Mobile malware attacks: Review, taxonomy & future directions

Attia Qamar, Ahmad Karim, Victor Chang

PII: S0167-739X(18)33160-1
DOI: https://doi.org/10.1016/j.future.2019.03.007
Reference: FUTURE 4831

To appear in: *Future Generation Computer Systems*

Received date : 19 December 2018
Revised date : 24 February 2019
Accepted date : 5 March 2019

Please cite this article as: A. Qamar, A. Karim and V. Chang, Mobile malware attacks: Review, taxonomy & future directions, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.03.007

# MOBILE MALWARE ATTACKS: REVIEW, TAXONOMY & FUTURE DIRECTIONS

[1]Attia Qamar, [2]Ahmad Karim, [3]Victor Chang
[1]National College of Business Administration & Economics, Pakistan
[2]Department of Information Technology, Bahauddin Zakariya University, Pakistan
[3]International Business School Suzhou, Xi'an Jiaotong-Liverpool University, China

## Abstract

A pervasive increase in the adoption rate of smartphones with Android OS is noted in recent years. Android's popular and attractive environment not only captured the attention of users but also increased security concerns. As a result, Android malware detection is one of the sizzling topics in the mobile security domain. This paper provides a comprehensive review of state-of-the-art mobile malware attacks, vulnerabilities, detection techniques and security solutions over the period of 2013-2019 that majorly targeted Android platform. We have presented various well organized and in-depth taxonomies that uncover mobile malware detection approaches based on their analysis techniques, working platform, data acquisition, operational impact, obtained results and artificial intelligence component involved. Another taxonomy comprises of mobile malware attack vector is presented to look threat clusters and loopholes to locate their malicious widespread impact on communities. Furthermore, we have discussed and classified forensic analysis efforts in mobile malware detection perspective. From intruder point of view, we have compared various evasion techniques that are used prominently by the malware authors to hinder detection efforts. Finally, future work directions are presented as guidelines for academia and industry alike to help them reduce or even avoid the harmful impact of these annoying efforts.

**Keywords:** *Mobile Malware, Malware, Mobile malware detection, Android Malware, Machine Learning, Forensics*

## 1. Introduction

Smartphones are the recent advent of mobile platforms having diverse kind of operating systems in which android, iOS, Blackberry, Symbian and windows are included. Advancement in technology opened the new horizons for application developers that results in providing wide variety of services and information at their fingertips, anytime, at any place. Popularity of android OS is not only attracting the wide range of users but it also became main target for malware writers.

Statista [1] reported that, number of smart phone users are reached 4.77 million from year 2013 to 2017 and forecasted that it can be raised over 5 billion in 2019. Recently, CEO of google reveals that, google android operating system reached major breakthrough with more than 2 billion monthly active android users[2]. Popularity of android smartphones is not hidden from malware authors to penetrate the security of mobile devices through malicious applications. G-DATA security blog[3], reported that 8,225 new malware sample targeted the android operating system and 744,065 malwares are counted during the Q4 2017. In addition, malicious applications used evasion techniques to hide themselves as a normal app and detection of these malware is really challenging task. Recently, 700,000 applications are founded that have malicious content and violate the google play store security policy [4].

It has been observed that built-in security structures of android smart phones are largely inadequate and other non-malicious programs can accidently render the intimate information. Many malware threats such as worms, backdoors, viruses, Trojans are vulnerable to smartphones, network devices and designed as open source projects. An infected device can perpetrate severe harmful activities such as lock the phone entirely or partially to make it unavailable, social engineering attacks causes to steal sensitive information, infect phonebook users and generate unwanted billing. Therefore, it is a challenge to develop robust detection system that mitigate malicious applications.

Detection approaches such as static, dynamic and hybrid are used by many researchers to analyze the android malware. From these researches, MADAM [5] follows hybrid analysis and achieved 96% accuracy to detect the malware in android devices. Another approach DroidGraph [6], adopt static method to analyze source code of applications which gives 87% android malware detection rate. Data protection and data privacy are identical terms in every sector. In this context, MoleDroid [7] detects the information theft from benign and malicious applications with 99.1 % precision by dynamically analyzes them. Similarly, Hussain et al. [8] proposed mhealth application for healthcare department to ensure the affordability and accessibility of health services all the time. Additionally, they handle sensitive data of patients with security and privacy.

In literature, authors discussed different aspects of mobile malware detection approaches in their review papers such as Feizollah et al.[9] focused on the general view of malware and detection approaches from the year 2010 to 2014. They discussed malware detection techniques with their four different types such as static, dynamic, hybrid and metadata analysis along with available android malware datasets. Although they did not presented any comprehensive comparison of detection approaches based on feature selection. In the work of [10], authors briefly describe android security, strengths, limitations and comparatively analyze the detection schemes based on taxonomy and survey of existing approaches. In addition, they explained the steps of android application development and provide baseline for entire community. Besides, they did not cover evasion techniques that becomes most interesting feature for malware writers, in that way malware easily evade from detection mechanism. Moreover, Yan et al.[11] concentrated on dynamic analysis to detect the mobile malware. They defined the emerging threats with detection techniques, classification approaches and their pros and cons widely. Likewise, they figure out evaluation criteria of existing methods, open issues and future directions. However, we have considered more elaborative and state-of-the-art dynamic, static and hybrid analysis methods to more strengthen the document.

The highlights of the contribution of this paper is as follows:

a. We presented mobile malware detection and mobile attack vector taxonomies, a state-of-the-art that covers different aspects such as, working environment, detection mechanism, data acquisition, malware detection architecture, types of cybersecurity attacks, operational impact, threat clusters and artificial intelligence techniques in aspects of malware detection and result evaluation.
b. A Comparative analysis is offered for existing detection approaches that are based on dissimilar features, datasets, analysis types, most targeted android malware and their performance level in the time period of 2013-2018.
c. Forensic analysis is provided with respect to mobile and malware forensics along with their operational tools, procedural steps from start to finish and challenges that are faced in forensic investigation method.
d. We elaborated some machine learning constraints, and which is overcome by deep learning proof-of-concept techniques in detection of mobile malware.
e. We also relatively define and compare evasion techniques such as polymorphism, java reflection, obfuscation and control flow alteration that is used by malware authors to evade from detection.

Besides, we explicitly determine some weaknesses from existing approaches that are not thoroughly investigated by researchers.

f. To cope with this growing hazard, challenges and future work directions are also manifested to provide a quick guideline for academia and industry alike.

Conclusively as a guideline, efficient tools that can detect malware from obfuscated code, analyze applications at runtime and identify zero-day attacks is really needed. Furthermore, code coverage problem needs to mitigate, and robust hybrid analysis methods are also required.

The remaining of this paper is structured as follows: section 2 describes the mobile malware evolution and its types, section 3, and section 4 consist of state-of-the-art taxonomies as mobile malware attack vector and mobile malware detection identically. In section 5, we have covered forensic analysis in context of mobile and malware forensics and section 6, presented a comparative analysis of various artificial intelligence approaches that includes machine learning and deep learning techniques in time period of 2013-2018 with their results. Furthermore, section 7 explain future directions that should be adopted by the researchers to avoid growing hazards. Finally, conclusion is provided in section 8 of the paper.

## 2. Mobile Malware Evolution and Types

Malware (malicious software) is any software with mischievous intention. It can be written to disrupt normal functioning, bypass access controls, gather sensitive information, display unwanted advertising, or getting control of device without user's knowledge. Moreover, malware and unintentionally harmful software are collectively termed as bad ware. Main categories in which malware can be grouped are virus, worms, Trojans, ransomware, rootkits, botnet, etc.

The evolution of Android malware has been observed to be at an accelerated pace. From the first ever virus, malware for mobile devices have evolved massively. F-Secure [12] reported that first virus named as Trojan for palm devices is introduced in year 2000. Cabir [13] made its appearance in June 2004, which was a worm settled as a proof of concept (POC) by Vallez, who was part of 29A group belongs to virus writers. The codes are written to contaminate Symbian devices, which propagate over Bluetooth as .sis package. Cabir was the very first malicious code that can exploit the networking technologies to spread itself and produced infection broadly. Correspondingly, FakePlayer [14] was the first android malware introduced in early August 2010. It showed the immediate motivation towards monetizing the mobile space. First Trojan application for android was exposed in August 2010 detected as ANDROIDOS_DROIDSMS.A.[15], it is a Russian SMS fraud application that sent SMS to premium rate numbers. The modern smartphones with evolving capabilities though offer more nefarious prospects to the enterprising criminals. Another Trojan was uncovered in a short time, masquerading as a game Tap Snake [16] . It has an ability to track GPS locations of septic devices over hypertext transfer protocol (HTTP) and then queried by other GPS spy applications. Ikee[17] was the first iOS based, a worm malware discovered in August 2010 that affects jailbroken devices and take benefit of secure shell (SSH) password to replicate other jailbroken iPhones. Furthermore, Trend micro [18] reported that, ZeuS malware escape from two way authentication factor and target the mobile banking sites. By each passing year, android operating system offers rich functionality and criminals reconnoiter multiple possibilities to hack devices. Similarly, criminals seek money making opportunities and motivated to put their efforts towards smartphones.

Another malware named as DroidDream [19], which is skillful to gain root access to smartphones devices. This malware family not only stole device details as International Mobile Station Equipment Identity (IMEI) and International Mobile Subscriber Identity (IMSI) but also causes to install more concealed malwares which draws off further information from device. On the release of Google's official security tool used to clean alterations made by malware writers, this is taken as an opportunity and release different tools that was cybercriminals used for information stealing and backdoor activities. Currently, more malicious android applications exploit users by forwarding messages, send premium rate SMS, spy on GPS location and detects telephone conversations using Google+ application. A recent report Kaspersky Lab [20], detected 1,319,148 malicious installation packages in Q2 of 2017. Moreover, it states that 200,054 mobile ransomwares are detected which is much more than as compare to Q4 2016. Likewise, Sophos [21] reported that Android ransomware grows with every passing year and number of malicious applications raised up to 3.5 million in 2017. Another newly invented android based malware RedDrop [22] is founded in fifty three android application packages (APKS) and causes to download alternative seven malicious applications automatically. Consequently, we summarized numerous categories of mobile malware with their families in table 1. Moreover, in figure 1, we presented timeline of mobile malware with top banking and general malware families from year 2000-2018.

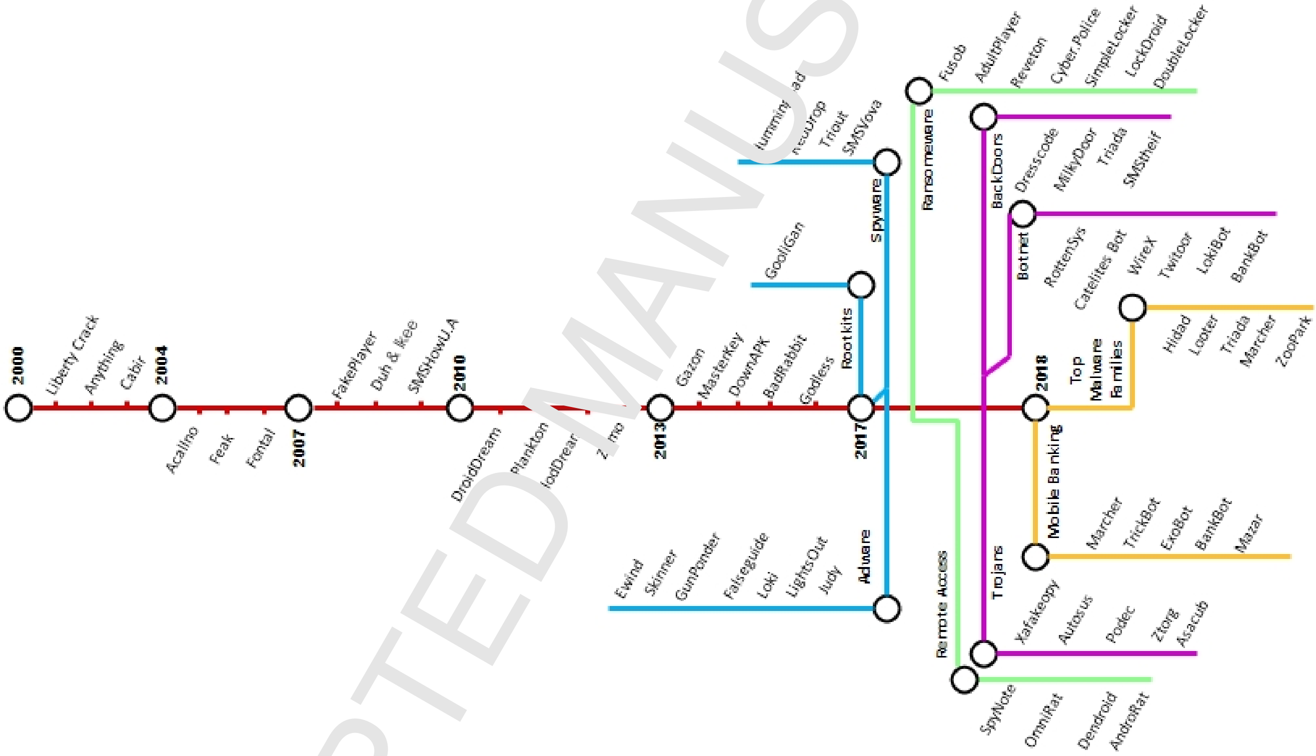


**Figure 1: Mobile Malware timeline from 2000-2018**

**Table 1: Mobile Malware Evolution**

| Year | Name | Targeted OS | Type | Description |
|---|---|---|---|---|
| 2000 | Liberty Crack[23] | Palm | Trojan | It pretends to be a hack and removes third-party apps. |
| 2004 | Cabir[23] | Symbian | Worm | It suppers from device to device by using object exchange (OBEX) push protocol that relates to Bluetooth. |
| 2005 | Cardblock[24] | Symbian | Virus | It installs as a fake software installation script (SIS) application that encrypts all data on the memory card with a random password. |

| | | | | |
|---|---|---|---|---|
| 2006 | Fontal[24] | Symbian | Trojan | It exploits vulnerability to over-write system files making the device stop functioning after reboot. |
| 2006 | Acallno[24] | Symbian | Spyware | It acts like a commercial software which collects information and steal money from user's device. |
| 2007 | Feak[24] | Symbian | Worm | A worm that sends SMS to all the contacts with a URL. |
| 2008 | Infojack[25] | Windows | Trojan | It comes with installation packages and disables the security settings of the device. |
| 2009 | Ikee and Duh [17] | iPhone | Worm | This malware used Cydia app distribution system to attack jailbroken iPhones, If default password is not changed after installing the security shell (SSH). |
| 2010 | FakePlayer[14] | Android | Trojan | First ever Trojan malware in Russia that makes money by sending SMS to premium numbers. |
| 2011 | DroidDream[19] | Android | Rootkit | It attacks on Google Play store by publishing more than 50 applications that contained a root exploit activities. |
| 2012 | Zitmo[19] | Windows | Botnet | A bot and banking malware Zeus improved to snip mobile transaction authentication numbers (mTANs). |
| 2013 | MasterKey[19] | Android | Trojan | This malware inject malicious code by putting itself as legitimate application. |
| 2014 | DownAPK[19] | Windows | Trojan | A malware causes to install fake banking applications by using Android debug bridge (ADB). |
| 2015 | Gazon[19] | Android | Virus | It spread himself through text messages and send a link to users to win gifts of $200. |
| 2016 | Godless[26] | Android | Root Exploit | It uses an open source frame work to gain root privilege and receive remote instructions that silently download and install an app that causes annoying apps and ads. |
| 2017 | Bad Rabbit[27] | Android | Ransom ware | It is distributed via drive by download and charge 0.05 bitcoin as a ransom form victim to release its resources. |
| 2017 | Judy[28] | Android | Adware | It is type of adware that activate by auto clicking and generate handsome amount from advertisement. |
| 2017 | GantSpy[29] | Android | Trojan | It acts as a useful application such as update Facebook, Android settings and they target images, contacts, call history and text messages. |
| 2018 | RedDrop[22] | Android | Spyware | It is a spyware found by the security researcher of Wandera in almost fifty three applications that can steal user information such as device information, record audio, images, files etc. |

The popular mobile malware types are the following that are versatile in nature such as:

**VIRUS:** A piece of code that has an ability to replicate itself and spreading to various applications on device is known as virus. Viruses often spread by attaching themselves to various programs followed by execution of code whenever a user initiate any septic program. Viruses propagate in system with the help of documents, script files, and from vulnerabilities in web applications. In some cases, viruses depend on human activities to launch themselves by opening any infected file or through running a program. Viruses causes different types of attacks such as snip information, steal money, damage host networks and computers, create command and control (C&C) activities , and others [30]. Duts, Lasco, Cardblock, CardTrap and Crossover [31] are examples of mobile viruses.

**WORM:** A worm is a piece of code, capable of replication and spread over computer networks from device to device without any human intervention [30]. Worms can contain "payloads" that damage host device and even it destroyed host networks by utilizing bandwidth and create congestion on web servers. Generally, payloads steal user's data, delete files from system and create botnets. Worms can be spread by opening an infected email attachment. Some known worms for smartphones are Cabir, CommWarrior, Feakk, Letum, Mobler, Beselo, Pmcryptic, Yxe, Ikee and ZeuS MitMo [31].

**TROJEN:** A type of malware that shows itself as a benign application to attract users to download and install malware is known as Trojan. In this type of malware, attackers gain remote access to steal data, money, delete and modify files, create variants of malwares, keep an eye on user activities as monitor screen and their logs etc. The Trojans that penetrated in mobile market are
MasterKey, DownAPK [19], GantSpy [29] etc.

**ROOTKIT:** A type of malware that gain remote access and control a device to exploit users known as rootkit. Rootkit consist of a dropper, loader and rootkit itself to do harmful action. It gains administrative access to install different malicious activities as steal information, disturb normal routine of system, apply changes in system, causes to alter system configuration etc. Once rootkit installed in computer it runs on every boot up. Due to secret operations of rootkit, it is difficult to identify and remove from system. As rootkit used obfuscation to hide their presence that's why it remains in system for a longer duration. Moreover, Check point [32] researchers found a rootkit HummingBad that installed deceptive application on mobile to stole credentials and generate fake ads.

**BOTNETs:** Bot is software program created to give an attacker, a remote access and control over the operations of infected device without users consent. Bots become part of botnets, which consist of number of computers to be controlled by botmaster. It is evolving to become a severe security threat because bonnets launch distributed denial of service (DDos) attacks, web spiders that hack the server data, malware masquerading on famous sites and spam bots that gathers information. DoubleDoor, DrakSky, jenX, Zyklon and Tofsee are the well-known examples of botnet malware[33].

**ADWARE:** It is an advertising-supported malware that is specifically designed to deliver advertisement to users spontaneously. Adware consist of advertisements and pop up ads that shows on websites. Generally, adware distributed as free of cost while in some cases advertisement companies sponsor them and generate revenue. Adware is only designed to deliver an ad, by clicking on ads, adware activates and steal information or track user activities. Moreover, RiskIQ [34] reported that, 14,758 android applications are flagged as adware in Q4 2107. Gunpoder, LightsOut, RottenSys, Judy and Skinner [35] are the examples of android Adware.

**SPYWARE:** A type of malware that monitor user activities without users consent. These activities consist of collecting key logs, screen watching and steal accounts information. Spyware create interference in network settings by changing its security operations. Spyware attach themselves with genuine software or in Trojans to exploit vulnerabilities. Acallno and FlaxiSpy are known as spywares in smartphones.

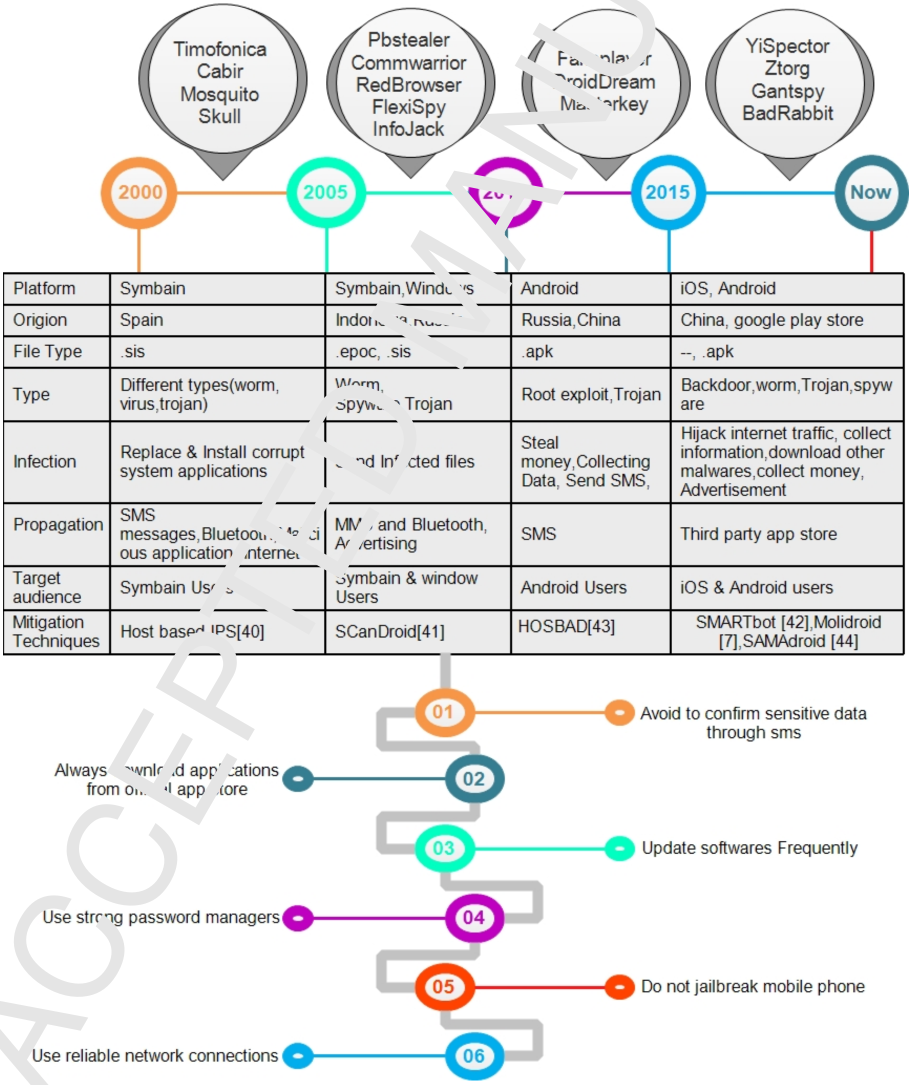
**RANSOMWARE:** A type of malware that did not release computer resources until victim pays some money as ransom. Ransomware causes to lock computer that leads to restriction on entry and encrypt files and display messages to force users to pay money. After payment ransomware malware will release the system. According to Symantec report [36], there is a 36% increase in the ransomware attacks and its hundred new malware variants are introduced in year 2017. Some common examples of android ransomware are Simplocker, Xbot and adult player [37].

**BACKDOORs:** Backdoors are type of malware that intended to set grounds for other malwares by opening a backdoor onto a device. It works as helper to other malicious activities by providing them a network connection to enter and snip information. Brador [38] a first ever malware that opens a backdoor in windows mobile.

**KEY-LOGGERS:** The malware that records everything as user type on system, in order to gather their log-in details and other subtle information and send it to the key-logging program. Key-loggers typically used by various organizations to obtain information related to computer usage. Flexispy [39] is a well-known spyware that keeps log of smartphone usage.

Consequently, in figure 2 we elaborated common malware types with their file type, origin, infection, propagation, etc. Moreover, we discussed mitigation techniques that are used in literature. Host based and network based intrusion detection system like (IPS) [40] methods are followed in earlier mobile malware mitigations. The purpose of this system is to block malicious content and unauthorized users.

Likewise, other mitigation techniques as Scandroid [41], SMARTbot [42], HOSBAD [43], and SAMAdroid [44] etc. are described in sub section 4.1.1 and sub section 6.1. Additionally, six easy steps are defined, how to stay away from mobile malware in figure 2.


Timofonica
Cabir
Mosquito
Skull
Pbstealer
Commwarrior
RedBrowser
FlexiSpy
InfoJack
Fakeplayer
DroidDream
Masterkey
YiSpector
Ztorg
Gantspy
BadRabbit
2000
2005
2010
2015
Now
01
Avoid to confirm sensitive data through sms
02
Always download applications from official app store
03
Update softwares Frequently
04
Use strong password managers
05
Do not jailbreak mobile phone
06
Use reliable network connections

| | | | | |
|---|---|---|---|---|
| Platform | Symbain | Symbain,Windows | Android | iOS, Android |
| Origion | Spain | Indonesia,Russia | Russia,China | China, google play store |
| File Type | .sis | .epoc, .sis | .apk | --, .apk |
| Type | Different types(worm, virus,trojan) | Worm, Spyware Trojan | Root exploit,Trojan | Backdoor,worm,Trojan,spyware |
| Infection | Replace & Install corrupt system applications | Send Infected files | Steal money,Collecting Data, Send SMS, | Hijack internet traffic, collect information,download other malwares,collect money, Advertisement |
| Propagation | SMS messages,Bluetooth,Malicious application,internet | MMS and Bluetooth, Advertising | SMS | Third party app store |
| Target audience | Symbain Users | Symbain & window Users | Android Users | iOS & Android users |
| Mitigation Techniques | Host based IPS[40] | SCanDroid[41] | HOSBAD[43] | SMARTbot [42],Molidroid [7],SAMAdroid [44] |

**Figure 2: Mobile Malware timeline with easy steps for protective measures**

## 3. Mobile Malware Attack Vector

This section elaborates the different characteristics of mobile malware attack vector that leads major loss to mobile users in terms of threat cluster, diverse kind of attacks, loop holes and operational impact as depicted in figure 3. Furthermore, a detailed description of figure 3 with state-of-the-art detection techniques are discussed in below sub sections.

### 3.1 Threat Clusters

Android employs security mechanism by providing the sandboxing feature. More ways are adopted by malware writers to manipulate the vulnerabilities in Android OS and network. At the time of application installation, it provides the user with a list of all permissions for granting access to resources and other apps data if there is a need for inter-app communication. The user can judge the malicious intent of an app if it requires such permissions that they are not needed. However, a naïve user often tends to ignore this safety mechanism. The applications may use permissions to grant malicious intentions.

Android is an open-source operating system, the vulnerabilities are exploited by the malwares to gain access to the device and fulfill their destructive intentions. Spywares are malwares that are intended towards gaining access to user's private data including their login information and passwords. Malware collect and send the data to remote devices without user's consent. Similarly, malwares intended to disrupt the actual functioning of the device, consume device resources including processor, storage and network resulting in high power usage of the device. The network may also be compromised by the malwares that causes too much network traffic either by sending data to servers or distributed denial of service (DDoS) attacks. A compromised smartphone may even causes to send premium charged SMS without user's consent. Therefore, we have classified threats into five categories that is posed by malwares in smartphones as shown in figure 3.

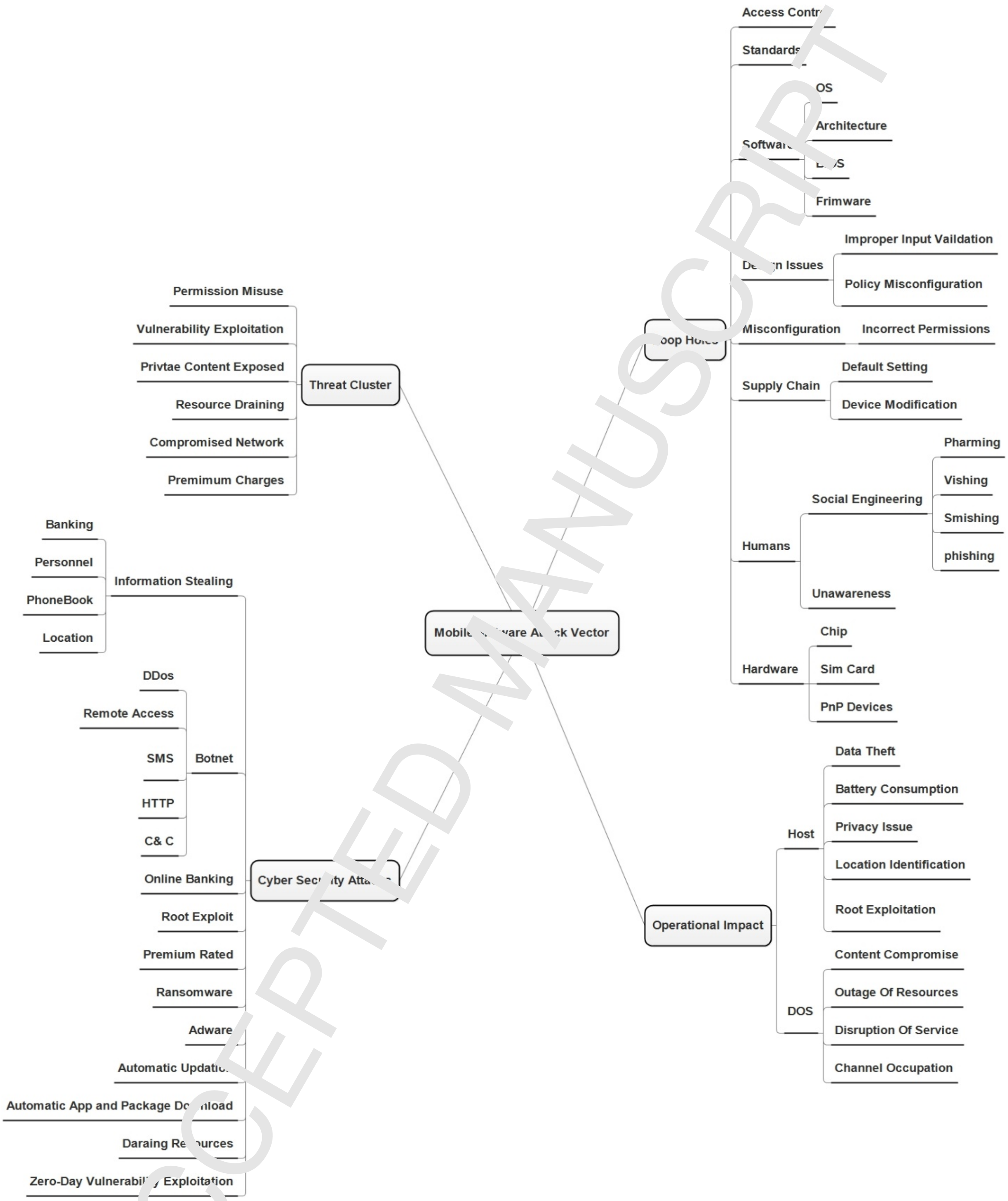


**Figure 3: Mobile Malware Attack Vector**

Permissions are basically acted as a firewall between the user and mobile applications that are used to protect the system in security point of view. In android OS all the permissions contain in AndroidManifest.xml file. Permissions are usually classified into three classes(a) normal, (b) dangerous, and(c) signatures[45]. Most of dangerous permissions are WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE, SEND_SMS, WRITE_CONTACTS etc. that causes an invasion for users.

Similarly, in the work of aung et.al [46] performed static analysis to detect the malware from permission based features. Another author [47] find out hidden permissions that sends short messages without user knowledge. Currently, smartphone vulnerabilities are increased as well as its adoption rate. The vulnerabilities in android phone occurs due to system flaws, connecting to unsafe internet connection and lack of user awareness[48]. Similarly, diverse kinds of weaknesses that leads to storage access, call, SMS, database, intents and in web view. Further, authors [49] detect the vulnerabilities in database of Cam Scanner and Clean master applications.

The private information comprises three different types of information i.e. social networking, sensitive and identification. The social networking information consists of e-mails, short messages, phone book etc. Furthermore, sensitive information includes credit card number, passwords, browsing logs, pictures and identification information having contacts, location, phone version, and IMEI etc.[7]. In the work of [50] authors performed experiment on smartphone numeric touchpad that is used by attackers to initiate keystroke inference attacks and they gain sensitive data though touch. If android smartphones are not protected it will be extremely vulnerable to be exploited. Recently, another researcher proposed a model Hybridroid [7] based on static and dynamic analysis that detect privacy leakage with the 97.8% precision in android applications. In addition DECIM [51], assure additional safety layer even attackers bypass the security of all keys in message transfer protocol.

The grabby android applications [52] that drain mobile phone resources in which battery, storage, CPU, memory and data are vibrant. According to Avast android application report [53] over 50% users did not discriminate among frank and fake applications. DevScope [54], is an energy efficient and dynamic power model for android smartphone that is based on BMU (Battery Monitoring Unit) to measure the power consumption. Moreover, security of android devices compromised and hacked by malicious Wi-Fi networks. As a result, Google introduced the security layer to analyze the applications before it is placed on the Google play store named as Bouncer [55]. In this context, a lightweight method DeDroid [56] is proposed that have an ability to detect the C&C related botnet mobile malwares.

Besides this, a famous method in which most of malicious applications send premium rate SMS messages and phone calls without user knowledge that causes the financial loss to the user. In that way, botmaster generate the revenue from it. Malwares tries to automatically subscribe users at the premium rate facilities[57]. Furthermore, another malware detection approach MADAM [5] is used to fruitfully identify the android malware families that financially charge the users.

### 3.2 Cybersecurity Attacks

With the passage of time, android malware is continuously growing and implement diverse methods to threaten the users. Correspondingly, proliferation of cyber security attacks in smart devices is the major challenge for organizations. Forensic experts are widely deal with cyber security threats. The great destructive threats of cybersecurity are ransomware, DDos, internet of things (IoT) devices and phishing etc. In below paragraphs, we deliberated various kinds of cybersecurity attacks along with their detection techniques.

Android applications furtively steal users information as new malware RedDrop [58] found in fifty three applications that causes to snip contacts, pictures, Wi-Fi information, device and SIM data and record audio. In this context, a HybriDroid model [59], is designed to detect data trickle in Android applications. As it covers dataset of multifarious android apps that causes data leakage during inter and intra app communication by analyzing applications in hybrid manner. Moreover, an app is developed named as MoleDroid [7], to detect malicious behavior from network flow that causes to steal users information. Consequently, it shows 99% true positive rate for detecting the information theft in network flows.

Another risky attack is botnet, the word "bot" and "net" derived from robot and network separately. In botnet attack, the invader breaches the security of all connected nodes of the network. A botnet attack is controlled by botmaster. The connection is established between botmaster and compromised network through heterogeneous protocols that include Peer to peer (P2P) and Hypertext transfer protocol (HTTP). Furthermore, botnet launch attacks such as Distributed Denial of service (DDOS), Remote Access and Short message service (SMS) via Command and Control(C&C)[60]. Whang et al.[61] presented a Venn-abers predictor for botnet detection in network traffic flow using K- Nearest Neighbor (KNN) and kernel Density Estimation (KDE) as an underlying scoring classifiers. Similarly, it covers the network traffic flow features as HTTP, IRC, and P2P that relates to botnet families. Another approach defined by da Costa et al.[62] in which they dynamically analyze the thirteen botnet families that having feature of API calls. Consequently, it provide 86% precision and 88% recall in 500ms time window.

Mobile banking relates to the online banking in which transaction done through android applications. Recently, Avast team cooperated with ESET and they examined the new version of malware known as BankBot [63]. They reported that, Bankbot consist of multiple malware variants that steal login credentials, money, slink into google play store and target the large banking application such as Diba, CitiBank and Chase [63]. Bojjagani et al. introduced a threat model VAPTAi [64] , that identify and detect the vulnerabilities at communication level of mobile banking applications. In an addition, they analyze the risky behavior of applications that is based on android and iOS. Correspondingly, another application that analyze the risky behavior of android applications at different levels with low, medium and high is named as MAETROID [65]. In this experiment, analysis is performed on 11.016 android applications in which both benign and malicious applications are included, collected from google play store and Genome identically. Consequently, the risk analysis app categorized them into good, bad and infected app based on their user ratings, number of downloads etc. Likewise, researchers [66] performed experiments on selection of mobile apps based on privacy priming which included self-relevant priming and other factual priming questions. As a result, users select an application on basis of user ratings. Furthermore, attackers gain access into the opcode that causes root exploit and steal private data [67]. DroidExec [68], a novel approach based on bipartite graph is designed that effectively detect root exploit malware families.

Ransomware is a form of malware in which attacker demand money from users to release their hacked system resources. It performs malicious actions such as steal users' information and lock the device to collect ransom from victim. A survey [69] states that around 4000 ransomware attacks launched every day and at the end of 2019 it will initiate after every 14 secs. In addition, to detect ransomware, maiorca et al. [70] proposed R-PackDriod that is based on a machine learning technique and efficiently marked the malicious application. Another approach [71] is applied to detect ransomware that uses HelDroid and google play store datasets to perform analysis on it. Finally, they achieved 100% detection rate that successfully discriminate between ransomware malware and trusted applications.

Malware such as shedun [72] is from adware family that automatically download and tries to install other malicious applications without users consent. It displays malicious ads to attract Android users with interesting features. In addition, RoughTed [73], a malware that bypasses the security of Ad-blocker. Similarly, an android app KS Clean[74] revealed fake system update pop-ups on screen. By clicking on them leads to exploit device vulnerabilities. In this context , Sanders et al. [75] identified six at risk permissions under the google automatically update policy that causes privilege escalation.

### 3.3 Loop Holes

In this section we will discuss loop holes as Android system grounds different vulnerabilities because users do not recognize the difference between original and fake applications. They compromised the security threats that causes different types of attack. Users blindly trust third parties and install fake application without understanding the requested permission from an application. In that way, attacker's record user's personnel information and misuse them. However, basic social engineering attacks that included: phishing, vishing, smishing and pharming are the common types of fraud that uses to steal sensitive data of users and gain profit. These terms are inter-related but there is a difference between their attack method and usage of medium. In phishing, an e-mail is sent to the targeted user from any authorized company or from bank and notify them to update their sensitive information such as passwords, credit card numbers, username, pin code etc.  As a result, couple of solutions are available against phishing attack such as install email virus detection software, educate employees of organization, assign least privileges etc. [76].

In the work of [77], they described connection between these social engineering terms as vishing is a kind of phishing done through voice or phone calls. On the other hand, smishing is a phishing that initiate attack by short message service (SMS). Similarly, authors [77] unveil common practices involved in phishing as drive by download, man in the middle, javascript obfuscation etc. Furthermore, in pharming a malicious code is installed on the victim's device and redirect clicks to another fraudulent website without users assent. In order to mitigation these attacks, two factor authentication (2FA) method is frequently applied by couple of organizations in which passcode is send on customers number to ensure its real identity [78]. Nevertheless, 2FA method is not sufficient as malware writers bypass this security mechanism reported by Wandera [79].

Subscriber Information module (SIM) is a tiny card that holds memory and facilitate users in communication. This module works on the Data Encryption Standard (DES) algorithms that is made by US government several years ago. In addition, DES is an old algorithm that did not meet with new security challenges that's why new robust kind of algorithms is required, that cope with recent security challenges. Attackers gain unauthorized access to the system that causes illegal activities. In access control, they gain extra advantage and find weaknesses from the system to perform malicious actions. As multiple malware attack vectors they try to gain access due to having some existing design flaws in the system such as vulnerabilities, open and unsecured system ports, unsafe network communication, unsatisfactory certifications etc. Recently, US Federal Bureau of Investigation (FBI) [80] announced legal notification for manufacturers if devices were found inadequate security measures. Furthermore, Unified Extensible Firmware Interface (UEFI) [81] provides a boot loader that is capable of unveiling different type of OS. It is not only for windows, as well as for mobile devices. By the rising rate of mobile users, it needs powerful security mechanism to fulfill emergent security challenges. Vendors faced major encounters in execution when they did not properly implement the secured boot loaders.

### 3.4 Operational Impact

Mobile malware attacks leave their impact on host and in form of Denial of Service (Dos). On the host it causes data theft, privacy issues, location identification, and root exploitation. Once attacker gain access in roots it can control all the system and generate any kind of attack. Moreover, malwares exploit the vulnerabilities in old security systems. In this case, easiest solution is to update the security patches of software's. Similarly, mobile malware drains the battery and causes CPU consumption. Different applications such as snapchat, tinder, Microsoft outlook are killing your battery life and keep running in background [52]. In denial of service, attacker makes all the available resources inaccessible for users, it may show the limited resources and compromised content. Furthermore, distributed denial of service (DDos) attack launch through botnets by sending heavy traffic on multiple nodes that are connected in a same network. Attackers gain backdoor access and compromise many IoT devices with DDos attack. In the work of, Ashraf et al. [82] deals with DDos attack to mitigate them by using machine learning methods comparatively.

## 4. Taxonomy of Mobile Malware Detection

This section comprehensively diagnosed different aspects of mobile malware detection methods as presented in figure 4. We have classified this domain knowledge in terms of (a) working environment (b) data generation and (d) analysis mechanism in detection process (i.e. data acquisition). Furthermore, malware detection approaches relies on analysis types such as (dynamic, static, and hybrid) and evasion techniques frequently adopted by malware writers. Each head gives the lengthy description about android malware detection schemes i.e. different analysis methods used for malware identifying purpose.

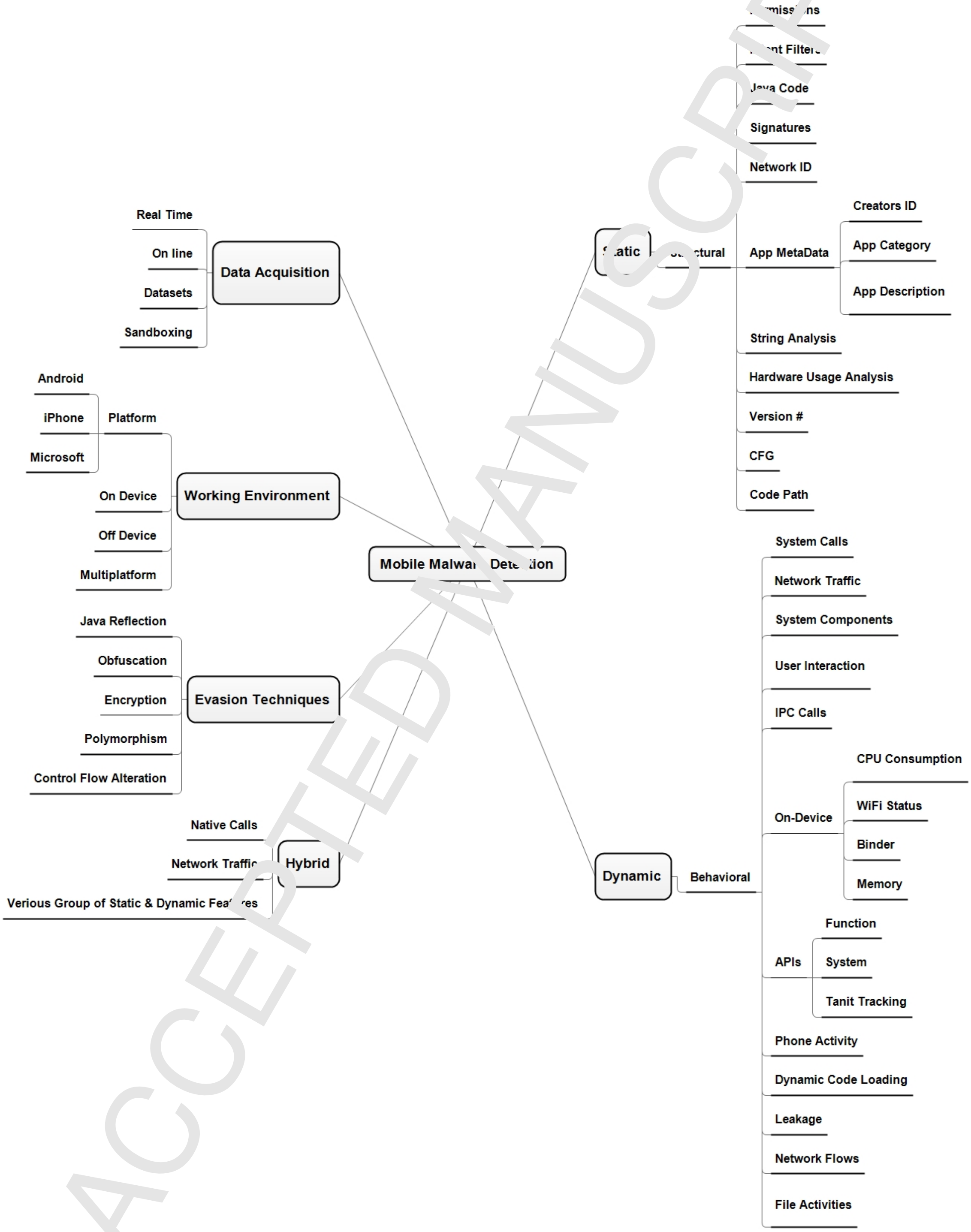


**Figure 4: Mobile Malware Detection**

### 4.1 Mobile Malware detection Approaches

This section illustrated the mobile malware detection approaches which are divided into three main categories (a) structural or static analysis (b) behavioral or dynamic analysis and (c) hybrid analysis to examine code and behavior of applications at runtime. These approaches are available for detecting malicious intentions and other security threats. Furthermore, we analyze existing studies based on malware detection techniques that covers main idea, proposed detection method, strengthens and their weaknesses.

### 4.1.1 Structural or Static Analysis

Static analysis (structural analysis), is a tactic of performing applications inspection by examining the program code without execution. The process provides an understanding of code structure and can help to understand the functionality it will perform. Code coverage is maximized in this approach as it involves the analysis of source code only. Another advantage of static analysis is that it reveals malicious intentions without paying the price of being noticed in actual execution and facing losses. However, this approach is highly ineffective in the presence of code obfuscation and dynamic code loading [83]. In static investigation method many applications suffer from challenges such as event-driven android application. The issues to be kept in mind for achieving effective results are [84] [85] :

(i) Multiple entry points to each app
(ii) The chances of multiple simultaneous, asynchronous app components execution
(iii) Event of frequent callbacks due to the application development stages
(iv) Involvement of both intra-app and inter-app ICC
(v) Java reflection, obfuscation and native code are also there as a big challenge

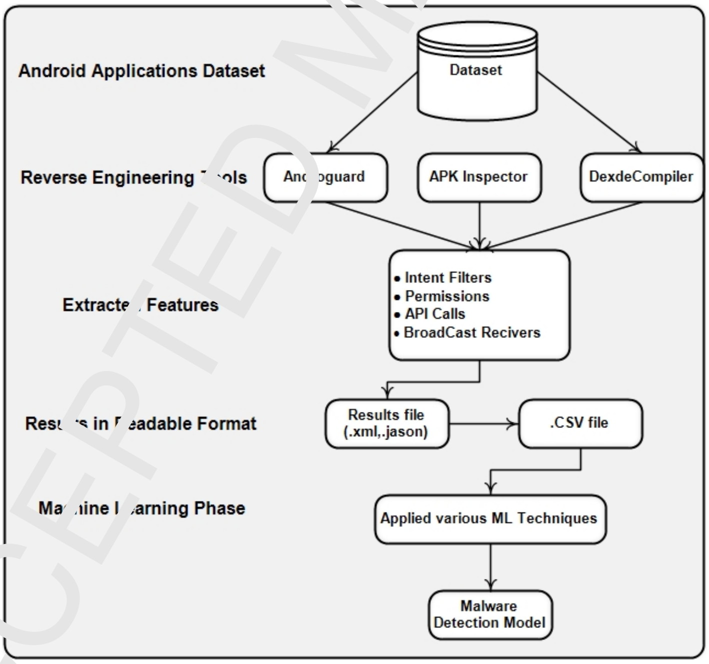


**Figure 5: common steps of Static Analysis with existing tools**

In figure 5, we presented well known steps as (a) datasets (b) reverse engineering tools (c) extracted features (d) result files (e) machine learning phase and finally malware detection model achieved. Likewise, we discussed famous static tools that are used in existing researches randomly. Grace et al. presents RiskRanker [86] with a pro-active scheme to spot zero-day malwares. It performs two-order risk analysis. In very first step, its purpose to determine non-obfuscated implementations that activate (a) known root, (b) illicit cost of creation, and (c) attacks on privacy leakage. RiskRanker scans native binary files for root exploit signatures for identifying known root exploits. It then checks for privacy leakage detection by employing slicing method to verify any information that sends to any connected nodes that relates to reveal personal information readings. Moreover, it uses a set of heuristics techniques that helps to uncover evasion process such as encryption or dynamic code loading and java reflection which could not detected in very first step of proposed method. Experiments of this approach is based on 118,318 apps, which is collected from various third-party application markets. Consequently, RiskRanker successfully detected 718 malwares, which relates to twenty-nine malware families and 322 zero-day malware attacks are included. However, RiskRanker has some limitation that it uses heuristics at second stage rather than in first step, that's why malwares easily escape in first detection process.

Fuchs et al. presents SCanDroid [41] incorporating automatic reasoning by applying modular incremental model about the security of Android applications. ScanDroid analyze data flow process of android application that is based on formerly settled language security model. It compares security mechanism to the analyzed manifest files and then decide whether it is consistent with required specifications or not. However, the approach was tested on some test cases thus it lacks to perform experimentations on real-time test scenarios.

Yerima et al. [87] developed and analyzed machine learning methodologies based on bayesian classification to reveal zero-day malware in a proactive manner via static analysis. The authors analyze one thirty one permissions from the manifest file. Baksmali unfolds .dex files into manifold files .smali files each containing only one class evidence. The files extracted related properties subsequently used to build the Bayesian classification-based models. The experiment comprised of 2000 Android apps; 1000 benign apps, 1000 malwares (from 49 known families) from Android Malware Genome Project, out of which 1600 samples (800 each) were used for training, while 400 (200 each) for testing. In addition, 5-fold cross validation was performed to show that accuracy to be 0.9 for permission-based model, 0.92 for code property-based model and 0.93 for mixed attributes model showing the last classifier model as the most promising one. However, constructing a feature vector to include all features is highly resource consuming.

Arzt et al. introduced FlowDroid [84] that perform static analysis to detect malware from android applications. It performs proper handling of callbacks invoked by the Android framework whereas analysis on data flow, context, objects, and field-sensitive data results in reduction of false alarms. Similarly, it models Android applications lifespan states and handles taint propagation due to user interface objects and callbacks. Moreover, innovative on-demand algorithms help FlowDroid to preserve high time proficiency and accuracy that gives a full open-source implementation. Experimenting FlowDroid on more than 500 benign applications from Google Play store and about 1000 malware applications from the virusShare confirms superior precision and recall as 86% and 93% respectively. While applying an inter-procedural dataflow analysis, FlowDroid does not track ICC-based (inter-components communication) dataflow for applications. In the work of [88], they introduced Amandroid approach that significantly track the ICC but does not deal with reflections and concurrency.

In the work of Wu et al. [89] proposed a static feature-based system DroidMat, meant to extract illustrative configuration, API call traces and permissions. The study involved malicious samples from Contagio dataset [90]. It deploys clustering algorithm for intent messages, K-Means to improve malware modeling ability and k-NN algorithm to classify applications as benign or malicious. Consequently, the recall rate and predicted efficiency of DroidMat proved to be significant by achieving 97.89% accuracy.

Another study by Samra et al. [91] focused their work towards apps in business and tools categories. The permissions that governs access to resources are about one hundred thirty. Other than permissions and the features extracted from manifest file are application name, category, description, rating value, price, etc. are also observed.

Schmidt et al. [92] performed static analysis particularly focusing on worms, and presented on-device solution to malware detection that can be benefitted from remote server for heavy-weight learning process. Moreover, Arp et al. [93] presented Drebin as a light-weight yet effective analysis framework that can give explanatory results. The features to be analyzed were taken from manifest file and disassembled code. Drebin was evaluated to be remarkable with 94% detection accuracy, ignorable low false positive rate of 1% and very efficient for on-device analysis.

Apposcopy was presented in research by Feng et al. [94] which was based on semantic approach by classification on the basis of data-flow and control-flow features from the manifest file. The malware focused were Trojans that caused private data stealth. Apposcopy's detection accuracy for known malwares was 90%. Furthermore, Huang et al. [95] evaluated various machine learning classifiers and deduced that C4.5 and SVM are better for precision. However, Naïve Bayes causes higher recall rate.

In the work of Chakrade et al. [96] presented MAST that practice multiple correspondence analysis (MCA) to direct the limited resources for malware detection towards potential apps that exhibit malicious behavior. The extracted features include permissions, Intent filters, zip archives, and native code. This approach generates ranking faster than any light-weight analysis. It was designed for specific purpose that only relates with extensive apps in markets. However, it fails to judge and rank zero-day malwares. A signature-based mobile botnet detection algorithm with the basic factor as Bayesian spam filter mechanism was suggested by Vural et al. [97]. The writers evaluated the system to be able of recognizing 87% of spam message from given dataset. Another study by Aswini et al. [98] offered a static analysis mechanism DroidPermissionMiner for the detection of malwares by analyzing permissions of an applications. The study involved analysis of 436 applications files and mined specific features that are related to malicious activities. The proposed model classified the apps based on machine learning classifiers. Moreover, DroidAPIMiner was proposed by Aafer et al. [99] to detect malwares based on the frequency of API calls by each application. It was concluded that the rate at which APIs are called in applications is 6% larger in malware than in benign applications. As K-NN, a machine learning classifier was used to evaluate the performance level.

Most prominent features that are used in literature such as Yerima et al.[100] describe the static features such as permissions and commands to detect the notorious malware from android applications and achieved the 95.8% detection accuracy by using the machine learning classifier. In the work of [101] , they proposed a Droid Detective approach that analyze the requested permissions in benign and malicious applications. Permissions are declared in androidmanifest.xml file that needs to invoke to run an application properly. Specifically, some permissions such as INTERNET, ACCESS_NETWORK_STATAE, and READ_PHONE_STATE are used in both normal and malicious applications. Conversely, some permissions such as WRITE_SMS, RECEIVE_SMS, SEND_SMS, and READ_SMS that are mostly invited by the malwares but hardly in benign apps. In this way, malicious application sends premium rate message without user interference that causes major financial loss. Hence, Droid detective boost the system security and experimental results shows 96% malware detection rate.

In the study of Egele et al. [102] offered a CRYPTOLINT tool that uses static method for examining cryptographic features and hooks common misappropriation of cryptography in android apks. The analysis is performed on 11,748 android applications in which result shows 88% of applications used cryptography unsuitably. Correspondingly, Chatzikonstantinou et al.[103] investigated forty nine android applications in static and dynamic manners that uses cryptographic operations. Accordingly, results indicated 87.8% of the applications exploit cryptography feature. Besides, X-ANOVA [104] method analyze opcode and detect malware with the accuracy of 88.30 percent. As compare to, dynamic analysis it is cost effective solution to analyze the android applications. The obfuscated techniques are used by malware writers to hide from detection mechanism, in that way dynamic analysis is necessary. Consequently, in table 2 noticeable static feature are defined that are used in literature to effectively detect android malware.

**Table 2: Summary of Prominent Static Features by Existing Approaches**

| Categories | File | Features | Ref |
|---|---|---|---|
| Permissions | AndroidManifest.xml | Packages, permissions, strings | [87] [101] |
| Intents | - | intent_filter, broadcast_receivers | [101] |
| Resources | Resources.arsc | Interesting strings | [91] |
| Hashes | Other files | Hash keys, md5 | [102] |
| Cryptographic operations | .xml | Data encryption, cp_traffic, uses_crypto, uses_reflection java packages | [102] [103] |
| Bytecode, opcode | .class | - | [104] |

### 4.1.2 Behavioral or Dynamic Analysis

Dynamic analysis is the method of evaluation of a program by implementing it in run-time. The purpose is to find the behavior of a particular app while it is being executed. The app is observed by actually executing it either on a real device as actual execution or on a virtual environment as the Android Virtual Device. Dynamic analysis is dominated to identify behavioral features for Android apps, yet there is many challenges existed, as event triggers, Android's managed resources, and Binder-based ICC. These are the some more challenging facts in context of dynamic analysis that are applied on applications and analyzed in virtual environment due to following considerations:

(i) Time constraints on executing and observing the behavior;
(ii) Simulation of graphical user interface response and actual system events;
(iii) Malicious apps attempt to avoid Android virtualization;
(iv) An enormous number of apps to be assessed by detection system

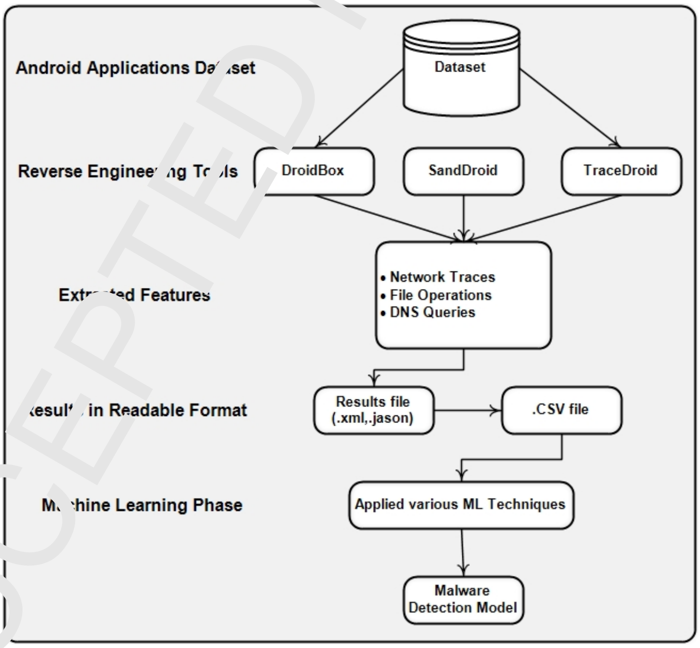


**Figure 6: Common steps of Dynamic Analysis with existing tools**

Aforementioned figure 6 illustrated that common steps and existing dynamic malware analysis tools that were used by researchers commonly in detection mechanism. Andromaly [105], is an on-device dynamic analysis framework was proposed by Shabtai et al. and It is based on processors with machine learning capabilities performing the analysis of running apps to decide the threat level also select the action to be performed. Authors evaluated the system on self-written malware samples, therefore lack in the observed behavior of various actual malwares penetrating the market.

A varying approach with an automatic VMI-based dynamic analysis system presented by Tam et al. [106] is CopperDroid capable of automatic and accurate reconstruction of an applications behavior. It generates detailed behavioral profiles to provide intuitive behavioral properties. Its performance was evaluated on more than 2900 malwares samples from the real world. Similarly, a framework Andlantis [107] proposed by Bierma et al. is a virtualized environment with artificial network data in order to closely replicate a physical device. System calls, forensic footprint left by malware families, runtime behaviors and network traffic of apps are parsed for anomalous behavior. Andlantis is a scalable dynamic analysis system as it can process over 3000 apps per hour by employing minima for parallelism.

The study done by Amos et al. [108] aimed to perform a comparative performance of various machine learning classifiers for dynamic analysis of apps. The study profiled 407 benign and 1330 malicious apps and extracted 6832 features vectors from them for analysis. A feature vector collection framework known as STREAM which allocates the collected feature vector to various emulators and android devices on the cloud for analysis using different ML-classifiers. The authors evaluated Bayes Net to be the best and logistic to be the worst among malware classifiers. Crowdroid [109] system introduced by Burguera et al. that mainly focused on detection of Trojan horses. The authors proposed a framework in which the response is collected from infinite number of real true users by crowd sourcing. Furthermore, the collected data is analyzed with the help of numerous deployed servers. Another theme of this work is to monitor initiated system calls and applies 2-means clustering algorithm to identify app as malicious or benign. The authors tested the proposed system on real malwares along with three self-written malwares and evaluated it to be promising.

AppsPlayground by Rastogi et al. [110] performs kernel-level checking, dynamic taint tracing and API monitoring. Moreover, it sets identifiers and data that are device related, causes to reduce sandbox detection. Event triggering and smart implementation techniques are implemented for comprehensive execution coverage. AppsPlayground is evaluated to achieve a code coverage of 33%. An efficient dynamic method DroidScope [111], used to analyze native code and events that based on three layers such as hardware, Dalvik and Linux. Furthermore, it can analyze APIs how they interact with system and two well-known android malware families: DroidKungFu and DroidDream.

Another study DroidDolphin presented by Wu et al. [112], a framework that comprises four stages: (a) pre-processing, (b) emulation and testing, (c) feature extraction and (d)machine learning. It collects run-time logs and traverse's applications code path. With the training dataset of 32000 benign and 32000 malicious apps and 1000 of each as test dataset, DroidDolphin evaluated to give estimated accuracy of 86.1% and F-Score is 0.86. Moreover, Alam et al. [113] aimed at applying Random Forest classifier for behavioral detection by observing features regarding permission, battery, CPU and memory usage, binder API and network. Experimental results based on 5-fold cross validation method in which Random Forest to be satisfying with an accuracy of over 99% with a training dataset of 407 benign and 1330 malicious apps and 48919 samples used as test dataset.

TaintDroid is a dynamic taint tracking system to detect privacy violations in Android was offered by Enck et al. [114] with four granularities of taint distribution (a) methods, (b) message, (c) variable and (d) file level. TaintDroid marks sensitive data that is likely to be leaked through untrusted apps. Furthermore, its main objective is to mark septic data before leaving the taint sync. It revealed that every two out of three apps used sensitive data suspiciously. However, the scope is limited for the proposed system as it lacks tracking of implicit control flows because of performance overhead. In an addition, a dynamic analysis with help of sandboxing proposed by Desnos et al. [115] can effectively analyze Android applications but excluding those which are released before Android version 4.2. Moreover, droidBox and taintDroid are available as open-source projects.

Another dynamic analysis based approach VetDroid [116] given by Zhang et al.  influences TaintDroid and implement permission-based analysis by executing applications in secure sandbox. The permission analysis module mines all permissions and highlight the connection between them which is then used to generate a Function Call Graph (FCG), for identifying malware. Analysis on 1,249 sample apps in store, VetDroid identified more privacy leaks than TaintDroid and points out the leaks details in context of permission used by different applications.

Livadas et al. [117] applied machine learning classifiers to detect botnet traffic in two stages. First stage distinguishes between IRC and non-IRC traffic, then comes the next stage to differentiate between real and botnet IRC traffic. The authors are concluded that Naïve Bayes performs best. Another study to evaluate the performance of machine learning classifiers was done by Narudin et al. [118] using 1000 malware samples of forty nine Android families from Genome Project along with 50 more apps from other random markets. The authors found Random Forest classifier as a best classifier with 99.99% detection accuracy. Moreover, lightweight system that recognize the dynamic behavior of an application called DroidLogger by Dai et al. [119]detects suspicious behavior using instrumentation by logging program APIs and system calls along with their comprehensive arguments.

Another approach that is used for network-based anomaly detection established on analytical modeling, simulating, learning, accompanied with the billing and control-plan data for anomaly-detection was presented by Abdelrahman et al. [120]. Furthermore, the malware detection performed in the research by Andrewset al.  [121] made a virtual lab environment or emulating  environment to analyze and detect malware. A malware detection technique which is based on traffic flow analysis by Shabtai et al. [122] specially designed for applications with self-updating capabilities. The classification algorithms REPTree and Decision Table were implemented to classify applications as malicious or not according to predetermined legitimate traffic patterns. Portokalidis et al. [123] emerged with a concept of remote server for the first time in malware analysis as Paranoid Android. The authors investigate reconstruction concept during application execution. The device records and sends a nominal execution trace to the security server which then replays the execution traces to perceive any potential malware.

Dynamic features that are frequently used in literatures such as DroidRevealer [124], a light weight approach that operate on the kernel level features and monitor the system level calls to detect the malicious behavior. This technique is divided into three stages, first it monitors both OS level and app level behavior. Second, Linux kernel that is lowest level of android architecture and has peak privileges. So, that was problematic to analyze the applications at this level due to low level application details. In third level, the DroidRevealer can run on real devices to detect runtime behavior. Finally, it shows the concrete results in form of graphs with detection accuracy. Furthermore, Zaman et al. [125] analyze the network traffic by generating the URL table that way all application communicate with the remote server. They analyze the features of .pcap files and match the URLs with blacklisted domains to get the appropriate results. Similarly, MoleDroid [7] detect the malicious behavior from network traffic that consist of outgoing, ingoing and complete flow of data. Furthermore, the result shows 99% true positive rate to detect the information theft in network flows. In the study of [62], they dynamically analyze the System calls attributes such as read, write, network operations, and directory operations to detect the mobile botnet. Consequently, the botnet detection rate shows the 86% and 88% precision and recall respectively in 500ms time window frame. In table 3, we described dynamic features that are mostly used in literature.

**Table: 3 Summary of Prominent Dynamic Features by Existing Approaches**

| **Categories** | **File** | **Features** | **Ref** |
|---|---|---|---|
| API Calls | .json, .xml | getConnect(),getWifiState(), getConnectionInfo(), getCellLocation(), getDeafult(),getSubscriberId() | [62] [126] [44] [127] |

| | | | |
|---|---|---|---|
| Network Operations | .xml,.json | TCP size, Duration, outgoing, ingoing and complete data flow | [62] [7] |
| File Operations | .xml,.json | File read, write, file leaks | [59] [110] |
| Running Services | .xml,.json | started_services | - |
| Network traces | .pcaps | Source IP, destination IP, host, port, path | [115] |
| System Calls | Classes.dex | classes, fields, methods,<br>prototypes, types, strings | [62] [107] [109] [128]<br>[129] [130] |

### 4.1.3 Hybrid Analysis (Static & Dynamic)

The basic theme behind hybrid analysis is to combine the features of static and dynamic analysis that relates to examining code and behavior of an application as depicted in Figure 7. Similarly, It shows that, hybrid analysis overlays the parameters of static and dynamic analysis. Researchers used Static and dynamic tools such as androguard [131], APK inspector [132], Droidbox [133], Sandroid [134], and Tracedroid [135] specifically to extract features from applications as we discussed in mobile malware detection approaches. Although there is fewer research is available in literature in detection of mobile malware that uses hybrid analysis method. The detection accuracy of malware is higher in hybrid analysis as compare to adopting static or dynamic method.

A HybriDroid model [59] that uses both static and dynamic analysis techniques to detect data leakage in android applications. In that way, static analysis examines the source of an applications that causes data loss. On the other hand, dynamic technique monitors behavior of an application at the runtime to detect the malware. At the end, authors compare this model with IccTa [136], DroidGuard[137] and DroidBox [133] tools that proves its effective performance.

Similarly, Spreitzenbarth et al. [126] presented a Mobile-Sandbox that combines the features of static and dynamic methods to detect mobile malware. Static analysis monitors manifest file and decompile the code, whereas in dynamic analysis API calls are monitored. In the work of [129], they also adopted hybrid scheme to analyze the features such as permissions and system calls. The static analysis shows best results by using the SVM classifier as a TPR (true positive rate) of 98.68%. However, dynamic analysis depicted 90.00% accuracy in case of system calls feature.

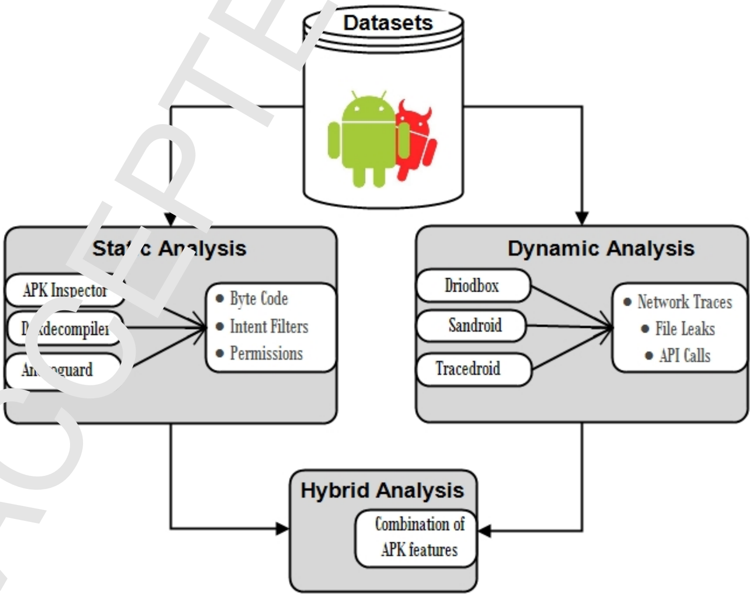


Figure 7: Hybrid Analysis Combination of Static and Dynamic

Moreover, another researcher [138] considered the network traffic analysis as a dynamic and permissions as a static feature to detect malware. Network traffic is captured by Wireshark tool that covers HTTP and TCP conversions. Prominent features in traffic analysis are packet size, average number of bits sent/received, ratio of incoming/ outgoing network bytes is analyzed to detect the normal and malicious application. The combined analysis of static and dynamic features evaluates 95.56% accuracy with trained data set of one hundred and fifteen malicious and normal applications that are collected form malgenome project and google play store respectively.

In addition, SAMADroid [44] model proposed by S. Arshad et al. that is based on hybrid analysis to identify malevolent behavior from permissions, API calls, Network Addresses. They effectively defined the 3-level hybrid structure which is consist of (a)static and dynamic analysis (b) remote and local host and (c) intelligent machine learning techniques in context of detection and prevention methods. This model archives 99.07% accuracy from random forest classifier and they comparatively analyze its performance with MADAM[5].

Correspondingly, another hybrid approach [127] that deals with permissions, API calls and discriminates between good ware and malware applications. In this study, authors find out most frequently used permissions and API calls in malicious applications. The proposed hybrid method is based on the adaptive neural fuzzy interface system with the inclusion of particle swarm optimization. Moreover, datasets of benign and malicious apks are gathered from google play store and other resources identically. Consequently, outcome of their approach shows 89% accuracy that effectively detected android malware applications. Furthermore, Afifi et al. proposed DyHAP [139] approach that works in similar manner and used 1260 malware samples from forty nine android malware families and capture 1,000 network patterns. Hence, they proved optimized results by dealing with complex parameters.

## 4.2 Data Acquisition

This section describes different data acquisition means that researchers have taken to detect malicious behavior of android applications. Data acquisition relates to data repository that is obtained from different sources for experiments. Data can be analyzed statically and dynamically or in hybrid manners at distinct levels using real time, on line data sets and sandboxing.

Android applications are examined at real time to detect mobile malware. PasDroi [140], a real time security mechanism that notifies the user, when malware tries to breach the security. Another approach proposed by Ruan et al. [124] also defined the real time monitoring of android application at the kernel level. In addition, they presented DroidRevealer [124] that can run on real devices to detect the real time behavior. This approach shows concrete results in form of graph with higher detection rate.

The prevalent dataset sources of android apks is contagio-mobile[90] having number of applications. In the study of [47] they taken dataset from contagio mobile dump to extract malicious features from them. In addition, other researchers [100] and [62] used McAfee internal repository and ICSX android botnet dataset respectively. Moreover, emulator is applied on applications to test their behavior and it has an ability to isolates applications from original android framework. Basically, it is used to dynamically analyze android applications. Google provided i.e. Genymotion [141] android emulator, Bluestacks, DuOS, Andy etc. to test applications. You et al.[142] focused on send and receive messages feature to detect SMS botnet by using the android emulator. Similarly, in the work of [47], they uploaded a malicious application on a sandbox sever to capture their suspicious behavior. In addition, island [143] provide sandbox environment to test android applications.

## 4.3 Working Environment

This section will elaborates the dissimilar options used by scientists to perform analysis and detection tasks. The identification and detection of malware is becoming major concern in android platform. As mobile malware authors target different working environments and researchers also implement their experiments on identical platforms to achieve required results. Different working environment includes platform, multiplatform, on device and off device. The two most common approaches of the analysis are (a) on-device and (b) off-device analysis. Similarly, another approach i.e. hybrid analysis is also used by existing studies. In on-device analysis approach an **"apks"** is installed on Android device simultaneously storing the log files and network traces on the device or on the cloud. Another application is used to fetch those log files and network trances and matches with the predefined patterns of malicious applications. Whereas in off-device environment researchers normally used reverse engineering tools to decode applications and matches the signatures with malicious contents.

The multifarious platform consist of android, iPhone, Microsoft and Symbian that are used in development of mobile phones. Among all of these operating systems, android is most pervasive in market and its number of users is growing day by day. According to recent antivirus (AV) security survey 2018 [144], 75.2% are android users, 17.1% are apple/ iPhone users, 4.2% are windows mobile users whereas 3.5% users relates to another operating system (OS).

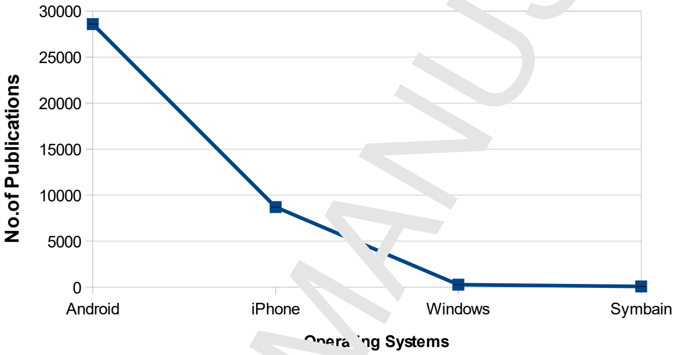


**Figure 8: Number of Publications from the year 2010 to 2018 based on multifarious platforms**

In figure 8, the line graph is presented number of publications with respect to OS type. As a result, it is observed that adoption rate of android is much higher as compare to other OS platforms. Moreover, popularity of android phone is increasing day by day due to its affordability, easy to use, and customized features as well as it is the main target of attackers.

### 4.4 Evasion Techniques

Recent studies shows that malwares tries to evade from detection mechanism, that's why they use evasion techniques such as java reflection, Obfuscation, repackaging, Polymorphism, encryption and control flow alteration. In repackaged applications, a malicious user downloads the legitimate app from legitimate play store, extract its files, add some harmful code and then repack it.

Moreover, Repackaging in android applications is very challenging, because these applications look like original apps. Likewise, obfuscations hides the malicious code that is hardly detected in android apps. Furthermore, Control flow alteration and polymorphism adds some additional code with original code. In control flow alteration sometime malware writer adds dead code that makes tough to detect applications as malicious. While, in polymorphism malware writers insert extra code in encrypted form, as a mutation engine (ME) that uses obfuscation methods to disrupts the normal routine of an application. In table-4 we presented a summary of existing studies that covers mobile malware detection approaches with specific evasion technique.

**Table-4: Summary of Existing Mobile Malware Detection Approaches that deals with evasion Techniques i.e. Obfuscation, polymorphism, Java reflection**

| References/ Approaches | Year | Point of detection. | Data Source | Result | Limitations |
|---|---|---|---|---|---|
| Droidcat [145] | 2018 | Obfuscation, reflection | Praguard, Contagio dataset ,AndroZoo (AZ) Google Play Store | For highly evasive malware families (DroidDream, BaseBridge, and Droid kungFu) with 11 others it gives 97% pricision | ---- |
| HyDroid[146] | 2017 | Obfuscation, Reflection, encryption, Name alteration | Genome Repository | ---- | It does not deal with native code. |
| Ordol[147] | 2017 | Detection of libraries in obfuscated applications | Libraries were collected from Librader ,Appbrain and 1000 apps randomly chosen from google play store | ---- | If the libraries are not comprised in ordol database, then ordol cannot identify it properly even if it is not obfuscated. |
| DroidRA [148] | 2016 | Reflection and uncover dangerous code | 500 application selected from Google play store as well as some malicious samples | DroidRA detect the reflected methods correctly and set as benchmark comparison with IccTA | ---- |
| [149] | 2015 | Detection of Polymorphic malicious applications | ---- | Malicious applications are examined in proposed system, then it generates a message to external server about the polymorphic code | ---- |

DroidCat [145] is a dynamic approach, that effectively detects most evasive malware families such as DroidDream, BaseBridge, and DroidkungFu including with 11 other malware families. In this experiment, they considered reflection, resource and system calls related obfuscated features. Similarly, Yuan et al. proposed an approach ACFinder [150] that relates to API dependency graph (ADG) created on birthmark that can detect repackaged applications. As compare to existing approaches that are based on system dependency graph (SDG) and IFDS, it works more efficiently. However, this approach is unable to deal with obfuscated code if it is encrypted because in static analysis code must need to be decrypt for analysis. Similarly, another novel technique Ordol [147] used to detect libraries from android applications as it is a difficult task. Consequently, it shows better results to detect exact version of libraries in comparison with publicly available LibRadar [151] tool. Despite that, ordol is also capable of detecting libraries even from obfuscated android applications. However, ordol is worthless if all versions of libraries are not included in its database because in that way it is unable to detect.

Furthermore, hydroid [146] is used to detect malicious code that have java reflection, control flow alteration and encryption like evasion techniques. Hydroid method is divided into two phases in which it consists of static and dynamic analysis methods to generate the API call traces even in presence of obfuscation. Moreover, they explained malware families such as Plankton, GlodDream and BeanBot that are using evasion techniques related to obfuscation and non-obfuscation samples. These malware families causes to steal confidential information such as deviceID, IMEI number location, send and receive messages automatically. In this method, results are evaluated only by using fourteen malware families. While, further experimental results are need to be generalize more malware families and make this method able to deal with native code.

DroidRA [148] perform static analysis that deals with reflection in android applications. Experiment is performed on 500 randomly selected applications from third party app store and google play store as well as malicious applications that usually contain reflective calls. Additionally, it is helpful in detecting dangerous code and sensitive API calls. As a result, DroidRA is set as a benchmark in comparison with IccTA [136] for researchers. Moreover, another researcher Lee et al.[149] detected server side polymorphic applications that contain malicious content. Unfortunately, evasion techniques in android applications are not fully investigated by the entire researchers as we discussed it in term of limitations.

## 5. Forensic Analysis of Mobile and Malware Detection

In this section, we presented forensic analysis in the perspective of mobile and malware forensics with their various aspects. As forensic analysis is the investigation of detecting, documenting and collection of evidences. Moreover, it refers to the digital forensic that involves computer forensics, mobile forensics, network forensics, malware forensics etc. [152]. Mobile malware forensics and digital forensics have similarities in perception of deep learning and features extraction. In figure 9, we presented mobile and malware forensic analysis comprehensively with its general procedural steps, tools and challenges individually.

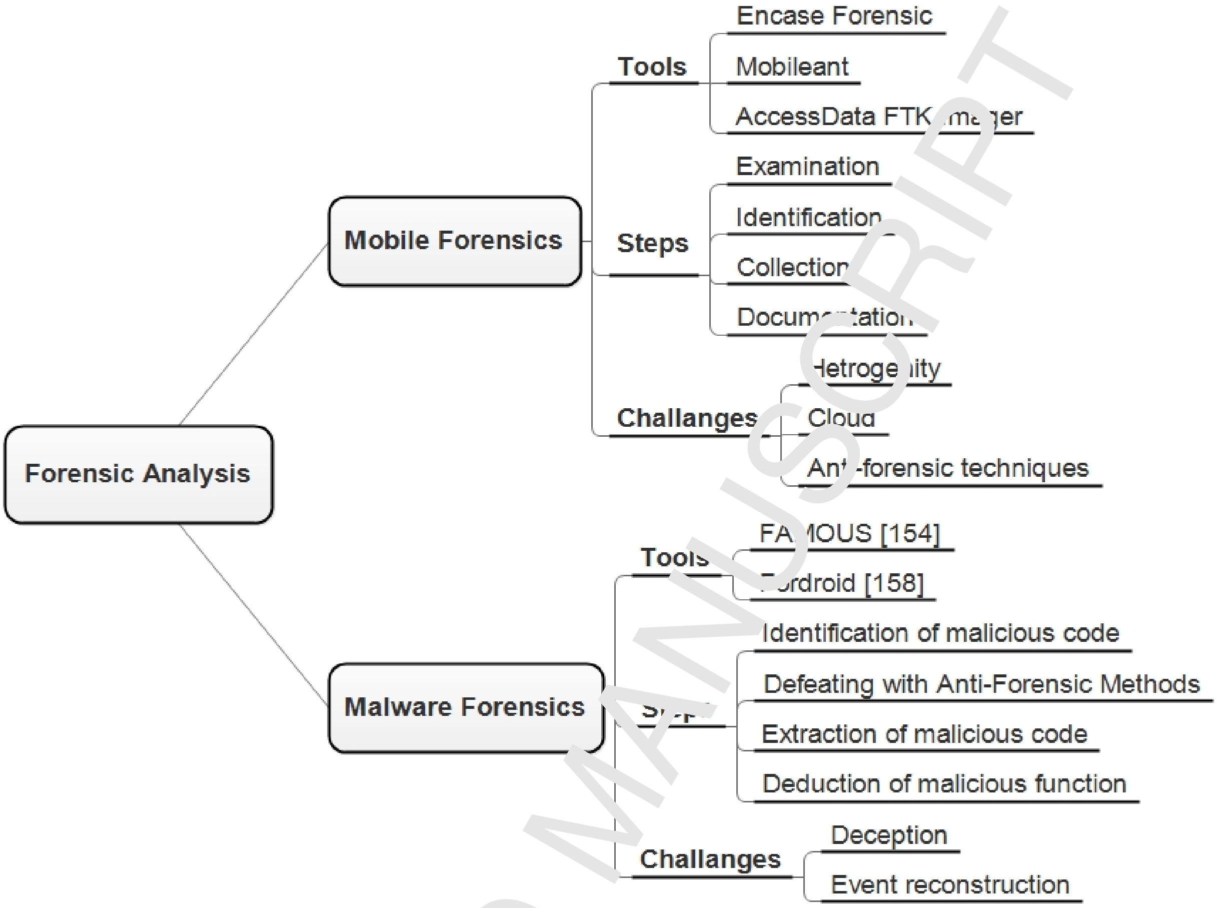


**Figure 9: Forensic Analysis with respect to mobile and malware forensics**

Malware forensics are progressively becoming more significant as cybercriminals causes to initiates malicious activates. There are four common steps have to follow in malware forensics in which (a) identification of suspicious code or program (b) defeating with anti-forensic methods (c) Extraction of malicious code and (d) deduction of malicious function [153]. The anti-forensic methods covers code obfuscation, that hides harmful code and evade from detection mechanism. Moreover, for identification of harmful code some static features has to be analyze critically such as inflationary use of permissions message digest hashes etc. In this context deception is the main challenge for malware forensics in which malware hides its identity and reconstruction of malicious events. Kumar et al. [154] introduces a forensic analysis tool named as FAMOUS for android applications that is based on static method to detect the malware. Authors used scoring method for permissions, dataset of malicious and benign applications is collected from drebin [155], contagio dump [156] and play drone [157] respectively. Consequently, it shows 94% accuracy by using random forest classifier. However, they overlooked other static feature including string, function calls and metadata. Similarly, another forensic tool Fordroid [158] used to analyze android applications and identify where the information is stored.

Fordroid, monitor a hundred applications that is collected from APPchina [159]. As a result, it successfully locate the sensitive information with 98%. Mobile forensics comprises the term digital evidences of mobile devices using accepted forensic methods. It is more challenging because heterogeneity and diverse feature set used in mobile devices technology [160]. Correspondingly, feature set is beyond from calling and messages feature. As couple of mobile devices used cloud services to store and retrieve data then it should be available all the time for evidences in case of investigation process. Furthermore, in the work of [161] they defined four general steps that follows in mobile forensics (a) examination (b) identification (c) collection and (d) documentation. The examination phase ensures the analysis of devices with its synchronized devices. In identification phase, process is started by recognizing the device type along with running operating system. Additionally, collection of evidences and proper documentation on the every stage of mobile forensics is provided. Alhassan et al. [162] comparatively analyze the mobile forensics tools named as Encase forensics, mobiledit and Access FTK Imager with respect to performance on different operating systems. In addition, these tools have access to different activities of device. However, they did not able to access SIM card and sometime not retrieved deleted data. Therefore, we required effective and efficient tool for mobile and malware forensics.

## 6. Artificial Intelligence towards Malware Detection

In this section, we will discuss artificial intelligence that is an umbrella term under which machine learning and deep learning take place. According to ESET report [163], a meek search of the word 'AI' gives 2.2 billion results and proved as sustainable in cybersecurity. Furthermore, it is stated that, cybercriminals and other malicious performers are now aware from benefits of artificial intelligence and they try to implement it in their malicious activities. Moreover, artificial intelligence also included machine and deep learning-based solutions.

In figure 10, we elaborated artificial intelligence with its sub types that are used by researchers in literature to evaluate their approaches. In below subsections, we described that how machine learning algorithms outperforms in detection of mobile malware. Unfortunately, machine learning has some limitations that's why we need deep learning algorithms to deal with cyberattacks that are described in section 6.3.

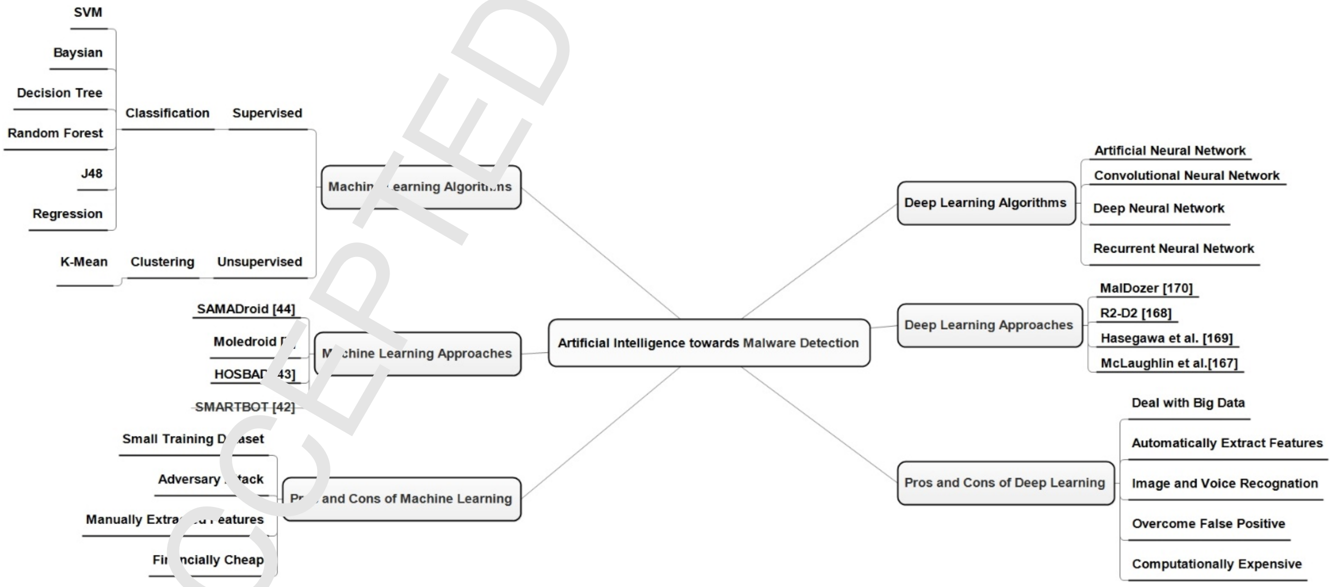


**Figure 10: Artificial Intelligence in malware detection**

### 6.1 Machine Learning Algorithms with Malware Detection Approaches

Machine learning (ML) approaches have been effectively applied in mobile malware identification and detection [7] [46, 47] [61] [62] [100] [130, 138] . ML techniques played a vital role to develop an intelligent system that can distinguish between both malicious and benign android application. Machine learning approaches are well-defined into two types that are *supervised* and *unsupervised*. Supervised term relates to classification algorithms i.e. SVM, Neural Networks, Baysian, Decision Tree, Random Forest, J48, Regression etc. Whereas unsupervised relates to clustering algorithms i.e. K-Mean, X-Mean, etc.[60] is described in Table 5.

**Tabe-5: Summary of Existing Mobile Malware Detection Approaches with Heterogeneous Machine Learning Methods**

| References/ Approaches | Year | Analysis Type | Features | Dataset Source | Targeted Malware | ML Classifiers/ Clustering | Accuracy (%) |
|---|---|---|---|---|---|---|---|
| [46] | 2013 | Static | INTENET, SEND_SMS, CHANGE_CONFIGURATION, WRITE_SMS, and CALL_PHONE | 200 samples of good ware and malware Android applications | Android malware binaries | K-Means, Random Forest and Regression Tree | -- |
| [164] | 2013 | Dynamic | Networks traffic (TCP size, duration, no of GET/POST request) | Network traffic that collects by using tPacketCapture | Mobile botnet | Naïve Bayes, K-NN, SVM, DT, and MLP | K-NN shows 99.94% detection rate |
| Composite parallel classifier[100] | 2014 | Static | API Calls, Permissions, Commands | Benign & malicious apps from McAfee internal repository | Malicious intensions | PART | 95.8% accuracy |
| [130] | 2014 | Static | Five sets of features including (System calls ,kernel based) | -- | Mobile Malware | MLP, J48,KNN,RF,NB | 83% accuracy in MLP |
| HOSBAD[43] | 2015 | Dynamic | SMS, CALLS, Device Status | Malicious and normal application dataset | Different types of suspicious behavior | K-NN | 93.75% accuracy |
| SMARTbot[42] | 2016 | Dynamic | -- | Drebin | Botnet | Random forest, Naïve bayes, SVM,648, simple logistic regression, & MLP | Random forest classifier gives the 99.49 % accuracy |
| [47] | 2016 | Hybrid | System call(permissions, sensitive function, intent priority, native permission) & short messages | 300 applications from Contagio Mobile dump & 70 malware apps to analysis SMS feature | -- | Logistic Regression | 91% accuracy in static , 41 out of 70 messages send without user knowledge |
| Moledroid [7] | 2017 | Dynamic | Network traffic flow(incoming, outgoing, complete) | Suspicious and normal application | Information theft | Random Forest | 99.1% precision |
| Host Based Mobile botnet detection[62] | 2017 | Dynamic | System calls(read, write, network operations, directory operations) | 13 botnet families with 31 apps from ICSX android botnet dataset | Android botnet | Random Forest | 86% precision , 88% recall in 500ms time window |
| Venn Abres Prediction[61] | 2017 | Static | Network communication traffic (HTTP based,IRC based,P2P based) | -- | Botnet | KNN & KDE | -- |
| [138] | 2017 | Hybrid | Permissions & Network traffic | Malgenome & Google Play Store | Android Malware | Decision Tree | 95.56% accuracy |
| SAMADroid [44] | 2018 | Hybrid | Permissions, API Calls, Network Addresses | Drebin | Mobile Malware | Random Forest | 99.07% Accuracy |

Generally, different types of malware detection techniques are employed. There are number of researches based on machine learning algorithms that can statically, dynamically or in hybrid manners can detect malicious behavior from android applications. Aung et al.[46] and Yerima et al.[100] adopted static analysis method to detect android malware binaries. They analyze different features such as permissions (INTENET, CHANGE_CONFIGURATION, SEND_SMS, CALL_PHONE, and WRITE_SMS), API calls and features related to commands execution. Furthermore, they applied machine learning clustering and heterogeneous classification algorithms (i.e. K-Means, Random Forest, regression tree, naïve Bayes, decision tree, simple logistic, PART, and RIDOR) in order to evaluate results.

However, another researcher[47] analyze the malicious behavior in both static and dynamic manners to identify android malware. Static analysis declares these four types of features as intent priority, code permissions, native permissions, and sensitive functions calls in this study. Furthermore, a behavior of the application is tested at runtime using an android emulator. The evaluated results shows that forty one out of seventy malware sends messages automatically without user awareness. Additionally, logistic regression classifier in static analysis is revealed 91% accuracy.

Feizollah et al.[164] and another author[7] dynamically monitor the network traffic flow with their extracted features to detect mobile botnet malware. Consequently, K-NN classifier shows the 99.94% detection rate of malware in network traffic. Furthermore, they developed an application named as MoleDroid to identify the malicious behavior of network flow. The result endorsed 99% true positive rate for detecting the information theft in network flows. Similarly, Whang et al. [61] presented a venn-abers predictor for botnet detection in network traffic flow using K-Nearest Neighbor (KNN) and kernel Density Estimation (KDE) as an underlying scoring classifiers. Similarly, it covers the network traffic flow features as HTTP, IRC, and P2P that relates to botnet families. By the use of evasion techniques most of data hardly detect as malware. To attain highly comprehend results, we have to deeply analyze the evasion techniques in future.

Karim et al. [42] designed a vital dynamic framework to detect botnet applications named as SMARTbot. They used malware dataset from Drebin, extract their features and random forest classifier gives the 99.49 % detection accuracy. Similarly, Costa et al.[62] presented the anomaly and host based detection approach and considered thirteen botnet families with their thirty one applications samples that is collected from ICSX android botnet [165]. The random forest classifier shows great performance in detection of botnet applications. Recently, SAMADroid [44], approach is presented that covers static and dynamic features and achieves high malware detection with 99.07% accuracy. Hence, these are several studies conducted to identify and detect android malware by using the machine learning algorithms.

### 6.2 Pros and Cons of Machine Learning

ESET [163] defines some limitations of machine learning techniques. In which they needs to correctly label the data and do not have ability to deal with big datasets, as deep learning can more effectively deal with large datasets. Moreover, cybercriminals are using machine learning method in adversary attacks and causes to initiate steganography attack in that way they hide malicious code into pictures, in this case machine learning algorithms did not work. Correspondingly, deep learning algorithms overcome the false positives and provide solution to where machine learning did not perform well.

### 6.3 Deep Learning Algorithms with Malware Detection Approaches

Deep learning algorithms work on fully connected neural network and outperforms in detection of mobile malware as given in Table 6. The well-known deep learning algorithms are convolutional neural network (CNN), artificial neural network (ANN), and recurrent neural network (RNN) etc.

**Tabe-6: Summary of Existing Mobile Malware Detection techniques with Heterogeneous Deep Learning Algorithms**

| Reference | Year | Features | Dataset Source | Dataset | | Deep Learning Algorithms | Result |
|---|---|---|---|---|---|---|---|
| | | | | Malware | Benign | | |
| [166] | 2017 | API calls | Contagio Mobile Dataset | 216 | 1016 | CNN | 99.4% Accuracy |
| [167] | 2017 | Op code | McAfee Labs | 9902 | 9268 | CNN | 87% Accuracy |
| R2-D2 [168] | 2017 | -- | Leopard Mobile Inc. | Over one million | One million | CNN | 93% Accuracy |
| [169] | 2018 | -- | AMD & Drebin Dataset | 5000 | 2000 | 1-D CNN | 96-97% Accuracy |
| MalDozer [170] | 2018 | -- | Malgenome, Drebin, Virus share, Contagio minidump | 33,066 | 37,627 | ANN | 96.29% Recall |

The table 6 illustrates that, Hasegawa et al. [169] proposed a light weight model by practicing one dimensional convolutional neural network in android malware detection. They performed experiment on two different datasets as malware and benign with 5000 and 2000 samples individually. Results demonstrated the 97.04% accuracy rate in detection of malware. Similarly, other authors [166, 167] focused on system calls, op code and use convolutional neural network (CNN) in comparison with long short term memory (LSTM) for classification. Their experimental results confirmed 99.4% and 87% accuracy respectively. Furthermore, another author [168] proposed R2-D2 approach based on CNN to detect mobile malware and perform experiment over one million datasets. Android apks are converted into images and fed to CNN for classification and malware detection. With large dataset, result shows 96% accuracy. Last but not the least, malDozer [170] a deep learning framework perform experiment on approximately 33000 malware and 37,000 benign datasets. MalDozer effectively deployed on mobile and Iot devices and shows 96% recall rate.

### 6.4 Pros and Cons of Deep Learning

Deep learning can deal with big data sets and no need to manually extract the features. The performance results of deep learning is comparatively high from machine learning approaches. Furthermore, it overcomes the false positive rate, provide accurate results and no need to label data. However, it is computationally expensive as it uses graphics processing unit (GPUs) to accomplish billions of operations in minimum time.

## 7. Future Research Directions

In this section, we present some challenges in perspective of fortifying devices and conceivable forthcoming research directions that can be considered as a future work by researchers.

- **Outrageous reverse engineering techniques:** There is an acute need to make reverse engineering more difficult in context of those malware authors who uses repackaging methods to add malicious code by disassembling the android applications. A use of crypto trick is required, that create harder path to reverse engineer any application and apply encryption algorithms to impel the code of an application unconditionally opaque.

- **Extensive hybrid analysis tool:** A robust and efficient tool is required to analyze the applications and notify users if it contain obfuscated code that causes malicious activities. In obfuscation method, malwares are hide themselves and tries to evade from detection process. Moreover, malware authors uses polymorphic techniques and insert dead code into the original application code. As Static analysis is unable to detect obfuscated code, therefore at this stage dynamic analysis should be mandatory to recognize malicious code.
- **Solution of Code coverage problems:** Dynamic analysis did not cover all traits of code such as native code and when code is dynamically loaded. Malware writers choose hardest path for execution of malicious activities that helps them to escape from detection mechanisms. Hence, in future stiff hybrid analysis and all versions of android OS are need to be analyzed with full code coverage for effective malware detection solutions.
- **Tool to detect zero-day attacks:** We required highly intelligent and light weight procedures that have an ability to examine behavior of applications at the run time and identify zero-day attacks. The existing methods are complicated and take long time in malware detection. However, this kind of detection procedures are not so easy to build, unless it required optimistic research directions.
- **Identical datasets with updated malware families:** As time flies, there is a striking increase in android malwares and its variants. Hence, we required standardized and updated datasets to perform malware detection analysis in an efficient way. Moreover, restructured and new simplified datasets is help to detect emerging malwares that are increases with every passing day. Now it's essential to develop multilevel procedures that deals with advance malware attacks and capable to detect and mitigate malicious mobile applications.
- **Laws and regulations:** Governments should make legislations to deal with information security along with cybersecurity related attacks and enforce them at international level. Recently, Federal Bureau investigation (FBI) [1] declared cybersecurity law for internet of things (IoT) devices in which they states that “if you found any toy is being compromised security terms then you have to report it.” Such methods are need to be implemented in mobile devices security.
- **Cybersecurity insurance:** The emerging rate of malware attacks such as distributed denial of services (DDos), phishing, ransomware, or any other ways that causes financial loss to individuals. Consequently, proper insurance structure should be applied on every scale of business that will surely mitigate the risk factor. Moreover, cybersecurity insurance covers cost of accidental attacks and hardware damage.
- **Big data challenge:** As big data familiarized with special malware detection challenges and machine learning techniques are unable to deal with big data. Therefore, we required efficient deep learning methods that overcome this issue and reduce the computational overhead with affordable cost.
- **Hardware attack detection:** In literature we have not still found any tools to detect hardware loss in terms of battery life failures and background consumption of CPU cycles. Moreover, virtual devices are also targeted by malware authors and we have not any other mechanism of safety expect to replace them. Hence, there is a need to devise effective and simple methods for stopping and cleaning untiring malware.

## 8. Conclusion

Mobile malware is progressing enormously with the same growth as mobile applications are designed and published on play-stores. Further the technological push of hardware (smartphones) and system software (e.g. Android OS) is also causing malware writers to introduce new ways and techniques in order to evade antivirus security traps (signatures). This paper uncovers all efforts towards mobile malware creation, propagations, dissemination and detection. The well-defined taxonomies are comprehensively presented and discussed the need to deteriorate its harmful impact on community. Furthermore, in this paper forensic analysis and the research work conducted during years 2013-2019 in the domain of mobile malware analysis along with artificial intelligence detection techniques are discussed. At the end, we suggest some future directions for researchers that helps to develop more accurate, efficient, robust and scalable mechanism in perspective of android malware detection.

## References

1. Portal, S.T.S. *Number of mobile phone users worldwide from 2013 to 2019 (in billions)*. 2018; Available from:

https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/.

2. Popper, B. *Google announces over 2 billion monthly active devices on Android*. May 17,2017; Available from: https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users.
3. Lueg, C. *Some 343 new Android malware samples every hour in 2017*. 2/20/2018; Available from: https://www.gdatasoftware.com/blog/2018/02/30491-some-343-new-android-malware-samples-every-hour-in-2017.
4. Ahn, A. *How we fought bad apps and malicious developers in 2017*. 1/30/2018 3/20/2018]; Available from: https://android-developers.googleblog.com/2018/01/how-we-fought-bad-apps-and-malicious.html.
5. Saracino, A., et al., *Madam: Effective and efficient behavior-based android malware detection and prevention.* IEEE Transactions on Dependable and Secure Computing, 2016.
6. Kwon, J., et al. *Droidgraph: discovering android malware by analyzing semantic behavior*. in *Communications and Network Security (CNS), 2014 IEEE Conference on*. 2014. IEEE.
7. Cheng, Z., et al. *Detecting Information Theft Based on Mobile Network Flows for Android Users*. in *Networking, Architecture, and Storage (NAS), 2017 International Conference on*. 2017. IEEE.
8. Hussain, M., et al., *Conceptual framework for the security of mobile health applications on android platform.* Telematics and Informatics, 2018. **35**(5): p. 1335-1354.
9. Feizollah, A., et al., *A review on feature selection in mobile malware detection.* Digital Investigation, 2015. **13**: p. 22-37.
10. Tan, D.J., T.-W. Chua, and V.L. Thing, *Securing android: a survey, taxonomy, and challenges.* ACM Computing Surveys (CSUR), 2015. **47**(4): p. 58.
11. Yan, P. and Z. Yan, *A survey on dynamic mobile malware detection.* Software Quality Journal, 2017: p. 1-29.
12. and K. Ohlson. *First Palm Virus Found*. 25 September, 2000 12:01; Available from: https://www.computerworld.com.au/article/78795/first_palm_virus_found/.
13. Lindorfer, M., et al. *Andrubis--1,000,000 apps later: A view on current Android malware behaviors*. in *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014 Third International Workshop on*. 2014. IEEE.
14. Symantec. *AndroidOS.FakePlayer*. 3/21/2018]; Available from: https://www.symantec.com/security_response/writeup.jsp?docid=2010-081100-1646-99.

15. Irinco, B. *First Android Trojan in the Wild*. August 10, 2010 Available from: https://blog.trendmicro.com/trendlabs-security-intelligence/first-android-trojan-in-the-wild/.
16. PEREZ, S. *Tap Snake Game in Android Market is Actually Spy App (UPDATE)*. August 17, 2010 Available from: https://readwrite.com/2010/08/17/tap_snake_game_in_android_market_is_actually_spy_app/.
17. Cluley, G. *First iPhone worm discovered – ikee changes wallpaper to Rick Astley photo*. 08 NOV 2009; Available from: https://nakedsecurity.sophos.com/2009/11/08/iphone-worm-discovered-wallpaper-rick-astley-photo/.
18. Leopando, J. *ZeuS Now Bypasses Two-Factor Authentication*. 2010; Available from: https://blog.trendmicro.com/trendlabs-security-intelligence/zeus-now-bypasses-two-factor-authentication/.
19. ZORABEDIAN, J. *Check out this infographic showing the history of mobile threats, 2004-2015*. 19/05/2015 3/21/2018]; Available from: https://news.sophos.com/en-us/2015/05/19/check-out-this-infographic-showing-the-history-of-mobile-threats-2004-2015/.
20. Unuchek, R., *IT threat evolution Q2 2017. Statistics*. 2017.
21. BRENNER, B. *2018 Malware Forecast: ransomware hits hard, continues to evolve*. 2017 3/21/2018]; Available from: https://news.sophos.com/en-us/2017/11/02/2018-malware-forecast-ransomware-hits-hard-crosses-platforms/.
22. Campbell, N. *RedDrop: the blackmailing mobile malware family lurking in app stores*. 2018; Available from: https://www.wandera.com/blog/reddrop-malware/.
23. CERTIFICATIONS, G. *Global Information Assurance Certification Paper*. 3/21/2018]; Available from: https://www.giac.org/paper/gsec/520/liberty-crack-first-palm-os-trojan-horse/100917.
24. Dunham, K., *Mobile malware attacks and defense*. 2008: Syngress.
25. Power, M.T.i. *Virus Profile: WinCE/Infojack*. 3/21/2018]; Available from: https://home.mcafee.com/virusinfo/virusprofile.aspx?key=144191.
26. Zhang, V. *'GODLESS' Mobile Malware Uses Multiple Exploits to Root Devices*. June 21, 2016 3/21/2018]; Available from: https://blog.trendmicro.com/trendlabs-security-intelligence/godless-mobile-malware-uses-multiple-exploits-root-devices/.
27. Kumar, M. *Bad Rabbit: New Ransomware Attack Rapidly Spreading Across Europe*. Tuesday, October 24, 2017 3/21/2018]; Available from: https://thehackernews.com/2017/10/bad-rabbit-ransomware-attack.html.
28. Team, C.P.M.R. *The Judy Malware: Possibly the largest malware campaign found on Google Play*. 2017; Available from: https://blog.checkpoint.com/2017/05/25/judy-malware-possibly-largest-malware-campaign-found-google-play/.
29. Xu, E. *New GnatSpy Mobile Malware Family Discovered*. 2017; Available from: https://blog.trendmicro.com/trendlabs-security-intelligence/new-gnatspy-mobile-malware-family-discovered/.
30. DuPaul, N. *Common Malware Types: Cybersecurity 101*. OCTOBER 12, 2012 3/21/2018]; Available from: https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101.
31. La Polla, M., F. Martinelli, and D. Sgandurra, *A survey on security for mobile devices.* IEEE communications surveys & tutorials, 2013. **15**(1): p. 446-471.
32. Koriat, A.P.a.O. *HummingBad: A Persistent Mobile Chain Attack*. 2016; Available from: https://blog.checkpoint.com/2016/02/04/hummingbad-a-persistent-mobile-chain-attack/.
33. NJCCIC. *BOTNETS*. 2018; Available from: https://www.cyber.nj.gov/threat-profiles/botnets/#LIST-OF-KNOWN-BOTNETS.

34. RISKIQ, *The Q4 2017 Mobile Threat Landscape Report*, 2017.

35. NJCCIC. 2018; Available from: https://www.cyber.nj.gov/search?q=adware&f_collectionId=577f9817f7e0abcfe31befe.

36. Symantec, *ISTR*

*Internet Security*

*Threat Report*, 2017.

37. Buntinx, J. *Top 4 Types of Android Ransomware*. January 20, 2017; Available from: https://themerkle.com/top-4-types-of-android-ransomware/.

38. F-Secure. *Brador*. Available from: https://www.f-secure.com/v-descs/brador.shtml.

39. Symantec. *Spyware.FlexiSpy*. [cited 2018; Available from: https://www.symantec.com/en/sg/security-center/writeup/2006-033012-3337-99.

40. Olzak, T., *Wireless Handheld Device Security.* world, 2005.

41. Fuchs, A.P., A. Chaudhuri, and J.S. Foster, *Scandroid: Automated security certification of android*, 2009.

42. Karim, A., R. Salleh, and M.K. Khan, *SMARTbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications.* Plos one, 2016. **11**(3): p. e0150077.

43. Abah, J. and W. O V, *A machine learning approach to anomaly-based detection on android platforms.* arXiv preprint arXiv:1512.04122, 2015.

44. Arshad, S., et al., *SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System.* IEEE Access, 2018.

45. Karthick, S. and S. Binu, *Static Analysis Tool for Identification of Permission Misuse by Android Applications.* International Journal of Applied Engineering Research, 2017. **12**(24): p. 15169-15178.

46. Aung, Z. and W. Zaw, *Permission-based android malware detection.* International Journal of Scientific & Technology Research, 2013. **2**(3): p. 228-234.

47. Su, M.-Y. and K.-T. Fung. *Detection of android malware by static analysis on permissions and sensitive functions*. in *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on*. 2016. IEEE.

48. Zaidi, S.F.A., et al., *A Survey on security for smartphone device.* IJACSA) International Journal of Advanced Computer Science and Applications, 2016. **7**: p. 206-219.

49. Shezan, F.H., S.F. Afroze, and A. Iqbal. *Vulnerability detection in recent Android apps: An empirical study*. in *Networking, Systems and Security (NSysS), 2017 International Conference on*. 2017. IEEE.

50. Maiti, A., et al., *Side-Channel Inference Attacks on Mobile Keypads using Smartwatches.* IEEE Transactions on Mobile Computing, 2018.

51. Yu, J., M. Ryan, and C. Cremers, *Decim: Detecting endpoint compromise in messaging.* IEEE Transactions on Information Forensics and Security, 2018. **13**(1): p. 106-118.

52. MCGHEE, B. *These 5 apps are killing your battery*. 2018; Available from: https://www.androidpit.com/battery-draining-apps.

53. Chrysaidos, N. *New Avast survey shows that over 50% of consumers cannot distinguish real apps from fake apps*. 2018 27 February 2018; Available from: https://blog.avast.com/mobile-security-and-new-data-on-risk-of-banking-trojans.

54. Jung, W., et al. *DevScope: a nonintrusive and online power analysis tool for smartphone hardware components*. in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. 2012. ACM.

55. Lockheimer, H., *Android and security*, 2012.

56. Karim, A., et al., *On the analysis and detection of mobile botnet applications.* 2016.

57. Zhou, Y. and X. Jiang. *Dissecting android malware: Characterization and evolution*. in *Security and Privacy (SP), 2012 IEEE Symposium on*. 2012. IEEE.

58. Morelli, O. *RedDrop malware is used to steal personal Android users' information*. 2018-03-03; Available from: https://www.2-spyware.com/reddrop-malware-is-used-to-steal-personal-android-users-information.

59. Chen, H., et al. *Automatic privacy leakage detection for massive android apps via a novel hybrid approach*. in *Communications (ICC), 2017 IEEE International Conference on*. 2017. IEEE.

60. Stevanovic, M. and J.M. Pedersen, *Machine learning for identifying botnet network traffic.* Networking and Security Section, Department of Electronic Systems, Aalborg University, Tech. Rep., 2013.

61. Zhi, W., et al., *Fortifying Botnet Classification based on Venn-Abers Prediction.* DEStech Transactions on Computer Science and Engineering, 2017(cst).

62. da Costa, V.G., et al. *Detecting mobile botnets through machine learning and system calls analysis*. in *Communications (ICC), 2017 IEEE International Conference on*. 2017. IEEE.

63. Chrysaidos, N. *Malicious mobile BankBot Trojan injected into everyday apps, taking advantage of unknowing users whose banking apps could be compromised*. 2017 20 November 2017; Available from: https://blog.avast.com/mobile-banking-trojan-sneaks-into-google-play-targeting-wells-fargo-chase-and-citibank-customers.

64. Bojjagani, S. and V. Sastry. *VAPTAi: A Threat Model for Vulnerability Assessment and Penetration Testing of Android and iOS Mobile Banking Apps*. in *Collaboration and Internet Computing (CIC), 2017 IEEE 3rd International Conference on*. 2017. IEEE.

65. Dini, G., et al., *Risk analysis of Android applications: A user-centric solution.* Future Generation Computer Systems, 2018. **80**: p. 505-518.

66. Chong, I., et al., *Influence of privacy priming and security framing on mobile app selection.* Computers & Security, 2018. **78**: p. 143-154.

67. Felt, A.P., et al. *A survey of mobile malware in the wild*. in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. 2011. ACM.

68. Wei, T.-E., et al. *DroidExec: Root exploit malware recognition against wide variability via folding redundant function-relation graph*. in *Advanced Communication Technology (ICACT), 2015 17th International Conference on*. 2015. IEEE.

69. Infradata. *Top Five Cyber Security Threats in 2019*. October 24 2018; Available from: https://www.infradata.nl/en/news-blog/top-5-cyber-security-threats-in-2019/.

70. Maiorca, D., et al. *R-PackDroid: API package-based characterization and detection of mobile ransomware*. in *Proceedings of the Symposium on Applied Computing*. 2017. ACM.

71. Ferrante, A., et al. *Extinguishing Ransomware-a Hybrid Approach to Android Ransomware Detection*. in *The 10th International Symposium on Foundations Practice of Security*. 2017.

72. Khandelwal, S. *This Malware Can Secretly Auto-Install any Android App to Your Phone*. Friday, November 20, 2015; Available from: https://thehackernews.com/2015/11/android-malware-auto-install.html#

73. N, B. *A Malvertiser called "RoughTed" Bypass Ad-blocker and Get Half a Billion visits in 3 Months*. May 25, 2017; Available from: https://gbhackers.com/a-malvertiser-called-roughted-bypass-ad-blocker-and-get-half-a-billion-visits-in-3-months/.

74. N, B. *Malicious Android ads leads to Automatically Download and Install Apps that Contain Malware in Android Devices*. 2017 June 8, 2017 Available from: https://gbhackers.com/malicious-android-ads-leads-to-automatically-download-and-install-apps-that-contain-malware-in-android-devices/.

75. Sanders, C., A. Shah, and S. Zhang, *Comprehensive analysis of the android google play's auto-update policy*, in *Information Security Practice and Experience*. 2015, Springer. p. 365-377.

76. Dosal, E. *Top 5 Cybersecurity Threats and Vulnerabilities*. May 17, 2018; Available from: https://www.compuquip.com/blog/top-5-cybersecurity-threats-and-vulnerabilities.

77. Chiew, K.L., K.S.C. Yong, and C.L. Tan, *A survey of phishing attacks: their types, vectors and technical approaches.* Expert Systems with Applications, 2018.

78. Potter, K., *Increased Use of Two-Factor Authentication Force New Social Engineering Tactics*, 2018, Utica College.

79. Wandera, *Mobile Phishing Report 2018* 2018.

80. Investigation, F.B.o. *CONSUMER NOTICE: INTERNET-CONNECTED TOYS COULD PRESENT PRIVACY AND CONTACT CONCERNS FOR CHILDREN*. July 17, 2017; Available from: https://www.ic3.gov/media/2017/170717.aspx.

81. Forum, U.E.F.I. and M. Pages. *UEFI TECHNOLOGY EXPANDS IN MOBILE DEVICES AND OTHER NON-PC MARKET SEGMENTS*. 2013; Available from: http://www.uefi.org/press-release/UEFI_Specifications_Expand_in_Mobile_Devices_and_Non-PC_Markets_May_8_2013.

82. Ashraf, J. and S. Latif. *Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques*. in *Software Engineering Conference (NSEC), 2014 National*. 2014. IEEE.

83. Poeplau, S., et al. *Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications*. in *NDSS*. 2014.

84. Arzt, S., et al., *Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps.* Acm Sigplan Notices, 2014. **49**(6): p. 259-269.

85. Yang, Z., et al. *Appintent: Analyzing sensitive data transmission in android for privacy leakage detection*. in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013. ACM.

86. Grace, M., et al. *Riskranker: scalable and accurate zero-day android malware detection*. in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. 2012. ACM.

87. Yerima, S.Y., S. Sezer, and G. McWilliams, *Analysis of Bayesian classification-based approaches for Android malware detection.* IET Information Security, 2014. **8**(1): p. 25-36.

88. Wei, F., S. Roy, and X. Ou, *Amandroid: A precise and general inter-component data flow analysis framework for security vetting of Android apps.* ACM Transactions on Privacy and Security (TOPS), 2018. **21**(3): p. 14.

89. Wu, D.-J., et al. *Droidmat: Android malware detection through manifest and api calls tracing*. in *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*. 2012. IEEE.

90. Mobile, C. 2018; Available from: http://contagiominidump.blogspot.com/.

91. Samra, A.A.A., K. Yim, and O.A. Ghanem. *Analysis of clustering technique in android malware detection*. in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*. 2013. IEEE.

92. Schmidt, A.-D., et al. *Static analysis of executables for collaborative malware detection on android*. in *Communications, 2009. ICC'09. IEEE International Conference on*. 2009. IEEE.

93. Arp, D., et al. *DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket*. in *Ndss*. 2014.

94. Feng, Y., et al. *Apposcopy: Semantics-based detection of android malware through static analysis*. in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 2014. ACM.

95. Huang, C.-Y., Y.-T. Tsai, and C.-H. Hsu, *Performance evaluation on permission-based detection for android malware*, in *Advances in Intelligent Systems and Applications-Volume 2*. 2013, Springer. p. 111-120.

96. Chakradeo, S., et al. *Mast: Triage for market-scale mobile malware analysis*. in *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. 2013. ACM.
97. Vural, I. and H. Venter. *Mobile botnet detection using network forensics*. in *Future Internet Symposium*. 2010. Springer.
98. Aswini, A. and P. Vinod. *Droid permission miner: Mining prominent permissions for Android malware analysis*. in *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*. 2014. IEEE.
99. Aafer, Y., W. Du, and H. Yin. *Droidapiminer: Mining api-level features for robust malware detection in android*. in *International conference on security and privacy in communication systems*. 2013. Springer.
100. Yerima, S.Y., S. Sezer, and I. Muttik. *Android malware detection using parallel machine learning classifiers*. in *Next Generation Mobile Apps, Services and Technologies (NGMAST), 2014 Eighth International Conference on*. 2014. IEEE.
101. Liang, S. and X. Du. *Permission-combination-based scheme for android mobile malware detection*. in *Communications (ICC), 2014 IEEE International Conference on*. 2014. IEEE.
102. Egele, M., et al. *An empirical study of cryptographic misuse in android applications*. in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013. ACM.
103. Chatzikonstantinou, A., et al. *Evaluation of cryptography usage in android applications*. in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
104. Raphael, R., P. Vinod, and B. Omman. *X-ANOVA ranked features for Android malware analysis*. in *India Conference (INDICON), 2014 Annual IEEE*. 2014. IEEE.
105. Shabtai, A., et al., *"Andromaly": a behavioral malware detection framework for android devices.* Journal of Intelligent Information Systems, 2012. **38**(1): p. 161-190.
106. Tam, K., et al. *CopperDroid: Automatic Reconstruction of Android Malware Behaviors*. in *NDSS*. 2015.
107. Bierma, M., et al., *Andlantis: Large-scale Android dynamic analysis.* arXiv preprint arXiv:1410.7751, 2014.
108. Amos, B., H. Turner, and J. White. *Applying machine learning classifiers to dynamic android malware detection at scale*. in *Wireless communications and mobile computing conference (iwcmc), 2013 9th international*. 2013. IEEE.
109. Burguera, I., U. Zurutuza, and S. Nadjm-Tehrani. *Crowdroid: behavior-based malware detection system for android*. in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. 2011. ACM.
110. Rastogi, V., Y. Chen, and W. Enck. *AppsPlayground: automatic security analysis of smartphone applications*. in *Proceedings of the third ACM conference on Data and application security and privacy*. 2013. ACM.
111. Yan, L.-K. and H. Yin. *DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis*. in *USENIX security symposium*. 2012.
112. Wu, W.-C. and S.-H. Hung. *DroidDolphin: a dynamic Android malware detection framework using big data and machine learning*. in *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*. 2014. ACM.
113. Alam, M. and S. Vuong. *Random Forest Classification for Android Malware*. in *Proceedings of IEEE International Conference on Internet of Things*. 2013.
114. Enck, W., et al., *TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones.* ACM Transactions on Computer Systems (TOCS), 2014. **32**(2): p. 5.
115. Desnos, A. and P. Lantz, *Droidbox: An android application sandbox for dynamic analysis*, 2011.

116. Pravin, M.N.P., *Vetdroid: Analysis using permission for vetting undesirable behaviours in android applications.* International Journal of Innovative and Emerging Research in Engineering, 2015. **2**(3): p. 131-136.

117. Livadas, C., et al. *Usilng machine learning technliques to identify botnet traffic*. in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. 2006. IEEE.

118. Narudin, F.A., et al., *Evaluation of machine learning classifiers for mobile malware detection.* Soft Computing, 2016. **20**(1): p. 343-357.

119. Dai, S., T. Wei, and W. Zou. *DroidLogger: Reveal suspicious behavior of Android applications via instrumentation*. in *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*. 2012. IEEE.

120. Abdelrahman, O.H., et al., *Mobile network anomaly detection and mitigation: The NEMESYS approach*, in *Information Sciences and Systems 2013*. 2013. Springer. p. 429-438.

121. Andrews, B., T. Oh, and W. Stackpole. *Android Malware Analysis Platform*. in *Proc. of 8th Annual Symposium on Information Assurance (ASIA'13)*. 2013.

122. Shabtai, A., et al., *Mobile malware detection through analysis of deviations in application network behavior.* Computers & Security, 2014. **43**: p. 1-18.

123. Portokalidis, G., et al. *Paranoid Android: versatile protection for smartphones*. in *Proceedings of the 26th Annual Computer Security Applications Conference*. 2010. ACM.

124. Ruan, H., et al. *Analyzing Android Application in Real-Time at Kernel Level*. in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. 2017. IEEE.

125. Zaman, M., et al. *Malware detection in Android by network traffic analysis*. in *Networking Systems and Security (NSysS), 2015 International Conference on*. 2015. IEEE.

126. Spreitzenbarth, M., et al., *Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques.* International Journal of Information Security, 2015. **14**(2): p. 141-153.

127. Altaher, A. and O.M. Barukab, *Intelligent Hybrid Approach for Android Malware Detection based on Permissions and API Calls.* INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 2017. **8**(6): p. 60-67.

128. Nadji, Y., J. Giffin, and P. Traynor. *Automated remote repair for mobile malware*. in *Proceedings of the 27th Annual Computer Security Applications Conference*. 2011. ACM.

129. Liu, Y., et al. *A hybrid malware detecting scheme for mobile Android applications*. in *Consumer Electronics (ICCE), 2016 IEEE International Conference on*. 2016. IEEE.

130. Mas' ud, M.Z., et al. *Analysis of features selection and machine learning classifier in android malware detection*. in *Information Science and Applications (ICISA), 2014 International Conference on*. 2014. IEEE.

131. GitHub. *androguard*. Available from: https://github.com/androguard/androguard.

132. GitHub. *honeynet/apkinspector*. Available from: https://github.com/honeynet/apkinspector.

133. *Droidbox:Dynamic analysis of Android apps*. Available from: https://github.com/pjlantz/droidbox.

134. SandDroid. *SandDroid - An automatic Android application analysis system*. Available from: http://sanddroid.xjtu.edu.cn/.

135. Amsterdam, V. *Tracedroid*

*Dynamic Android app analysis* Available from: http://tracedroid.few.vu.nl/.

136. Li, L., et al. *Iccta: Detecting inter-component privacy leaks in android apps*. in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. 2015. IEEE Press.

137. Bagheri, H., et al., *Automated dynamic enforcement of synthesized security policies in android.* George Mason University, Tech. Rep. GMU-CS-TR-2015-5, 2015.

138. Kandukuru, S. and R. Sharma. *Android malicious application detection using permission vector and network traffic analysis*. in *Convergence in Technology (I2CT), 2017 2nd International Conference for*. 2017. IEEE.

139. Afifi, F., et al., *DyHAP: dynamic hybrid ANFIS-PSO approach for predicting mobile malware.* PloS one, 2016. **11**(9): p. e0162627.

140. Hsiao, S.W., et al. *PasDroid: real-time security enhancement for Android*. in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on*. 2014. IEEE.

141. MOTION, G. 2018; Available from: https://www.genymotion.com/.

142. You, C.-Y., et al., *A Light-weight Method to Send and Receive SMS messages in an Emulator*, 2016, National Central University.

143. Hoff, J. *Island is a cool little sandbox environment for testing your apps*. March 3, 2017; Available from: https://androidcommunity.com/island-is-a-cool-little-sandbox-environment-for-testing-your-apps-20170303/.

144. Comparatives, A., *IT Security Survey 2018*, 2018.

145. Cai, H., et al., *Droidcat: Effective android malware detection and categorization via app-level profiling.* IEEE Transactions on Information Forensics and Security, 2019. **14**(6): p. 1455-1470.

146. Khanmohammadi, K. and A. Hamou-Lhadj. *HyDroid: A Hybrid Approach for Generating API Call Traces from Obfuscated Android Applications for Mobile Security*. in *Software Quality, Reliability and Security (QRS), 2017 IEEE International Conference on*. 2017. IEEE.

147. Titze, D., M. Lux, and J. Schuette. *Ordol: Obfuscation-Resilient Detection of Libraries in Android Applications*. in *Trustcom/BigDataSE/ICESS, 2017 IEEE*. 2017. IEEE.

148. Li, L., et al. *Droidra: Taming reflection to support whole-program analysis of android apps*. in *Proceedings of the 25th International Symposium on Software Testing and Analysis*. 2016. ACM.

149. Lee, H.S. and H.-W. Lee, *Dynamic Analysis System for Detecting Remote Server-Side Polymorphic Malicious Mobile Apps on Android based Smartphone.* International Journal of u-and e-Service, Science and Technology, 2015. **8**(11): p. 295-302.

150. Yuan, C., et al. *Scalable and Obfuscation-Resilient Android App Repackaging Detection Based on Behavior Birthmark*. in *Asia-Pacific Software Engineering Conference (APSEC), 2017 24th*. 2017. IEEE.

151. Ma, Z., et al. *LibRadar: fast and accurate detection of third-party libraries in Android apps*. in *Proceedings of the 38th International Conference on Software Engineering Companion*. 2016. ACM.

152. Ahmed, M.K., M.N.A..., *A Review of Forensic Analysis Techniques for Android Phones.* Journal of Independent Studies and Research 2017.

153. Li, J., D. Gu, and Y. Luo. *Android malware forensics: Reconstruction of malicious events*. in *2012 32nd International Conference on Distributed Computing Systems Workshops*. 2012. IEEE.

154. Kumar, A., K. Kuppusamy, and G. Aghila, *FAMOUS: Forensic Analysis of MObile devices Using Scoring of application permissions.* Future Generation Computer Systems, 2018. **83**: p. 158-172.

155. *The Drebin Dataset*. Available from: https://www.sec.cs.tu-bs.de/~danarp/drebin/.

156. *Contagio malware dump*. Available from: http://contagiodump.blogspot.com/.

157. Viennot, N., E. Garcia, and J. Nieh. *A measurement study of google play*. in *ACM SIGMETRICS Performance Evaluation Review*. 2014. ACM.

158. Lin, X., et al., *Automated forensic analysis of mobile applications on android devices.* Digital Investigation, 2018. **26**: p. S59-S66.

159. ; Available from: http://www.appchina.com/.

160. Chernyshev, M., et al., *Mobile forensics: Advances, challenges, and research opportunities.* IEEE Security & Privacy, 2017. **15**(6): p. 42-51.

161. Almarri, S. and P. Sant, *Optimised malware detection in digital forensics.* International Journal of Network Security & Its Applications, 2014. **6**(1): p. 1.

162. Alhassan, J., et al. *Comparative Evaluation of Mobile Forensic Tools*. in *International Conference on Information Theoretic Security*. 2018. Springer.

163. ESET. *CAN ARTIFICIAL*

*INTELLIGENCE*

*POWER FUTURE*

*MALWARE?* 2018; Available from: https://www.welivesecurity.com/wp-content/uploads/2018/08/Can_AI_Power_Future_Malware.pdf.

164. Feizollah, A., et al., *A study of machine learning classifiers for anomaly-based mobile botnet detection.* Malaysian Journal of Computer Science, 2013. **26**(4): p. 251-265.

165. UNB. *Android Botnet dataset*. Available from: http://www.unb.ca/cic/datasets/android-botnet.html.

166. Nix, R. and J. Zhang. *Classification of Android apps and malware using deep neural networks*. in *Neural Networks (IJCNN), 2017 International Joint Conference on*. 2017. IEEE.

167. McLaughlin, N., et al. *Deep android malware detection*. in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 2017. ACM.

168. Huang, T.H.-D. and H.-Y. Kao, *R2-D2: color-inspired convolutional neural network (cnn)-based android malware detections.* arXiv preprint arXiv:1705.04448, 2017.

169. Hasegawa, C. and H. Iyatomi. *One-dimensional convolutional neural networks for Android malware detection*. in *Signal Processing and Its Applications (CSPA), 2018 IEEE 14th International Colloquium on*. 2018. IEEE.

170. Karbab, E.B., et al., *MalDozer: Automatic framework for android malware detection using deep learning.* Digital Investigation, 2018. **24**: p. S48-S59.

**Attia Qamar** has done BS (IT) from Bahauddin Zakariya University, Pakistan and MS (IT) from National College of Business Administration and Economic, Pakistan. Her area of research includes, mobile forensics, machine learning and malware detection using CNN.

**Ahmad Karim** is working as Researcher at TM R&D Cyberjaya, Malaysia. He did his PhD from University of Malaya, Malaysia in the domain of Computer Security. Previously, he received his MS (CS) degree from Bahauddin Zakariya University Multan, Pakistan with distinction. He also holds Cisco International Certifications (CCNA, CCNP, CCAI). His area of research includes Mobile Botnet Detection, Computer Security, Computer Networks, Software Defined Networks, Big-Data Analytics, and IoT.

**Prof. Victor Chang** is a Senior Associate Professor, Director of Ph.D. (June 2016 – May 2018), Director of MRes and Interim Director of BSc IMIS at International Business School Suzhou (IBSS), Xi’an Jiaotong-Liverpool University (XJTLU), Suzhou, China. He joined XJTLU since June 2016. He is also a very active and contributing key member at Research Institute of Big Data Analytics (RIBDA), XJTLU. He's an Honorary Associate Professor at the University of Liverpool and Visiting Researcher at the University of Southampton, UK. Previously he worked as a Senior Lecturer at Leeds Beckett University, UK, for 3.5 years.

Within 4 years, he completed Ph.D. (CS, South ampton) and PGCert (Higher Education, Fellow, Greenwich) while working for several projects at the same time. Before becoming an academic, he has achieved 97% on average in 27 IT certifications. He won a European Award on Cloud Migration in 2011, IEEE Outstanding Service Award in 2015, best papers in 2012 and 2015, the 2016 European award: Best Project in Research, 2016 SEID Excellent Scholar, Suzhou, China, Outstanding Young Scientist award in 2017, 2017 special award on Data Science, 2017 and 2018 INSTICC Service Awards and numerous awards since 2012.

Dr. Victor Chang is a visiting scholar/Ph.D. examiner at several universities, an Editor-in-Chief of IJOCI & OJBD journals, Editor of FGCS, Associate Editor of TII, founding chair of two international workshops and founding Conference Chair of IoTBDS http://www.iotbd.org and COMPLEXIS http://www.complexis.org since Year 2015 and FEMIB femib.scitevents.org since Year 2018.

He was involved in different projects worth more than £12.5 million in Europe and Asia. He has published 3 books as sole authors and the editor of 2 books on Cloud Computing and related technologies. He gave 17 keynotes at international conferences. He is widely regarded as one of the most active and influential young scientist and expert in IoT/Data Science/Cloud/security/AI/IS, as he has experience to develop 10 different services for multiple disciplines. He is founding conference chair for IoTBDS, COMPLEXIS and FEMIB to build up and foster active research communities globally.

He is one of the most highly cited researchers at XJTLU apart from being an international leading young scientist. His most-recent publications can be viewed at: https://scholar.google.com/citations?hl=en&user=IqIYZ[illegible]4AAAAJ&view_op=list_works&sortby=pubdate

## MOBILE MALWARE ATTACKS: REVIEW, TAXONOMY & FUTURE DIRECTIONS

# (HIGHLIGHTS)

1. Android malware detection is one of the sizzling topic in cybersecurity domain.

2. This paper provides a comprehensive review on state-of-the-art mobile malware attacks, vulnerabilities, detection techniques and security solutions over the period of 2013-2018.

3. We presented two well-organized and in-depth taxonomies.

- First to uncover mobile malware detection approaches that spans under umbrella term of artificial intelligence techniques as machine and deep learning-based models, analysis approaches, working platform, data acquisition to various other tools and techniques covered by researchers in the last few years to detect such types of attacks.

- Second taxonomy covers mobile malware attack vector based on type of attacks, threat clusters, loopholes, and their malicious operational impact on wide scale. Besides, we explained different evasion techniques that are used prominently by the malware authors to hinder detection efforts.

4. Presented various security measures as future work that should be opted to cope this growing hazard.