A Framework for Android Malware Detection and Classification

Author

Muhammad Murtaz Amir Naqvi

Registration Number

172537

Supervisor

Dr. Saad Rehman

DEPARTMENT OF COMPUTER AND SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

MAY, 2019

A Framework for Android Malware Detection and Classification

Author

Muhammad Murtaz Amir Naqvi

Registration Number

172537

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Computer Engineering

Thesis Supervisor:

Dr. Saad Rehman

Thesis Supervisor's Signature: _____

DEPARTMENT OF COMPUTER AND SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

MAY, 2019

Declaration

I, *Muhammad Murtaz Amir Naqvi* declare that thesis titled "A Framework for Android Malware Recognition and Classification" and the task presented in this research are my own work. This research has not been presented anywhere else for assessment purposes. The research taken from other sources has been acknowledged and referred properly.

Muhammad Murtaz Amir Naqvi,

172537

Copyright Notice

- Copyright in text of the thesis rests with the college student author. Copies (by any process) either completely, or of extracts, could be made only relating with instructions distributed by the writer and lodged in the Library of CEME, NUST. Details could be obtained by the Librarian. This site must form part of such copies produced. Further copies (by any process) might not be made without the permission (on paper) of the writer.

- The ownership of any intellectual property rights which might be described in this thesis is vested in CEME, NUST, at the mercy of any prior agreement to the contrary, and might not be made designed for use by third celebrations without the created permission of CEME, which will prescribe the conditions and conditions of such agreement.

- More info on the conditions less than which disclosures and exploitation might take place is obtainable from the Library of CEME, NUST, Rawalpindi.

Dedicated to *my exceptional parents* whose tremendous support and cooperation led me to this wonderful accomplishment.

Acknowledgments

I am thankful to my Superlative cherisher and sustainer to have got guided me throughout this just work at every stage and for each new thought that you setup in my own mind to boost it. Indeed, I possibly could have done nothing at all without Your priceless assistance and help. Whosoever helped me through the entire span of my thesis, whether my any or parents other specific was Your will, so non-e be worthy of praise but You indeed.

I actually is profusely thankful to my beloved parents who raised me when We were not with the capacity of jogging and continued to aid me throughout atlanta divorce attorneys department of my entire life.

I would like expressing special because of my supervisor Dr also. Saad Rehman for his help throughout my thesis and for all your classes which he has trained me during expert coursework. I can properly state that I haven't discovered any other engineering subject matter in such depth compared to the types which he has trained.

I'd like to pay special because of Brig also. Muhammad Mr and Abbas. Hassan Azwar for his tremendous cooperation and support. Each right time I acquired stuck in something, he developed the answer. Without his help I wouldn't have already been able to full my thesis. We appreciate his assistance and patience

through the entire whole thesis.

I'd like to thank Dr also. Ali Dr and Hassan. Farhan Riaz to be on my thesis evaluation and guidance committee. Finally, I'd like expressing my gratitude to all or any the individuals and institutions specifically C4I Directorate Pakistan Army General Mind Quarters Rawalpindi, which funded my research and approved the full total results after numerous evaluations.

Abstract

The android platform is that the fastest growing hand-held OS package. And it's really become the foremost appealing and practical objective of malevolent applications. Android malware growth has been increasing significantly in conjunction with increasing the guiltiness and variety of their developing techniques. Mobile malware is usually pernicious and therefore, on the increase, therefore having a trusted and quick detection system is very important to the users. Subtle Android malware make use of detection shunning ways to cover their malicious actions from analysis tools. In this evaluation, a brand-new recognition and characterization program for investigation significant deviations within the network behavior of a smart-phone program is proposed. The many objective of the proposed program is to protect mobile gadget users and cellular infrastructure companies from malicious applications through the use of simply nine visitors feature measurements. The proposed program isn't solely prepared to take notice of the malicious or masquerading apps, however could also determine them as general malware or particular malware (i.e. adware) on a mobile gadget. The proposed methodology demonstrated the common precision 94% a tagged dataset of mobile malware visitors with a whole lot of applications contains benign and twelve very different groups of each adware and general malware. Recent substantial evaluation on machine learning algorithms evaluate options from mischievous program and

use those choices to catalogue and find out unidentified malicious applications. This research condenses the progression of malware recognition techniques backed machine learning algorithms devoted to the Android Os's.

Keywords: *Xgboost, Adwares, Malwares, Andriod, Smart Phones, Operating System, Framework, Network*

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

APT	Advance Persistence Threat
Xgboost	Extreme Gradient Boosting
CNN	Convolutional Neural Network
KNN	K Nearest Neighbors
WEKA	Waikato Environment for Knowledge Analysis
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
CIC	Canadian Institute of Cyber Security
ML	Machine Learning
DM	Data Mining
PC	Personal Computer

SC	Smart Computers
ANN	Artificial Neural Networks
RMA	Rich Mobile Applications
HIDS	Host-Based Intrusion Detection System

Chapter 1

Introduction

In this technological era of whizz and efficacy everyone is more inclined towards user friendly portable devices. Smartphones emerged as a state of art facility to serve the purpose. Their powerful sensing, socializing and networking abilities make them unbeatable and stronghold of user's attraction. Additionally many surveys have validated their popularity repeatedly. Furthermore, Smartphones popularity lies in users' interest in Rich Mobile Applications (RMA). Everyone is more dependent on apps like Maps navigation, Food delivery etc. which deliver immersive and interactive users experiences (Knoernschild, 2016). "Technology and thinking will shift to a point where the experience will connect people with hundreds of edge devices", Gartner technology trends for 2019 [1]

With the light there comes the darkness, Cyber threats played a vital role to curse this blessing but failed to affect their increasing popularity. "As global cybercrime is estimated to cost $600 billion in 2018, the preferred choice of access for a majority of the world's population is a mobile device", McAfee Mobile Threat Report Q1, 2018. Though we bought technology home, but we must be aware of associated threats. "As we use technology to speed up the transfer of information,

it creates amazing opportunity and potentially greater risk", AON Cybersecurity report 2019 [2]. Increase in the fame of Smartphones have made them vulnerable to many malicious activities also made them open to many privacy and security treats. The renowned Anti-Virus company McAfee recently shared the mobile malware stats in McAfee Mobile Threat Report Q1, 2018. It says, "McAfee Labs detected over 16 million mobile malware infestations in the third quarter of 2017 alone, nearly doubling the number we saw a year earlier" [3].

The threats are not only limited to single users but also affecting the organizations by compromising their data and with cybercrimes. Currently the organizations are moving towards the defensive measure more rapidly. Based on M-Trends 2019, FIRE EYE Report: "From October 1, 2017, to September 30, 2018, the global median dwell time was 78 days. That means attackers are operating for just under three months, on average, before they are detected. That's roughly a quarter of the global median dwell time of 101 days in last year's report - a modest improvement." [4] This is a clear indication of awareness regarding cyberthreats within organizations. The report further adds on and concludes: "Organizations are getting better at detecting breaches quickly. Over the past eight years, dwell times have decreased significantly - from a median dwell time of 416 days in 2011 to 78 days in 2018" [4].

Building on this the victims are not limited to organizations, in fact it goes beyond, and countries are into Cyber Wars. The latest report by FireEye named, World War C based on best selling book and a Hollywood movie, it states: "Serious cyber-attacks are unlikely to be motiveless", said Martin Libicki, Senior Scientist at RAND Corp. "Countries carry them out to achieve certain ends, which tend to reflect their broader strategic goals. The relationship between the means chosen

and their goals will look rational and reasonable to them if not necessarily to us." [5]

Main reason behind Android Malwares is the opensource Android platform. The app store, Google play is most vulnerable platform. "In 2017 we saw an increase in malicious banking Trojans, such as the Android/Marcher malware, that take advantage of the auto install vulnerabilities in the Android platform". "It victimized millions of Google Play users by impersonating legitimate apps for video players, Flash players, games, and system utilities", says the McAfee Mobile Threat Report Q1, 2018. Mobile phones are known to be key to your digital environment. It includes your personal information, children's inheritance and everything connected. Due to immensely high market share, Android platform is always targeted. Whether they are banking trojans or cryptocurrency scams, your android phone is vulnerable to threats.

Criminal quest for money makes the android malware more appealing to them. "Cyber criminals load malicious code onto retailers' websites to steal shoppers' credit card details, with 4,800+ unique websites compromised on average every month" [3], says the 2019 Internet Security Threat Report. Android also allows its user to extend functionalists by allowing third party applications. This feature although is very attractive and useful but it also allows certian applications which are malicious in nature and can be damaging. "One of the most significant campaigns discovered by McAfee in late 2017 and in early in 2018 was Android Grabos. Grabos, a campaign that pushes unwanted apps on unsuspecting users is commonly known as pay-per download scam. In total, 144 apps on Google Play were identified and taken down. An estimated that 17.5 million global smart phone devices downloaded apps from the campaign before they were taken down."

Says the Mobile Malware report 2019 by McAfee.

Moving towards the detection and prevention techniques. In literature, previously analysis is done in the form of static, dynamic or a combination of both static and dynamic to detect and analyze Android malware. Comparatively, dynamic analysis is weaker than static as it only investigates the code which is being executed. As a result, to correctly assess whether an application can exhibit malicious behavior, dynamic analysis needs to execute a fairly significant portion of the paths through the program. For this we need to do the analysis of code and a test-case generation is to be done. This all can have as much as 70x overhead, to extract path constraints. In contrast, static analysis can efficiently analyze all the code in the application, but it is inherently imprecise, meaning that it may miss malicious behavior (false negatives) or falsely detect malicious behavior (false positives). The loss of precision is a cost of an analysis that can scale to an entire program, which often requires the analysis to make trade-offs that reduce precision in terms of context-sensitivity, flow-sensitivity or pointer disambiguation. Such imprecise analyses can make it difficult to accurately disambiguate malicious applications from benign ones as they may have similar behaviors superficially. Dynamic analyses are generally more precise than static analyses and can provide more information, but dynamic analyses need specific inputs to cause the application to execute the suspicious code.

1.1 Contribution

Our contribution to make the Cyber world a safer place is in the form of this research which propose a detection model for Android platform and serves as Android Malware Detection framework. It is a network traffic-based framework

which operates on new feature set to enhance the efficiency of traffic classifier. Our proposed detection method not only detect unknown Malwares but also it can label the type of Malwares. It demonstrates this by characterizing benign, adware and apps, with the efficient use of 76 network flow based features. To serve the purpose we used a labeled data of mobile Malware traffic with six lac instances including benign and malicious apps of 12 different malware families [6]. CIC flowmeter is used to capture network traffic and develop a versatile dataset [7]. The research work is organized hierarchically, firstly literature review is done by discussing the previous work and techniques used in malware detection for Android platform. Building on it further, all the techniques are discussed in this thesis and their performance was analyzed using software platform, WEKA [8]. Eventually, best possible machine learning based algorithm, XGBOOST [9] is chosen. Its performance was tested on publicly available CIC Dataset after the pre-processing. Pre-processing part is also explained in experimental section below. Lastly the thesis discusses the obtained results of malware detection and conclusion.

1.2 Thesis Organization

The research work in this document is divided into four basic parts i.e. Introduction, Background, Literature review, Proposed method and Experimentation setup along with results. Each part is explained in detail in the respective chapters.

Chapter 1 provides an overview of cyber threats related to Android Malwares. This section demonstrates the need of cybersecurity for Android platform by referring to the stats provided by major surveys and cyber security reports.

Chapter 2 provides a comprehensive Literature review of the research topic. All the renowned detection methods are referred, and a compact literature analysis of various techniques is done thoroughly. The explanation of attacks and risks associated are main deliverable of this section.

Chapter 3 propose a dynamic and versatile detection system for Android platform. The framework is evaluated for performance by different experimentation. Overall flow diagram of deployment and results related to every technique are included. The feature selection part is mainly focused with the deliverable of 76 selected features and the full data set is explained in detail. In this section, results are verified, and graphs are included for a clear comparison. Fundamentals of used algorithm is discussed in terms of its effectiveness.

Chapter 4 gives details about experimental setup and of testing is provided in this chapter. This chapter talks about results on testing data by evaluating different metrics. Also, test setup for live traffic is also given in this chapter.

Chapter 5 Finally, this chapter discuss about the contribution of this research also highlights some limitations and discuss about some future aspects.

Chapter 2

Literature Review

This chapter provides literature review and basic trends in the field of Android Malware detection. This chapter is divided into two main parts:

- Existing techniques
- Datasets Evaluation

2.1 Existing Techniques

Lot of research has been done in the are of malware recognition and characterization using features based on network traffic. In this sector 1st efforts was offered by Iland et al. in 2011 [10], in this research writers shown a light-weight strategy of discovering Google android malware and personal privacy breaches of consumer, through the network visitors evaluation. The writers carried out a series of managed tests. They initial produced the virtualized Google android products with a legitimate E-commerce thrid party application, and mocked consumer data, such as get in touch with data, accounts security passwords, internet browser background, and credit cards info and contaminated them with eighteen malware

examples; Then this research did an anlysis of the gathered network traffic for finding any leakage of user data and any attempts of accessing C&C (Command and Control Center); Lastly, for abnormal behavior classifciation of any malware they purposed two methods: 1) blacklisting of IP and DNS 2) Pattern matching and purposed 4 features which are flag of HTTP header, GET request, POST request pattern and content, POST request well structured identifiers. Nevertheless the proposed techniques have got many flaws. Initial, the blacklisting technique greatly depends on stationary malware behavior which needs regular upgrading over period. Also, efficient detection of complicated malwares such as fast-flux and botnet are even harder. The pattern matching techniques also need data in clear so it was not effective of encrypted traffic.

In 2012 Kuhnel and Meyer [11] proposed a malware detection technique by usage of a sensing application. This technique includes 30 families of malwares belonging to Android platform. These families was divided into four types RAT (Remote Access Tool), HTTP based, SMS based and Calls based. Also, a filtering component was added which works on user space of Architecture of Android which was used for network analysis and was controllable by the sensor app. The sensing app was also able to inform about malicious traffic, listing of events of the database, sending of blocked SMS and blocking preference changes. The achieved an accuracy of 95% this research claimed that by filtering outgoing and ingoing traffic they can detect malicious activities.

Tenenboim et al in 2013 [12], put forwarded a network traffic behavior analysis framework for detection of a new set of malwares present of Android app store having a unique feature of self-updating. This research also gives an analysis of available signature based, dynamic and static techniques on these malwares and

concluded that they are not effective. As a representative model they defined a specific pattern of the traffic generated by these malware applications and compared them with normal traffic patterns by using machine learning. They used nine features from there pervious [13] having five fixed intervals for calculation of minimum, maximum, average and standard deviation values the feature includes application concurrent connection count, count of TCP and UDP packets sent and received and count of TCP segments received. Their evolution was based on 15 applications having two version first version was real benign applications and second version includes repacked applications after injection of malware code of 5 real malware apps and 10 self-generated Trojans. There experimental results concluded that within 5 minutes they are able to identify malicious apps [12].

Dai et al. [14] in the same year, offered a brand new automated network profiler based on HTTP for uncovering Google android applications. The authors mentioned that, large number of applications based on HTTP/HTTPS are appearing every day, traditional technique of visitors category are no longer helpful for visitors evaluation. In this task, they initial installed hundreds Google android applications in a virtual environment and gathered the network traffic footprints. After that for extraction of fingerprints of apps, they suggested and created a light- weight method that can break the demand to "technique", "question" that can become break up to key-values and "web page" that can end up being damaged into "web page parts" and "filename". There are two restrictions of this suggested technique. Initial the program requirements a consumer seeds route when login can be included and secondly it cannot identify applications which possess no unique traffic behavior and usage of the same support system.

Researchers, Arora et al. [15] purposed a Google android malware recognition

technique based on network traffic features evaluation. For their experimental setup they setup a virtual Android environment having a public ip. For their dataset they consider 13 malware applications and captured there traffic. Sixteen features was selected based on their prior research. The features are average number of bytes sent, average packet size and average flow duration. After applying feature reduction out of 16, 7 features were selected the features are average count of packets sent, average count of packets received and average count of bytes received. In the test and evaluation section they divided their dataset to 3 dangerous amounts of Malware (high, moderate and low). The classifier proposed properly classified 45 of 48 instances of network samples with a precision of around 95%. The major deficiencies of this extensive research was the small size of dataset as well as not a broad range of malwares was considered.

Likewise, Shabtai et al. [16] presented a great recognition technique which uses network patterns of application for malware detection. The research purposed a novel model which was based on feature set like bytes of data send or received, state of network, mode of send or receive, total amount of time and last active or modification time in minutes also with aggregation functions like average, minimum, maximum and standard deviation, for representation of certain traffic features of each App. Also, they purposed a semi-automated ML model which can detect any deviations from normal app behavior. For evaluating there purposed model they selected 5 real malware applications along with 10 self-generated malware applications. For benign behavior analysis, original applications with no malware code injected were used and for the harmful purpose, applications with malware code injected was used. The results showed that particular classes of applications can become discovered by particular visitor's patterns. They also realized after that after starting execution, malware with self-updating capability possess different

diagnosable visitors patterns for a few minutes [16].

For improving Android abilities to tackle malicious attacks as well as APT (Advanced Persistent Threats) attacks. Li et al. [17] introduced a monitoring system based of network traffic. Visitors monitoring, visitor's anomaly acknowledgement, response digesting, and cloud storage space were the four main components of this system. In general the methodology was composed of following steps 1) Protocol parsing, 2) Extraction of features (Identification of the procedure, network connection start time, network connection end time, flow upward, down downward, src IP, dest IP , protocol, src port and dest port), 3) SVM as a classifier, 4) classification of network patterns to find unusual patterns, 5) through the relationship evaluation correlate the traffic with source application. The experimental results depicted that the system is effective in identifying Android malware with a low false positive rate.

Carrasquillo et al. [18] in 2014 conducted a study having focus on network based malware detection system. The study presented a combination of both signature based and network flows based traffic analysis. This research aim was to work on andriod platform on an VPN (virtual private network). The main motivation was providing a system that can generate alarms and visual analytics of any malicious activities for user and network administrator. This framework used an open source tool known as SNORT for signature matching and CISCO NETFLOW for network based detection. Following features was proposed by this framework source and destination IP address, source and destination port, packet size, sum of payload and time info.

An effective method for identification and classification of malwares was proposed by Wang et al. [19] in 2016. This method was the combination of traffic analysis

and ML algorithm based analysis. The method proposed, was focused on user experience, so to minimize resources usage, as Android devices are not resource intensive, they performed all the analysis on a server which was receiving all the mirrored traffic generated from Android apps, connected to a particular wireless access point. The authors used C4.5 decision tree based machine learning model and they achieved an accuracy of 98.2% and 99.7%, with FP rates of 5.1% and 1.9% respectively. In 2017, a study [20] shows that how a better accuracy and processing time can be achieved by prioritizing features extracted from network traffic. They deduced that how a minimum number of features can give better results. They presented that an accuracy of 85% to 100% can be achieved by just using 9 features out 22 features. Also, these reduced features can save a considerable amount of time in testing and training of the dataset. There results depicted that time reduced for testing of 230 apps was from 25.1 seconds to 17.3 seconds and for training of 300 apps it was 11.7 seconds to 5.8 seconds.

Besides study on Android malware analysis, the review was done on previous work associated with network traffic analysis. In this regard a study conducted by Karagiannis et al. [21] in 2005 known as BLINC. BLINC was a multilayer classification of transport layer based behavior of the host. It was one of the pioneer attempt to association of host with apps instead of characterization of flows by application. This association worked on three standards: (1) Network - src and dest IP's and ports characteristics analysis, (2) Social - get src & dest IP's, (3) Application - getting flows addition information which includes average packet size and transport layer info. Following applications was studied in this research gaming, data transfer, P2P, web, streaming, mail, chat and network management. The authors done this study in a dark mode to avoid any privacy issue which means that no payload access, no port number access and nothing other than the

provided information by collectors of current flow. A graphical representation of these patterns was presented. BLINC showed a classification of 80 to 90% of traffic with an accuracy of 95%.

Nguyen & Armitage [22] later in 2008, wrote a survey on the techniques which are using machine learning for traffic classification of internet traffic. This survey recorded papers from 2004 to early 2007. 18 significant papers was selected and discussed. The discussion includes following: (1) importance of operational network IP traffic classification, (2) Port and payload based classification limitations, (3) Classification accuracy metrics, (4) in operational IP networks what are key requirements for implication of ML based classifiers. Their research showed that for offline analysis most of the ML algorithms such as Decision Trees, Auto Class, Navie based etc. resulted in an accuracy of up to 95%. Also, they emphasis on the important of ML traffic classification problems and highlighted that a lot of new research can be done in this area.

A recent study done in 2016 by Bartos et al. [23] presented a novel technique involving supervised ML techniques for malware classification. This research didn't consider flows individually instead they are grouped in bags having similar flows grouped in each bag. Also, a robust technique was presented that combined the learning of representation process to learning of classifier process. A large corporate network was deployed and real time monitoring of HTTP traffic was carried out. There dataset includes 15M samples overall containing 43k samples belonging to malicious traffic. This system was able to detect unseen and new samples of malware traffic with precision on 90% means that out of 10, 9 alerts was malicious.

Habibi Lashkari et al. [7] in 2016, presented there work of classification of VPN and Non VPN traffic using time based network features. They proposed a flow

based classification method for characterization of encrypted and VPN traffic by using time based features. Additionally, they reduced the features set to reduce computational cost. Also, they released a dataset of there work which includes 14 different classes of encrypted and VPN traffic 7 from each category. The time based features used in this research was based on TCP and UDP flows. They achieved an accuracy of above 80%.

One more research in this regard was classification of Tor and Non Tor traffic based on time based features extracted from UDP and TCP flows was presented by Arash Habibi Lashkar el al. [24]. There contribution includes a feature set to classify and identify Tor traffic. In this research they concluded that only they have managed to classify Tor traffic using these time based features. Also, they researched on the optimal length of the flow for up to the mark classification they proposed that 15s is the optimal number for the length. Additionally, they compiled and published a label dataset in this regard this dataset have 8 different classes in relation to 8 different traffic captured including VOIP, P2P, file transfer, video streaming, audio streaming, mail, chat and browsing. They achieved an precision of about 80%.

Iman Sharafaldin el al. [25] [26], presented there work of using UDP and TCP flows based features for classification of intrusion attacks. They generated a new Intrusion detection dataset known as CICIDS2017 the dataset includes 13 classes of updated attacks and covers all necessary criteria of an IDS dataset. The dataset includes more than 80 features extracted from CICFlowMeter. This research also analyzed these dataset of different ML classifiers and presented a high precision.

2.2 Datasets Evaluation

This sections gives a comparative analysis of some of previous publicly available datasets of Android malwares and how the dataset used in this research is comprehensive and reliable for testing and validation of Android Malware detection systems.

The first dataset is known as Genome project [27]. Can be considered as a pioneer attempt in providing a publicly available dataset resleased in 2012. 1260 malware samples was included in this dataset, collected in a period of 2010 to 2011 from different lenders of Android malwares. Static analysis was performed in this project for defining behaviors of malwares. They evaluated the activation, installation and payload of the samples. The method was all about static analysis of API calls, malicious source code and permission lists. The dataset was also used on a real Android device for testing the effectiveness of already present malware detection tools and anti viruses.

After Genome project [27] Drebin [28] dataset was introduced in 2014. This dataset comprises of samples taken from 20 malware families and 123,453 benign samples taken in time period of 2010 to 2012. They evaluated there dataset by building a classification model extracted from static features. The features set comprises of permissions requested, components of hardware, Application components, network addresses (Disassembled code was used for extraction), filtered intents (used permissions,manifest files and restricted API calls was used for extraction).

The HCRL lab [29], presented four malwares dataset related to Android. These dataset includes: 1) AndroTracker 2) SAPIMMDS 3) Andro-Dumpsys 4) Andro-

Profiler. In these datasets AndroTracker dataset [30] was released in 2015. This dataset includes malware apps developed by the same creator and a certificate specific to a creator was used to develop malware apps each certificate had a unique key associated with it. They also presented a classifier which was based on similarity associated with each developer and features which are static in nature such as API calls which are suspicious, important permissions and intent.

SAPIMMDS [31] dataset released by the KISA (Korea Internet Security Agency) comprises of 906 malware instances from 13 malware categories and benign samples with count of 1776 taken from a period of March to December 2014. They used memory dumps techniques from bytecode of application for extracting API calls patterns of suspicious API calls. These API calls pattern was used to distinguish benign and malware traffic in this dataset.

Andro-Dumpsys dataset [32] released in 2016, presented a correlation of intent based attributes with malware centric attributes for purpose of classification. 1776 benign instances and 906 malware instances was include in this dataset. The feature vector of this dataset includes certificates serial numbers, call sequences of suspicious API calls, grants of permissions, intents and system commands used to execute forged files. They used memory dump techniques for profiling of patterns with respect to opcode and bytecode relationships. The implemented classification and detection on server where they calculated correlation of similarity between APK request and there feature set.

In dataset Andro-Profiler 2016 [33], In this research authors used an emulator to run malicions applications and during execution they done behaviour profiling by analyzing system calls and logs. A hybrid malware detection system based on client server model was main contribution of this research.

Table 2.1: Comparison between previous datasets

Ref.	Dataset	Benign Samples	Malware Samples	Static / Dynamic Features	Remarks
[27]	Genome Project	1,260	1260	Static	Static features can be tricked.
[28]	Drebin	123,453	5,560	Static	Reasonable samples but still uses static features.
[30]	AndroTracker	51,179	4,554	Static	Good sample but static features.
[31]	SAPIMMDS	906	1776	Static	Static features can be tricked.
[32]	Andro-Dumpsys	906	1776	Static	Static features can be tricked.
[33]	Andro-Profiler	643	8840	Dynamic	Less samples also uses emulator.

Another dataset known as Kharon dataset [34] presented in 2016, this dataset used AndroBlare tool1 on a Android device having 7 malicious apps installed on it. The tool tracked traffic flows between system objects including files, sockets and processes. They provided a graph readable to humans of each malware apps. A detailed documentation of the behavioral analysis was included in this dataset.

A part for above datasets AMD [35] is also a publicly available dataset presented in 2017. The dataset contains malware samples belonging to 4 categories and 71 families of malware was included. For evaluation they prioritized malicious components of malwares.

A summary of all previous datasets available is given in Table.

2.2.1 Summary

In summary, the above mentioned shows different approaches for Android malware dataset creation by different researchers. But, these dataset contains a lot

of deficiencies. These dataset lack a present of diverse malware categories and families also they lack a good amount of malware samples. Also, in general malware doesn't depict there true behaviors if not installed of a real android device although most of dataset contains some dynamic features but the real problem is that for capturing samples these datasets used an emulator or a virtual machine which most of advance malwares can detect and change their behavior after detection.

Malton [36] presented that, simulating android malware on real devices can result in overcoming of many anti emulator techniques and enable to get real behavioral analysis of malwares. Many malware applications are emulator sensitive they need user interaction to get activated if these malwares detect emulator they change their behavior accordingly. Moreover he mentioned that a complete dataset for malware detection should contains both static and dynamic features. However, the above mentioned datasets doesn't have a board set of features which are discrete features like network, battery, memory usage, memory dump and permission also continuous features like logs, system calls and API calls. Also there should be a balance between benign instances and malware instances for a trustworthy research. A high accuracy score will not be valid if distribution is not valid. As depicted in paper [6], the distribution of benign apps and malware apps according to Symantec is 80% to 20% in real world scenario.

To conclude briefly according to SISTR (Internet Security Threat Report) [37] all the dataset listed above fall short in aspects of real user interaction and real phone installation also a vast categories of malware are not considered in these dataset. Also, there is no balance between malware samples and benign samples.

Chapter 3

Proposed Methodology

This chapter provides a detailed overview of the proposed method. The chapter is organized in the following parts.

- Introduction
- Dataset
- Feature Extraction
- Proposed framework

3.1 Introduction

This work targets the classification of Android Malware based on certain network traffic characteristics. The Malwares are divided into three general categories i.e. Benign, Adware's and General Malwares. The network characteristics are based on TCP and UDP flows which results in 76 features which will be discussed in next section. The network characteristics for above malware categories are accumulated in a dataset which gives enough data for classification purposes.

The classification further consist of training and testing phases. During the first phase Xgboost was trained on 80% of dataset. Before using Xgboost some data cleansing is performed. After dataset cleansing dataset are fed into the Xgboost and after rigorously training, it is able to identify the Benign, Adwares and General Malwares which an accuracy of 94%.

3.2 Dataset

Android Malware are smart now they are now able to detect an occurrence of signature based malware detection and they change their conduct to avoid them. To overcome this a dataset is needed which is dynamic in nature for that a dataset set is used which is based on network based features of different malwares categories. AAGM2017 [6] is a dataset publicly available by CIC (Canadian Institute of Cyber Security). The dataset includes 1900 applications installed of a real smartphones with real user interaction was recorded in a semi automated way. The dataset includes three categories each category is analyzed below.

3.2.1 Benign Dataset

This category includes a collection of 1527 non malicious or benign apps collected in a time span of 2015 to 2016 from Google Play Store. Most popular apps were collected from different categories (i.e. top free, top new). These 1527 apps were then analyzed with Anti Virus (AV) items in Virustotal web assistance [38], 27 of these apps were not flagged as benign category by more than two Anti Virus (AV) items so these apps were dropped giving us 1500 benign apps.

3.2.2 Malware Dataset

This category comprises of a collection of 400 malware applications which are divided into two categories adware and general malware having 250 and 150 applications respectively. The first category adware comprises of following adware malwares families.

- Airpush: Shows unwanted ads on user device. The purpose is to steal information.
- Dowgin: It is an advertisement library which also seeks for stealing user information.
- Kemoge: Designed for enable a backdoor in Android devices. This adware use repackaging to show itself as popular apps. This adware can also be called a hybrid of botnet.
- Mobidash: Display ads for compromising user information.
- Shuanet: Similar to Kemoge, Shuanet also use back dooring to take over a user device. An Antivirus company Lookout found out that Kemoge and Shuanet have a lot of code similarity of 71 percent to 82 percent for building their versions of the auto-rooting [39].

The following households have been chosen for the apps:

- AVpass: Designed as a injection in the guise of a Clock app.
- FakeAV: Tricks the user to buy a full version of the software for removing infections which really doesn't exists.
- FakeFlash/FakePlayer:A Flash app which directs user to a site.

- GGtracker: For information stealing by using SMS also sending SMS to premier account.

- Penetho: A fake tool for infecting user device claiming to crack WIFI passwords. by using email attachment, fake updates, external mass media and infected documents the malware can infect user computer also [40].

Further, for analysing diversity between different classes of apps an analysis was done to find out relation between each malware categories (adware, general malware, and benign). The analysis shows that there is a weak relationship between these categories also if only malware categories is consider it also depicts a week relationship. Fig. 3.1 shows an overview of the analysis. This shows that dataset is diverse which is an important data point for balancing data for further analysis.

For more diversity the dataset was embedded with network captures of local network comprising of Android application traffic of different categories of application i.e. non malicious and malicious apps.

3.3 Feature Extraction

The Dataset provided by Canadian Institute of Cyber Security contains pcaps of Bengin, Adware and Malware traffic. For extracting the feature vector, we used CICFlowMeter [24] which is a a flow features extraction software designed by CIC. CICFlowMeter is an opensource tool developed in java and expandable for implementation of some new features. Overall, 76 extracted having 600k instances of traffic flows using the CICFlowMeter. These was categorized in Behavior based, Byte based, Packet based, and Time based. Moreover, features like source port,

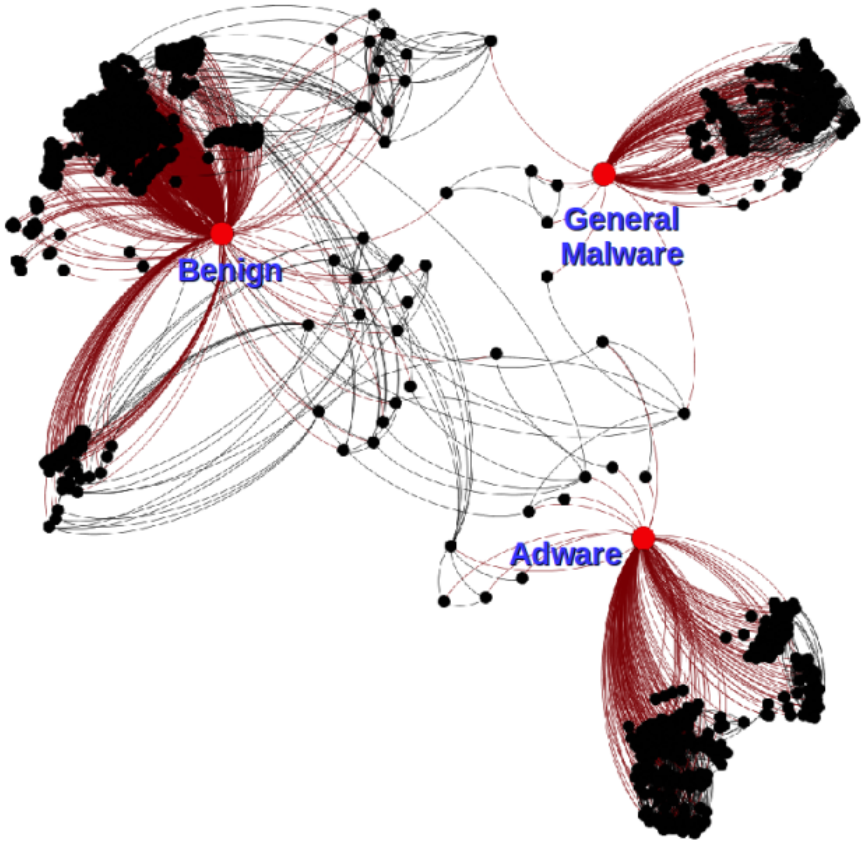

Figure 3.1: Droidkin [25] based apps similarity detection

destination port, Source IP, and Destination IP was removed from list of features. A list of all the features extracted is given in Fig. 3.2. A Bash script was written to accomplish this task. The script takes a patch of folder containing pcap's and fed it to CICFlowMeter to generate a csv files containing flows. CSV's files of different categories was then put together and labeled according. Weka was used to perform some data cleansing on these combined CSV's to remove instances having NULL values. The output dataset with all three categories contains more 600k instances each having 76 features. The feature extraction process is depicted in Fig. 3.3. Also, a detailed breakdown of all instances of dataset with respect to classes are given in Table. 3.1.

Behavior-based		
Feature	Feature name	Description
F_1	Duration	The duration of the flow

Byte-based		
F_2	Total Forward Bytes	Total bytes in the forward direction
F_3	Total Backward Bytes	Total bytes in the forward direction
F_4	Forward Header Length	The total bytes used for headers in the forward direction
F_5	Backward Header Length	The total bytes used for headers in the backward direction

Packet-based		
F_6	Total Forward Packets	Total packets in the forward direction
F_7	Total Backward Packets	Total packets in the backward direction
F_8	Forward packet length (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the size of packet in forward direction
F_9	Backward packet length (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the size of packet in backward direction

Time-based		
F_{10}	Forward Arrival Time (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation time between two packets sent in the forward direction
F_{11}	Backward Arrival Time (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation time between two packets sent in the backward direction
F_{12}	Idle Time (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation time a flow was idle before becoming active
F_{13}	Active Time (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation time a flow was active before becoming idle

Flow-based		
F_{14}	Flow Packet Length (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the length of a flow
F_{18}	Flow Forward Bytes	The average number of bytes in a sub flow in the forward direction
F_{19}	Flow Backward Bytes	The average number of bytes in a sub flow in the backward direction
Flow-based (Proposed)		
F_{15}	Backward Variance Data Byte	Variance of total bytes used in the backward direction
F_{16}	Forward Variance Data Byte	Variance of total bytes used in the forward direction
F_{17}	Flow FIN	Number of packets with FIN
F_{20}	Idle (Max, Min)	Min/Max time a flow was idle before becoming active
F_{21}	Initial Window Forward	The total number of bytes sent in initial window in the forward direction
F_{22}	Initial Window Backward	The total number of bytes sent in initial window in the backward direction
F_{23}	Segment Size Forward (Max, Min)	Max/Min segment size observed in the forward direction
F_{24}	Segment Size Backward (Max, Min)	Max/Min segment size observed in the backward direction

Figure 3.2: List of All Features.

3.4 Proposed framework

The complete framework is shown in Fig. 3.4 in the form of a block diagram. The proposed methodology consists of two parts Training and Predicting. The Training process consists of following steps.

1. Firstly, Labeled dataset gets loaded. The dataset is in a csv file having 600k instances and 76 features.

2. Data is then preprocessed to remove any unwanted values.

3. Then data is divided into Training set and Testing set. Validation set is a sub part of training set.

4. Xgboost model is trained on training dataset with default parameters. Also, parameter tuning is performed to get best parameters for training set for which cross validation of 10 folds is done.

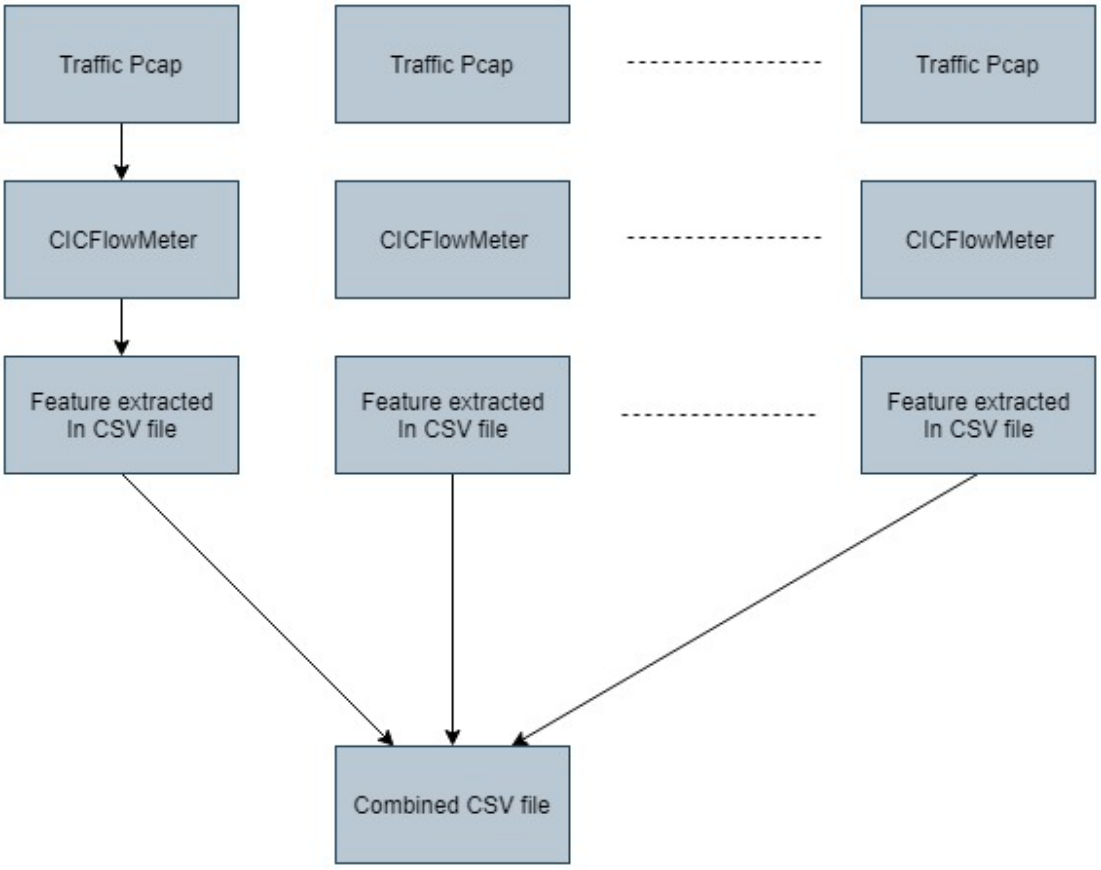

Figure 3.3: Dataset Generation Overview.

5. The generated model from above setup is then subjected to Test set. Accuracy on testing set is recorded in this setup.

6. Above model is saved for Predicting purposes.

The Predicting consists of following setups.

1. Network traffic is captured from a network interface. This data traffic is then subjected to a feature extraction program to get flows information.

2. Preprocessing is performed to remove any unwanted values.

3. This new data acts as an input to saved model for prediction.

4. The results are then analyzed and evaluated.

Table 3.1: Dataset Details

Data Type	CSV
Classes	3
Total Instances	631,955
Benign Instances	471,597
Adware Instances	155,613
General Malware Instances	4745
Total Features	76

Each on above steps will be discussed in detail in coming sections.

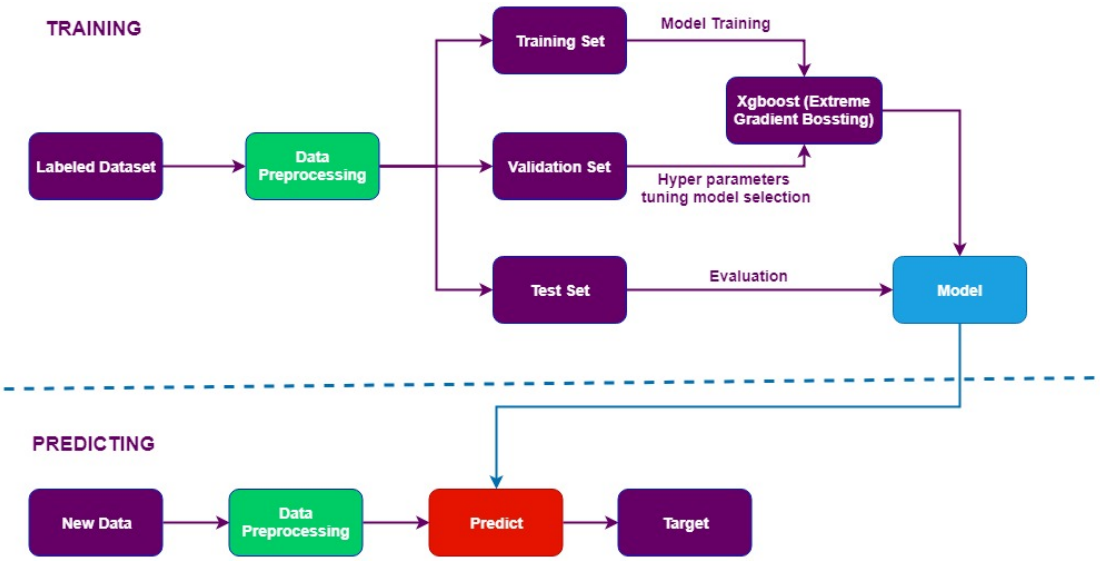

Figure 3.4: Purposed Framework.

3.4.1 Dataset Loading

Dataset loading is the first step of proposed framework of malware classification and detection. This step consists of following steps.

1. The dataset is loaded from csv file having more than 600k rows and 82 columns.

2. Source IP, Destination IP, Time stamp and some other features are dropped due to their low significance for classification and detection purposes. This narrow downs the feature set to 76 features.

3. A python library pandas was used for this purpose. Pandas loads the dataset in data frames.

3.4.2 Preprocessing

Pre-processing includes following steps.

1. Loaded dataset is scanned for any instances having "NULL" values.

2. These instances are than removed for having an unbiased detection model.

3. Python library Pandas have a feature to remove "NULL" values which is used for this purpose.

3.4.3 Dataset Splitting

Splitting of dataset involves below steps.

1. Preprocessed data is subjected to "train_test_split" function by scikit learn library python.

2. A training set and testing set in ratio of 80% and 20% is obtained respectively.

3. Validation set is a sub part of training set used for cross validation.

3.4.4 Model Training

Xgboost is the main algorithm in this research. Xgboost is known for its speed and performance it's an implementation of gradient boosted decision trees. It's has been used by many data scientist to provide state of the art results on many machine learning problems. XGBOOST is an ensemble of decision tree. Tree ensemble model is a combination of different classification and regression trees known as CART. Fig. 3.5 provides a simple example of CART classification about liking of a computer game X.

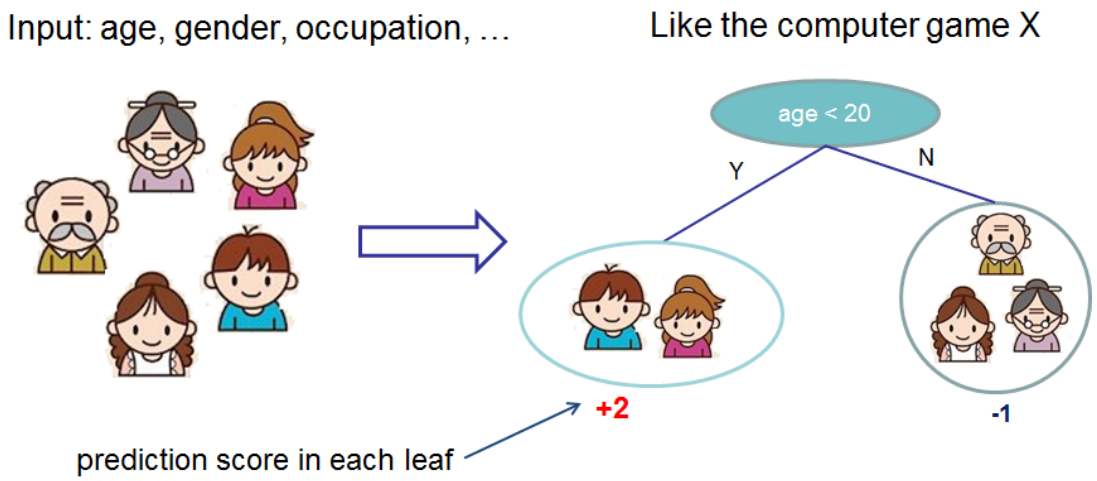

Figure 3.5: CART classification about liking of a computer game X.

Each member of family is classified into different leaves having a score assigned to them. A CART is different from standard decision trees where each leave have a decision value instead to this is CART real score is associated with every leave as shown in Fig. 3.6

Also one decision tree is not accurate enough for a proper classification. So we use ensemble of different trees and combine the results to get better accuracy.

Fig. 3.6 shows an example of the tree ensemble. The score predicted by each tree

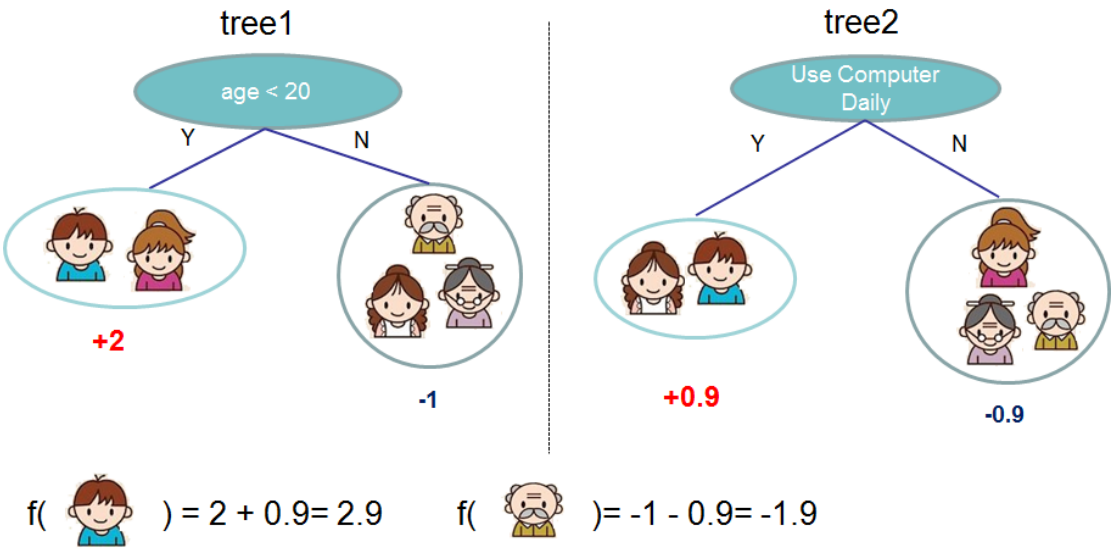

Figure 3.6: Example of the tree ensemble.

is summed up for a final score. Mathematical representation of model is given in Eq. 3.4.1

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{3.4.1}$$

Where K is representing number of trees, f is a function belonging to functional space F, where F is set of all CARTs. So, the objective function can be defined as Eq. 3.4.2

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.4.2}$$

Model training consists of below steps.

1. Training set is subject to Xgboost Model with default parameters.

2. Feature importance score shown in Fig. 3.7 is evaluated in this section.

3.4.5 Parameter Tuning

The most crucial step is parameter tuning XGBoost modelling is a easy task. But, improving it takes a considerable amount of work and resources. XGBOOST uses

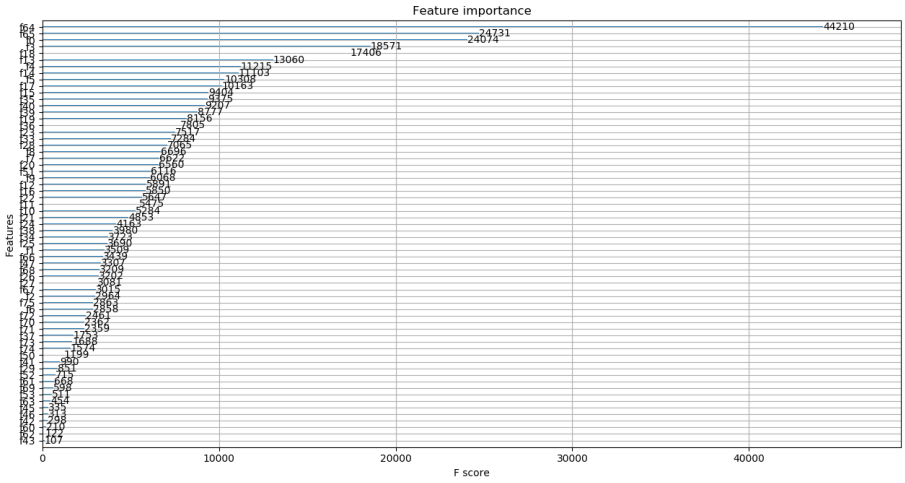

Figure 3.7: Feature Importance Score.

a variety of parameter. Parameter tuning is a key factor for improving the model. In parameter tuning most crucial questions are set of parameter to tune? and the optimal values for these parameters. The parameter choosen are given in Table. 3.2 parameter tuning involves below steps.

1. Firstly, select a range for a particular parameter.
2. Perform Grid Search by using Scikit learn library GridSearchCV method on above range. This method also perform cross validation to reduce over fitting.
3. GridSearchCV will provide best parameters based on evalution metric used. In this case the metric is accuracy.
4. Repeat above step to analyze any enhancement in accuracy.
5. lastly, repeat for all other parameters.

A list of tune-able parameters with suggested ranges in given in Fig. 3.8.

Table 3.2: Parmeters and there optimal value

Parameters	Optimal Value
learning_rate	0.05
max_depth	10
min_child_weight	1
Gamma	0
Subsample	0.85
colsample_bytree	0.9
scale_pos_weight	1

Parameter Name	Parameter Type	Recommended Ranges
alpha	ContinuousParameterRanges	MinValue: 0, MaxValue: 1000
colsample_bylevel	ContinuousParameterRanges	MinValue: 0.1, MaxValue: 1
colsample_bytree	ContinuousParameterRanges	MinValue: 0.5, MaxValue: 1
eta	ContinuousParameterRanges	MinValue: 0.1, MaxValue: 0.5
gamma	ContinuousParameterRanges	MinValue: 0, MaxValue: 5
lambda	ContinuousParameterRanges	MinValue: 0, MaxValue: 1000
max_delta_step	IntegerParameterRanges	[0, 10]
max_depth	IntegerParameterRanges	[0, 10]
min_child_weight	ContinuousParameterRanges	MinValue: 0, MaxValue: 120
num_round	IntegerParameterRanges	[1, 4000]
subsample	ContinuousParameterRanges	MinValue: 0.5, MaxValue: 1

Figure 3.8: Tunable XGBoost Hyperparameters.

3.4.6 Model Saving

The last step for purposed methodology of "Andriod Malware Classification and Detection" is saving the model generated after fine tuning of parameters. The purpose of saving the model is for evalution purposed which is discussed in next chapter. The model is saved in a dat file by using joblib module of scikit learn.

Chapter 4

Experimentation

This chapter provides the detailed discussion on the implementation and results of the proposed technique. This chapter is divided into the following parts.

- Experimental Setup
- Results

4.1 Experimental Setup

The experimental setup consist of the standard data set which is the 20% of the whole dataset. Also experimental setup includes testing on a copy of a live network traffic to test its performance on local network. For experiments we used R620 server having 32 cores and 62 GB of RAM. The details of test dataset is given in Table 4.1.

Table 4.1: Dataset Details

Data Type	CSV
Classes	3
Total Instances	126,391
Benign Instances	94,037
Adware Instances	31,391
General Malware Instances	963
Total Features	76

4.2 Results

Results section has been divided into two parts based on datasets. Firstly, results are based on test dataset and secondly, the model is subjected to passive analysis of live traffic. Each of them have been explained in the subsections below.

4.2.1 Test Dataset

The model is firstly tested on test dataset which is 20% of the complete dataset. The details of test data is given in Table 4.1. Four common metrics, Accuracy (Pr) or Positive Predictive, Recall (Rc) or Sensitivity Precision and False Positive (FP) rate have already been chosen to evaluate the standard of the classification procedure. The Pr is equal to the ratio of correctly classified instances (TP), let us say X, in front of all the instances classified as X (TP+FP). Whereas the Rc is usually add up to the ratio of properly classified instances (TP), why don't we say Y, before all Y situations (TP+FN).

Table 4.2: Detailed Accuracy by Class

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.968	0.167	0.944	0.968	0.956	0.821	0.974	0.988	benign
0.841	0.032	0.896	0.841	0.868	0.827	0.976	0.941	adware
0.416	0.001	0.754	0.416	0.536	0.558	0.891	0.522	GeneralMalware

$$Pr = \frac{TP}{TP + FP} \tag{4.2.1}$$

$$Rc = \frac{TP}{TP + FN} \tag{4.2.2}$$

Where the TP holds true Positive, FP is False Positive, and FN is the False Negative. In the assessment step we utilized the weighted normal of precision, accuracy and recall to select the best mixture of data set and features.The weighted typical of accuracy is calculated as:

$$WPr = \frac{\sum Prc \cdot (TPc + FPc)}{TP + FP + TN + FN} \tag{4.2.3}$$

Where PrCi may be the precision of class Ci, TPCi + FPCi is the accurate number of samples categorized as Ci, and TP + FP + T N + F N may be the final number of samples. The weighted accuracy and recall is calculated following a same procedure. Further results are depicted in paper published in [41]. Detailed accuracy by class are show in Table 4.2.

Confusion matrix of test results is show in Table 4.3. The overall, accuracy achieved is approximately 94%.

Table 4.3: Confustion Matrix

a	b	c	
90989	2940	108	a = benign
4970	26398	23	b = adware
446	116	401	c = GeneralMalware

4.2.2 Passive Analysis of Live Traffic

In this analysis passive analysis of live traffic was performed to detect the ability of our model to real world traffic. An interface of our test setup R620 server, which was receiving tap traffic of a mobile device having malicious and normal apps. A copy of this traffic was subjected to CICFlowMeter for feature extraction these feature after performing some preprocessing was given as an input to our model. Our model performed the classification and the result was pooled by help of a Web API. The designed Web UI shows the result of the classification in form of counter and a pie chart. The overview of whole process is depicted in Fig. 4.1.

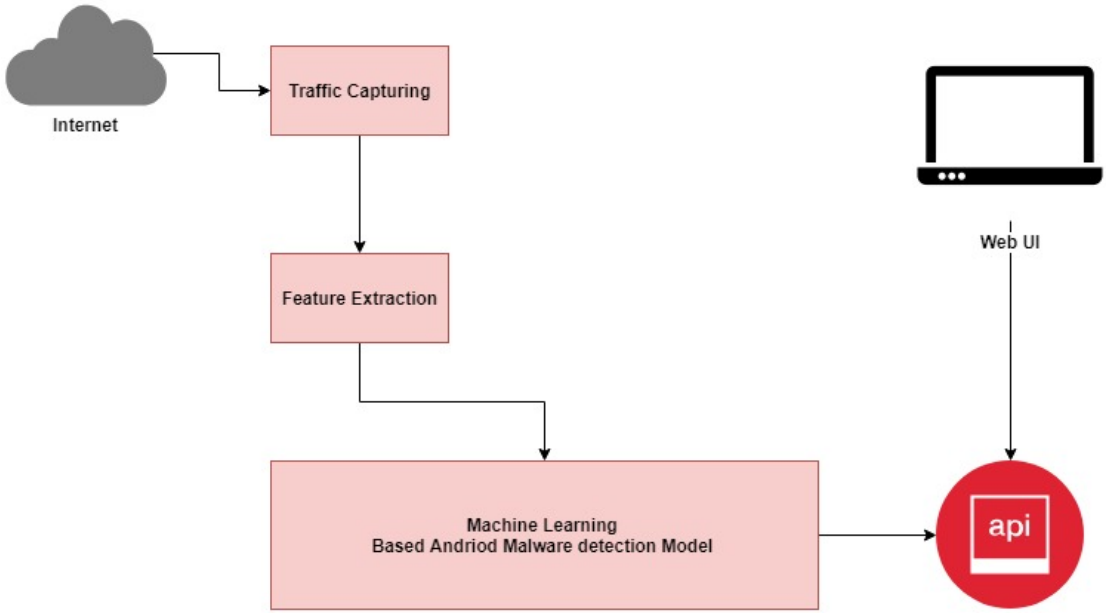

Figure 4.1: Purposed Framework.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this research, a machine learning based Android malware detection model is purposed this model works on 76 features for effective classification of Android malwares. Moreover, this framework uses TCP and UDP flows based features which are divided into behavior based, time based, packet based and flow based. The experimental analysis of purposed model depicts that purposed model have a high accuracy of 94% having false positive rate of 0.08%. Also passive testing of model on live traffic shows that, model is performing pretty well on live scenario also. The purposed model is also speed efficient and exploit parallelism which is good for handling high bandwidth of network traffic. The Algorithm used also 10x less time in training and testing as compared to other Machine learning Algorithm.

5.2 Future Work

Although this scholarly study achieved its objectives, a true number of recommendations for future studies have been identified. This section presents recommendations for future works predicated on the discussed restrictions. Attackers always make an effort to evade detection strategies by finding new methods to bypass such strategies. Although this ongoing function experimented on a multitude of real-world malware households, it is beneficial to collect even more samples of malware. This allows researchers to discover new attack and behaviours methods of malware families. The development of a sophisticated version of the Google android emulator would allow researchers to analyse even more malware samples in much less time, with an increase of realistic results. The Google android emulators absence some features when compared to real gadget, such as for example IMEI, routing desk, timing attacks, sensory result, and serial number. Research upon this presssing concern would benefit the study community later on.

References

[1] Gartner. Gartner top 10 strategic technology trends for 2019, 2019. URL https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019/.

[2] Aon. Cyber security risk report, 2019. URL https://www.aon.com/getmedia/4c27b255-c1d0-412f-b861-34c5cc14e604/Aon_2019-Cyber-Security-Risk-Report.aspx.

[3] Macfee. Mobile threat report q1 2018, 2018. URL https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2018.pdf.

[4] FIREEYE. M-trends, 2019. URL https://content.fireeye.com/m-trends.

[5] FIREEYE. Annual threat reports, 2019. URL https://www.fireeye.com/current-threats/annual-threat-report.html.

[6] Arash Habibi Lashkari, Andi Fitriah A.Kadir, Hugo Gonzalez, Kenneth Fon Mbah, and Ali A. Ghorbani. Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual Conference on*

Privacy, Security and Trust (PST). IEEE, aug 2017. doi: 10.1109/pst.2017.00035.

[7] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Mamun, and Ali Ghorbani. Characterization of encrypted and vpn traffic using time-related features. 02 2016. doi: 10.5220/0005740704070414.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278.

[9] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. ACM Press, 2016. doi: 10.1145/2939672.2939785.

[10] Danny Iland, Alexander Pucher, and Timm Schäuble. Detecting android malware on network level. 2011.

[11] Marian Kuhnel. *Detection of Traffic Initiated by Mobile Malware Targeting Android Devices in 3GPP Networks*. Dissertation, RWTH Aachen University, Aachen, 2017. URL http://publications.rwth-aachen.de/record/697470. VerÃűffentlicht auf dem Publikationsserver der RWTH Aachen University; Dissertation, RWTH Aachen University, 2017.

[12] L. Tenenboim-Chekina, O. Barad, A. Shabtai, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici. Detecting application update attack on mobile devices through network features. In *2013 IEEE Conference on Computer Com-*

munications Workshops (INFOCOM WKSHPS), pages 91–92, April 2013. doi: 10.1109/INFCOMW.2013.6970755.

[13] Lena Chekina, Dudu Mimran, Lior Rokach, Yuval Elovici, and Bracha Shapira. Detection of deviations in mobile applications network behavior. *CoRR*, abs/1208.0564, 2012. URL http://arxiv.org/abs/1208.0564.

[14] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song. Networkprofiler: Towards automatic fingerprinting of android apps. In *2013 Proceedings IEEE INFOCOM*, pages 809–817, April 2013. doi: 10.1109/INFCOM.2013.6566868.

[15] A. Arora, S. Garg, and S. K. Peddoju. Malware detection using network traffic analysis in android based mobile devices. In *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, pages 66–71, Sep. 2014. doi: 10.1109/NGMAST.2014.57.

[16] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici. Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security*, 43:1 – 18, 2014. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2014.02.009. URL http://www.sciencedirect.com/science/article/pii/S0167404814000285.

[17] J. Li, L. Zhai, X. Zhang, and D. Quan. Research of android malware detection based on network traffic monitoring. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 1739–1744, June 2014. doi: 10.1109/ICIEA.2014.6931449.

[18] Abimael Carrasquillo, Albert E Maldonado, Eric Santos, and Poster Ortiz-Ubarri, Jose. Poster: Towards a framework for Âťnetwork-based malware detection system. In *35th IEEE Symposium on Security and Privacy*, 2014.

[19] and Z. Chen, L. Zhang, Q. Yan, B. Yang, and L. Peng and. Trafficav: An effective and explainable detection of mobile malware behavior using network traffic. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–6, June 2016. doi: 10.1109/IWQoS.2016.7590446.

[20] Anshul Arora and Sateesh K. Peddoju. Minimizing network traffic features for android mobile malware detection. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*, ICDCN '17, pages 32:1–32:10, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4839-3. doi: 10.1145/3007748.3007763. URL http://doi.acm.org/10.1145/3007748.3007763.

[21] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: Multilevel traffic classification in the dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 229–240, New York, NY, USA, 2005. ACM. ISBN 1-59593-009-4. doi: 10.1145/1080091.1080119. URL http://doi.acm.org/10.1145/1080091.1080119.

[22] T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials*, 10(4):56–76, Fourth 2008. ISSN 1553-877X. doi: 10.1109/SURV.2008.080406.

[23] Karel Bartos, Michal Sofka, and Vojtech Franc. Optimized invariant representation of network traffic for detecting unseen malware variants. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 807–822, Austin, TX, 2016. USENIX Association. ISBN 978-1-931971-

32-4. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/bartos.

[24] Arash Habibi Lashkari., Gerard Draper Gil., Mohammad Saiful Islam Mamun., and Ali A. Ghorbani. Characterization of tor traffic using time based features. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 253–262. INSTICC, SciTePress, 2017. ISBN 978-989-758-209-7. doi: 10.5220/0006105602530262.

[25] Iman Sharafaldin., Arash Habibi Lashkari., and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 108–116. INSTICC, SciTePress, 2018. ISBN 978-989-758-282-0. doi: 10.5220/0006639801080116.

[26] H. Azwar, M. Murtaz, M. Siddique, and S. Rehman. Intrusion detection in secure network for cybersecurity systems using machine learning and data mining. In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–9, Nov 2018. doi: 10.1109/ICETAS.2018.8629197.

[27] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *2012 IEEE Symposium on Security and Privacy*, pages 95–109, May 2012. doi: 10.1109/SP.2012.16.

[28] Daniel Arp, Michael Spreitzenbarth, Malte HÃijbner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket. 02 2014. doi: 10.14722/ndss.2014.23247.

[29] HCLR. Hacking and countermeasure research lab, 2018. URL http://ocslab.hksecurity.net2018.http://ocslab.hksecurity.net.

[30] Hyunjae Kang, Jae wook Jang, Aziz Mohaisen, and Huy Kang Kim. Detecting and classifying android malware using static analysis along with creator information. *International Journal of Distributed Sensor Networks*, 11(6): 479174, jan 2015. doi: 10.1155/2015/479174.

[31] Jae wook Jang and Huy Kang Kim. Function-oriented mobile malware analysis as first aid. *Mobile Information Systems*, 2016:1–11, 2016. doi: 10.1155/2016/6707524.

[32] Jae wook Jang, Hyunjae Kang, Jiyoung Woo, Aziz Mohaisen, and Huy Kang Kim. Andro-dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information. *Computers & Security*, 58: 125–138, may 2016. doi: 10.1016/j.cose.2015.12.005.

[33] Jae-wook Jang, Jaesung Yun, Aziz Mohaisen, Jiyoung Woo, and Huy Kang Kim. Detecting and classifying method based on similarity matching of android malware behavior with profile. *SpringerPlus*, 5(1):273, Mar 2016. ISSN 2193-1801. doi: 10.1186/s40064-016-1861-x. URL https://doi.org/10.1186/s40064-016-1861-x.

[34] Nicolas Kiss, Jean-Francois Lalande, Mourad Leslous, and Valérie Viet Triem Tong. Kharon dataset: Android malware under a microscope. In *The LASER Workshop: Learning from Authoritative Security Experiment Results (LASER 2016)*, pages 1–12, San Jose, CA, 2016. USENIX Association. ISBN 978-1-931971-35-5. URL https://www.usenix.org/conference/laser2016/program/presentation/kiss.

[35] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. Deep ground truth analysis of current android malware. In Michalis Polychronakis and Michael Meier, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 252–276, Cham, 2017. Springer International Publishing. ISBN 978-3-319-60876-1.

[36] Lei Xue, Yajin Zhou, Ting Chen, Xiapu Luo, and Guofei Gu. Malton: Towards on-device non-invasive mobile malware analysis for art. 08 2017.

[37] Symantec. 2017 internet security threat report, November 2017. URL https://www.symantec.com/security-center/threat-report.

[38] Virus total, December 2016. URL https://www.virustotal.com/en/.

[39] Lookout discovers new trojanized adware, December 2016. URL https://blog.lookout.com/blog/2015/11/04/trojanized-adware/.

[40] How to remove android penetho, December 2016. URL http://www.solvusoft.com/en/malware/viruses/Android-penetho.

[41] M. Murtaz, H. Azwar, S. B. Ali, and S. Rehman. A framework for android malware detection and classification. In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–5, Nov 2018. doi: 10.1109/ICETAS.2018.8629270.