as part of the so-called "smartphone wars" between technology companies.

MODULES

- Login/Registration
- Android Application Scan
- Malware Detection
- Android Permission Check
- Privacy Policy Database

LOGIN MODULE

This is the first activity that opens when user open the website. User needs to provide a correct contact number and a password, which user enters while registering, in order to login into the website. If information provided by the user matches with the data in the database table then user successfully login into the website else message of login failed is displayed and user need to reenter correct information. A link to the register activity is also provided for registration of new users.

REGISTRATION MODULE

A new user who wants to access the app needs to register first before login. By clicking on register button in login activity, the register activity gets open. A new user registers by entering full name, password and contact number. A user needs to enter password again in confirm password textbox for confirmation. When user enters the information in all textboxes, on the click of register button, the data is transferred to database and user is directed to login activity again. Registered user then needs to login in order to access the app. Validations are applied on all the textboxes for proper functioning of the app. Like information in each textbox is must that is each textbox, either it is of name, contact, password or confirm password, will not be empty while registering. If any such textbox is empty app will give message of information is must in each textbox. Also data in password and confirm password fields must match for successful registration. Another validation is contact number must be valid one that is of 10 digits. If any such validation is violated then registration will be unsuccessful and then user needs to register again. Message that app will display when one of the fields is empty. If all such information is correct user will be directed to login activity for login into the app.

Application Download

In this module, many applications are already download in mobile phone through play store. After installation just find the malware present in various applications.

Android Application Scan

In this module, many application's are already download in mobile phone through play store. After installation just find the malware present in various applications. Malware of this kind can't be detected by victimization the quality signatures approach or by applying regular static or dynamic analysis strategies. The detection is performed on the basis of application's network traffic patterns solely. For each application, a model representing its specific approach pattern is learned regionally (i.e., on the device). Semi-supervised machine-learning methods are used for learning the normal behavioral patterns and for detecting deviations from the application's expected behavior. These methods were implemented and evaluated on Android devices.

Malware Detection

In this proposed system in order to improve the security of the mobile apps one methodology is proposed which will evaluate the mobile applications security based on the cloud computing platform and data mining. The task of identifying malware will be categorized into analysis, classification, detection and ultimate containment of malware. Several classification techniques are utilized in order to classify malware in keeping with their instances and this has created it potential to acknowledge the sort and activities of a malware and new variant. For such kinds of threats to mobile devices there should be some security mechanism to be implemented. Analysis of malware has got to do with distinctive the instances of malware by completely different classification schemes by exploitation the attributes of well-known malware characteristics. Here also a prototype system named Malware detection system is presented to identify the mobile app's virulence or benignancy. Malware detection has to do with the quick detection and validation of any instance of malware in order to prevent further damage to the system. The last a part of the work is containment of the malware, which involves effort at stopping escalation and preventing further damages to the system. A commercial antivirus uses signature based mostly technique wherever the information should be often updated so as to possess the newest virus information detection mechanisms. However, the zero-day malicious exploit malware can not be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file to detect abnormous file segments.

Android Permission Check

A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene. If an app needs to use resources or information outside of its own sandbox, the app has to request the appropriate permission. You declare that your app needs a permission by listing the permission in the app manifest and then requesting that the user approve each permission at runtime. If your app needs a dangerous permission, you must check whether you have that permission every time you perform an operation that requires

that permission. The system's behavior after you declare a permission depends on how sensitive the permission is. Some permissions are considered "normal" so the system immediately grants them upon installation. Other permissions are considered "dangerous" so the user must explicitly grant your app access. For more information about the different kinds of permissions, see Protection levels. If a user keeps trying to use functionality that requires a permission, but keeps denying the permission request, that probably means the user doesn't understand why the app needs the permission to provide that functionality.

APPLICATION OF JAVA

Java is widely used in every corner of world and of human life. Java is not only used in software but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java:

1. Developing Desktop Applications
2. Web Applications like LinkedIn.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems
5. Robotics and games etc.

FEATURES OF JAVA

The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language. Those features are;

1) Simple

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

Eg: Pointers and Operator Overloading are not there in java but were an important part of C++

2) Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model.

3) Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

4) Platform Independent

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language. On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.

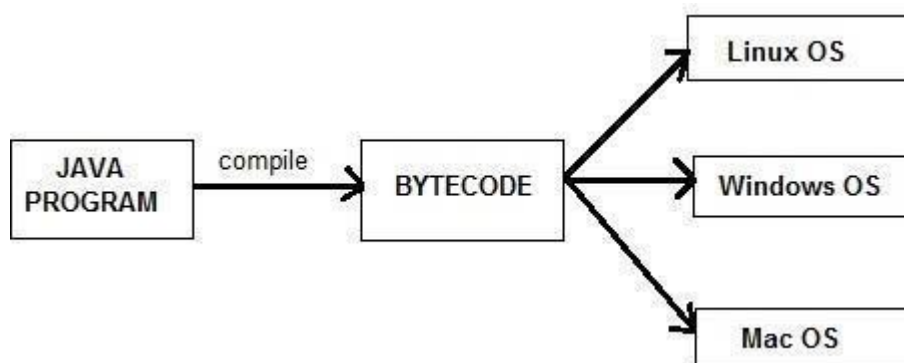


Fig 5.6 Block Diagram of Platform Independent

5) Secure

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

6) Multi-Threading

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

7) Architectural Neutral

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

8) Portable

Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

10) High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But Java enables high performance with the use of just-in-time compiler.

COLLECTION FRAMEWORK

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.

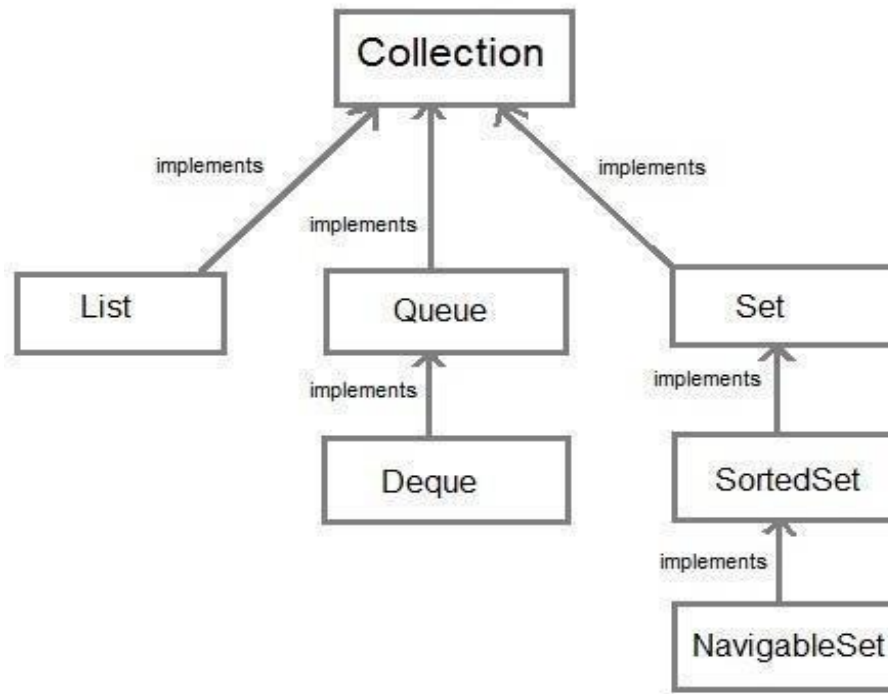


Fig 5.7 Collection of Frame Work

SYSTEM DESIGN AND TESTING PLAN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

USE CASE DIAGRAM

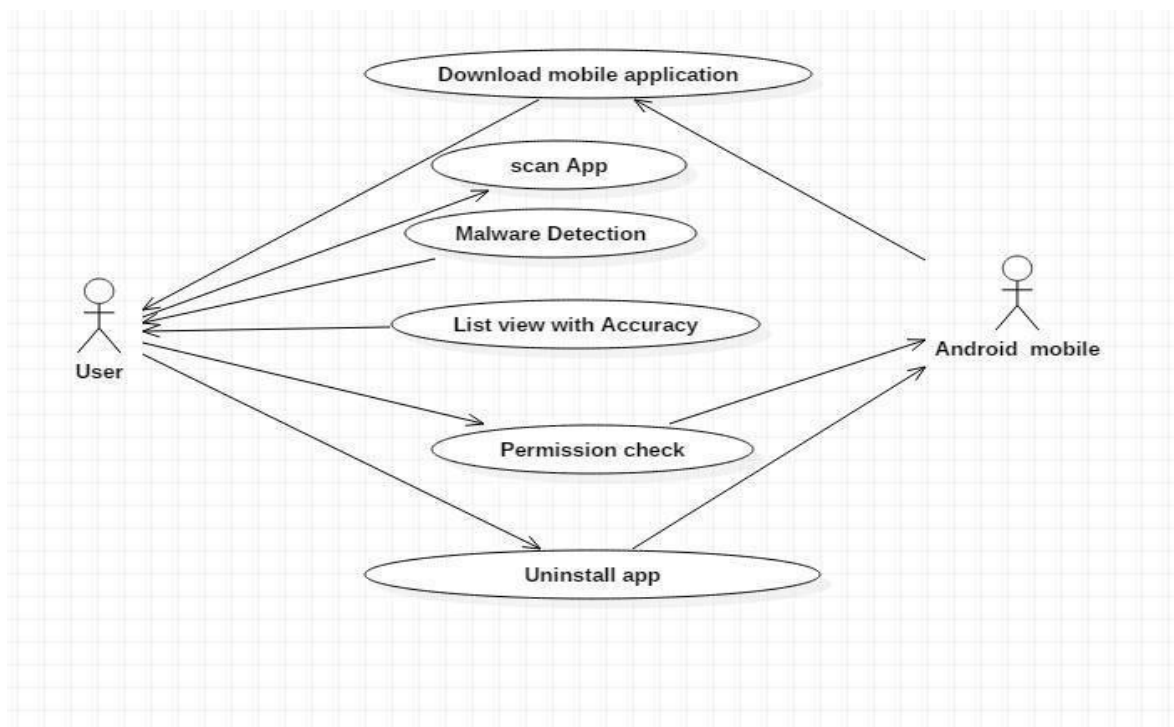


Fig 5.8 Block Diagram of Use Case Diagram

ACTIVITY DIAGRAM

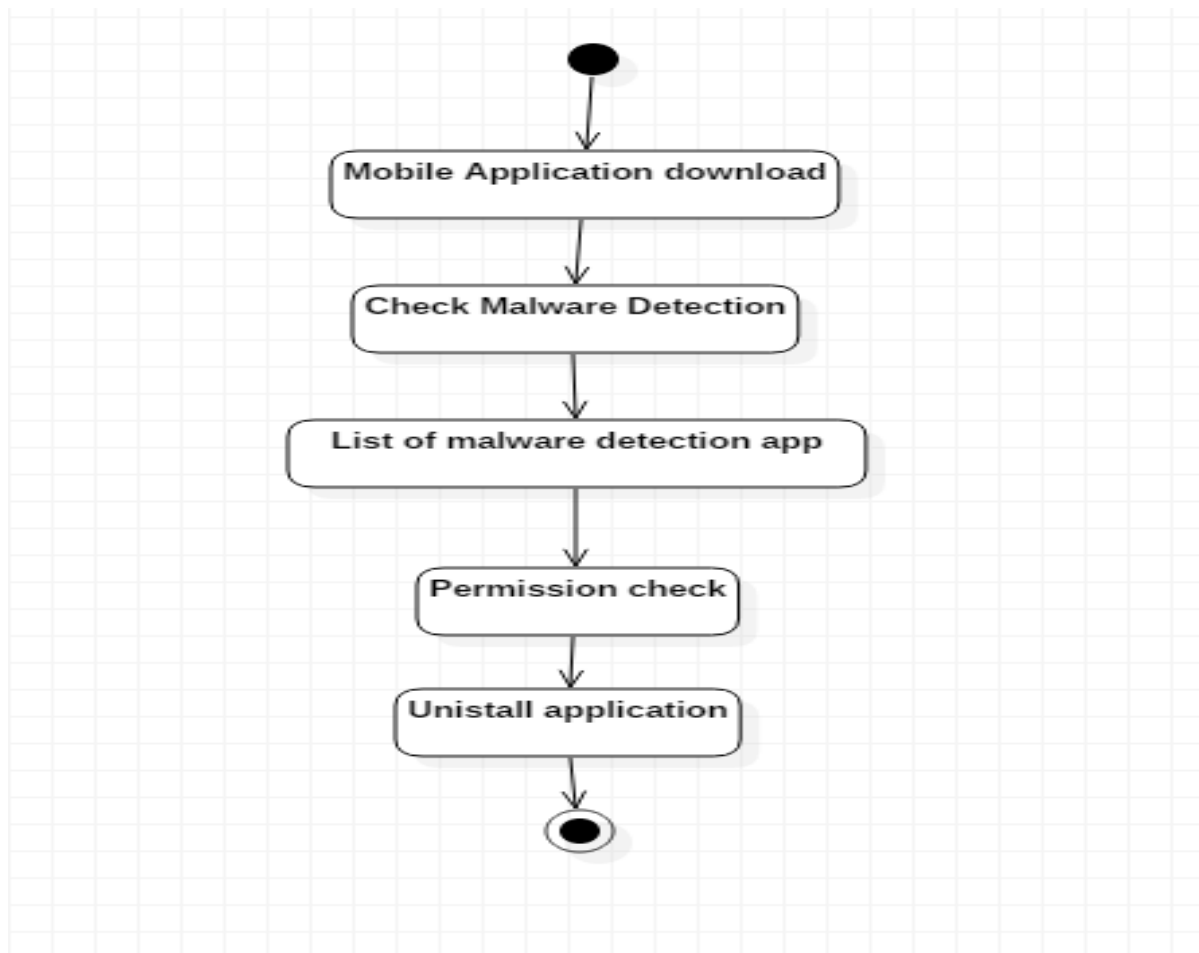


Fig 5.9 Activity Diagram

SEQUENCE DIAGRAM

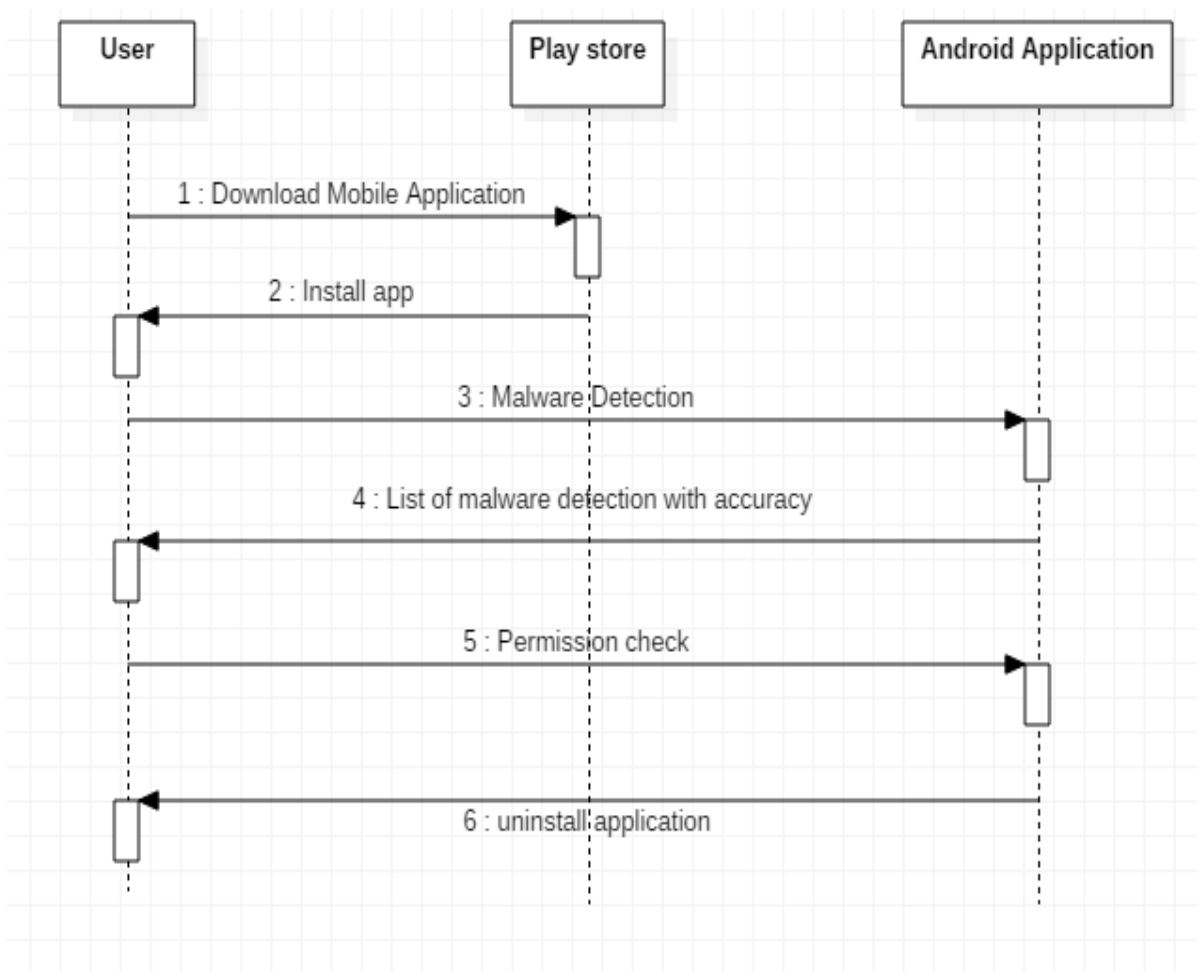


Fig 5.10 Sequence Diagram

COLLABORATION DIAGRAM

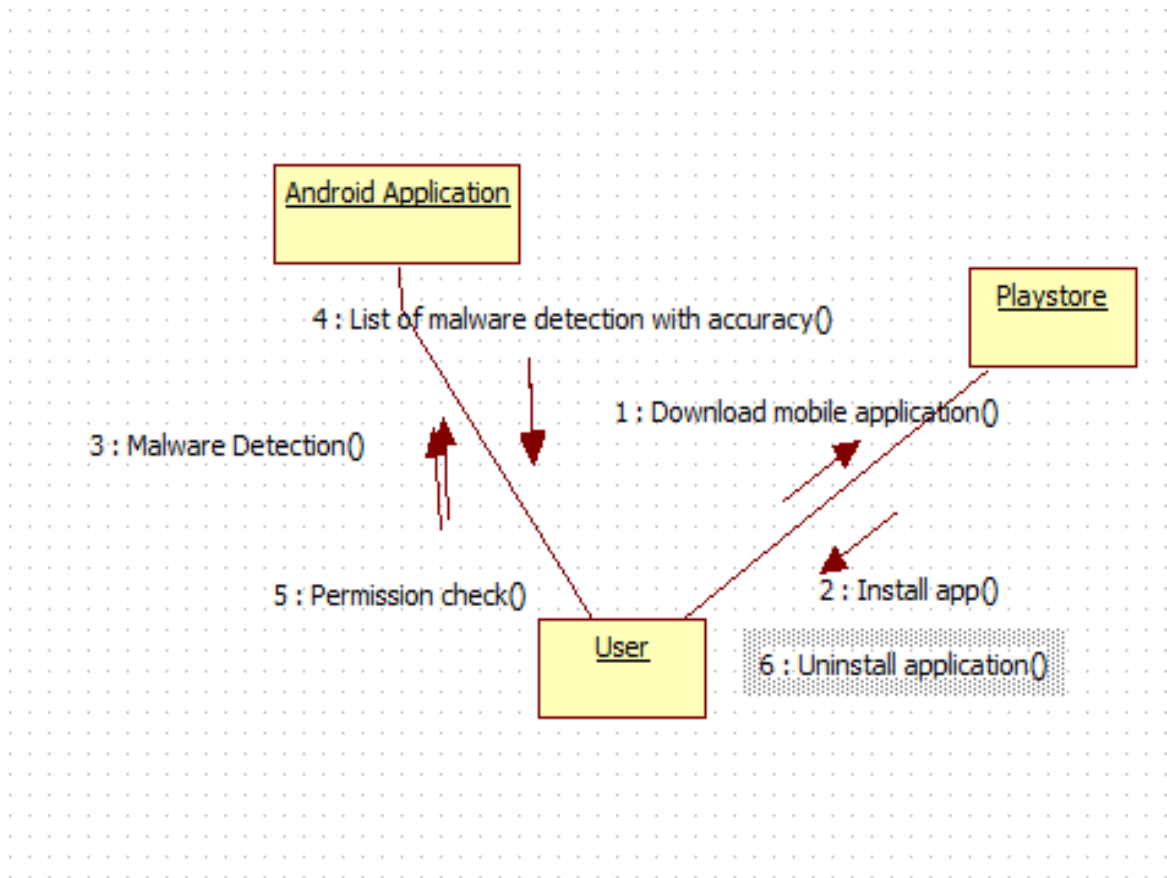


Fig 5.11 Collaboration Diagram

MODULES

- Application Download Module
- Android Application Scan
- Malware Detection
- Android Permission Check
- Privacy Policy Database

Application Download

In this module, many applications are already download in mobile phone throughplay store. After installation just find the malware present in various applications.

Android Application Scan

In this module, many applications are already download in mobile phone through play store. After installation just find the malware present in various application's. Malware of this kind can't be detected by victimization the quality signatures approach or by applying regular static or dynamic analysis strategies.

The detection is performed on the basis of application's network traffic patterns solely. For each application, a model representing its specific approach pattern is learned regionally (i.e., on the device).

Semi- supervised machine-learning methods are used for learning the normal behavioural patterns and for detecting deviations from the application's expected behaviour. These methods were implemented and evaluated on Android devices.

Malware Detection

In this proposed system in order to improve the security of the mobile apps one methodology is proposed which will evaluate the mobile applications security based on the cloud computing platform and data mining. The task of identifying malware will be categorised into analysis, classification, detection and ultimate containment of malware. Several classification techniques are utilized in order to classify malware in keeping with their instances and this has created it potential to acknowledge the sort and activities of a malware and new variant. For such kinds of threats to mobile devices there should be some security mechanism to be implemented. Analysis of malware has got to do with distinctive the instances of malware by completely different classification schemes by exploitation the attributes of well-known malware characteristics. Here also a prototype system named Malware detection system is presented to identify the mobile app's virulence or benignancy. Malware detection has to do with the quick detection and validation of any instance of malware in order to prevent further damage to the system. The last a part of the work is containment of the malware, which involves effort at stopping escalation and preventing further damages to the system. A commercial antivirus uses signature based mostly technique wherever the information should be often updated so as to possess the newest virus information detection mechanisms. However, the zero-day malicious exploit malware cannot be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file.

Android Permission Check

A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene. If an app needs to use resources or information outside of its own sandbox, the app has to request the appropriate permission. You declare that your app needs a permission by listing the permission in the app manifest and then requesting that the user approve each permission at runtime. If your app needs a dangerous permission, you must check whether you have that permission every time you perform an operation that requires that permission. The system's behavior after you declare a permission depends on how sensitive the permission is. Some permissions are considered "normal" so the system immediately grants them upon installation. Other permissions are considered "dangerous" so the user must explicitly grant your app access. For more information about the different kinds of permissions, see Protection levels. If a user keeps trying to use functionality that requires a permission, but keeps denying the permission request, that probably means the user doesn't understand why the app needs the permission to provide that functionality.

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposals put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that it has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision

branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER-6

RESULTS

In this proposed system for enhancing mobile application security, one method is proposed to evaluate the security of mobile applications based totally on cloud computing and statistics mining. The venture of malware detection will be divided into the evaluation, class, detection and final containment of the malware. Several methods of malware class are used to become aware of their times, and this makes it possible to discover the sort and sports of malware and new editions. To cope with such threats to cellular devices, some type of security mechanism have to be applied. Malware evaluation is ready distinguishing malware times based on absolutely special category schemes by theuse of specific malware characteristics. Malware detection is all approximately speedy detecting and blockading any instances of malware from causing in addition damage to the machine. The ultimate part of the activity includes malware containment, whichincludes efforts to stop its unfold and prevent harm to the gadget. However, no-day malware can't be detected by using an antivirus-primarily based antivirus, but by using the use of a mathematical analysis of the contents of a binary record to detect anomalous file segments.

OUTPUT SCREEN SHOTS

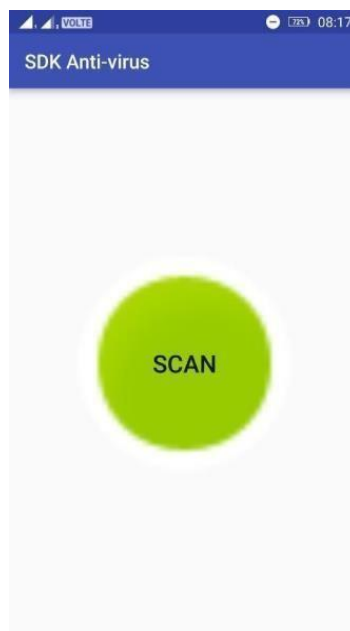


Fig 6.1 Initial Process

SCANNING PROCESS



Fig 6.2 Scanning Process

MALWARE DETECTION PROCESS

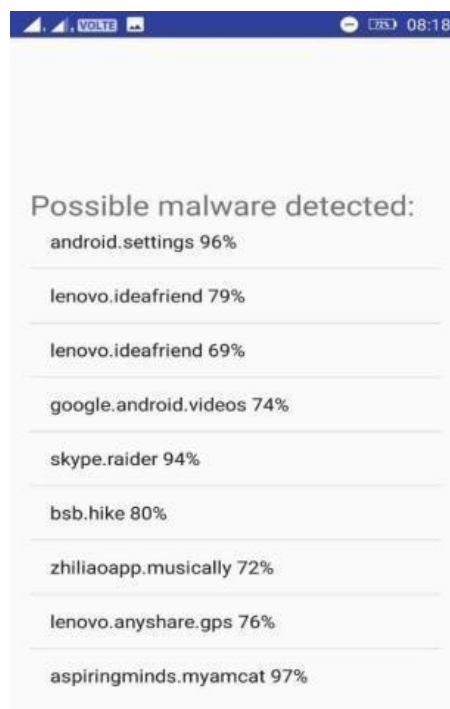


Fig 6.3 Malware Detection Process

EXISTING VS PROPOSED METHODOLOGY

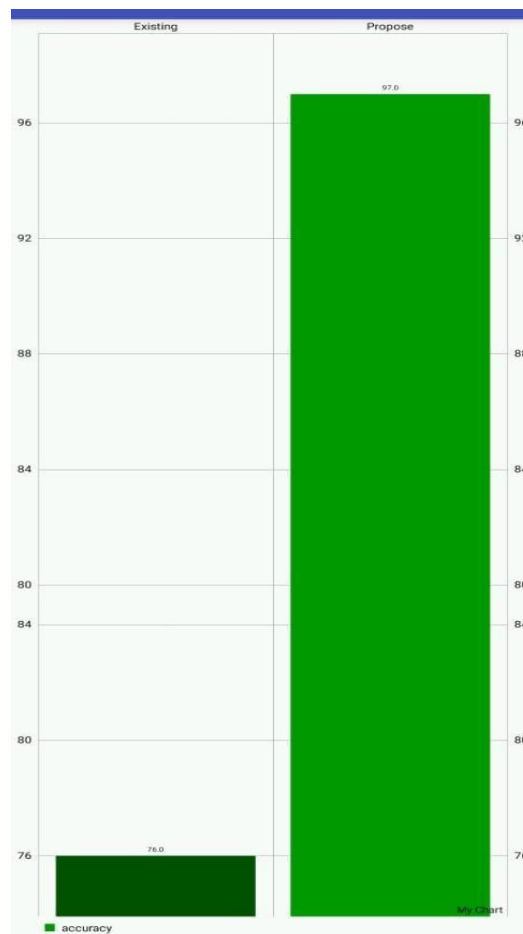


Fig 6.4 Existing vs Proposed Methodology

CONCLUSION

Our project using machine learning algorithms in the detection and classification of malware has become an important aspect of modern cybersecurity. With the increasing number and complexity of malware attacks, traditional signature-based detection methods have become less effective, and machine learning techniques have emerged as a promising solution. Machine learning algorithms can analyze vast amounts of data and learn from it to identify patterns and anomalies that can be used to detect and classify malware with a high degree of accuracy. Some of the popular machine learning techniques used for malware detection and classification include artificial neural networks, decision trees, random forests, and support vector machines. However, there are still challenges associated with the use of machine learning in malware detection and classification. One of the biggest challenges is the ability of malware authors to evade detection by modifying their code to make it look different from previously identified malware. Additionally, there is a risk of false positives, where legitimate software is flagged as malware. Overall, this project has proven to be an effective and efficient method for detecting and classifying malware, and ongoing research is focused on improving the accuracy and reliability of these techniques. As the threat landscape continues to evolve, the development of more advanced and sophisticated machine learning algorithms will be necessary to stay ahead of the constantly changing tactics of malware detection.

REFERENCES

- [1] Aerima, S. Y., Sezer, S., & McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security*, 8(1), 25-36.
- [2] Chia, P. H., Yamamoto, Y., & Asokan, N. (2012, April). Is this app safe, a large scale study on application permissions and risk signals. In *Proceedings of the 21st international conference on World Wide Web* (pp. 311-320). ACM.
- [3] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011, October). Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 627-638). ACM.
- [4] Lin, J., Amini, S., Hong, J. I., Sadeh, N., Lindqvist, J., & Zhang, J. (2012, September). Expectation and purpose. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (pp. 501-510). ACM.
- [5] Mann, C., & Starostin, A. (2012, March). A framework for static detection of privacy leaks in android applications. In *Proceedings of the 27th annual ACM symposium on applied computing* (pp. 1457-1462). ACM.
- [6] Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012, June). Android permissions: a perspective combining risk and benefits of the 17th ACM symposium on Access Control Models and Technologies (pp. 13-22). ACM.
- [7] Stevens, R., Gibler, C., Crussell, J., Erickson, J., & Chen, H. (2012, May). Investigating user privacy in android ad libraries. In *Workshop on Mobile Security Technologies (MoST)* (p. 10).
- [8] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., & Wang, X. S. (2013, November). AppIntent: In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 1043-1054). ACM.
- [9] Zhang, W., Li, X., Xiong, N., & Vasilakos, A. V. (2016). Android platform-based individual privacy information protection system. *Personal and Ubiquitous Computing*, 20(6), 875-884.
- [10] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012, February). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings the second ACM conference on Data and Applications Security and Privacy* (pp. 317-326).

