# A Review on several vulnerabilities detection techniques in Android Mobile

**Ms. Jigna Solanky[1], Dr. Dharmendra Bhatti[2]**

*[1]Assistant Professor, Department of Computer Science, Uka Tarsadia University, Bardoli, Gujarat, India*
*[2]Professor, Department of Computer Science, Uka Tarsadia University, Bardoli, Gujarat, India*

---***---

**Abstract -** *Mobile Devices, tablets and Smartphone's have rapidly growing because of their extremely personal and powerful attributes. Android has been the most famous mobile operating system. As Android has governing most of the market, the problem of malware threats and security is also increasing. Android has continuously becoming the most targeted platform for attackers. Although there have been many number of studies reviewing the current analysis and detection methods, they are unable to fully address this research domain. Hence, in this review paper, we group the recent analysis and detection methods in mobile vulnerabilities detection. Addition to that, we review the Android features available in mobile vulnerabilities detection, and various trusted and widely used datasets.*

***Key Words*:** Vulnerabilities, Android, Mobile Vulnerabilities, Detection techniques, Permissions, static and dynamic analysis

## 1. INTRODUCTION

Now a day's smart-phones are becoming very popular all around the globe. Mobile devices have becoming important part of many of the people's life. As the study says, among all the mobile platforms, Android is the widely used platform. With the growing use of these mobile platforms in delicate applications, there is a problems linked with malicious activities targeted at mobile devices. Smart phones have replaced use of personal computers in terms of internet usage. Addition to that, smart phone allows users to check their emails, tweets or social media in device. In terms of smart phone usage, 50.3% of all web traffic came from mobile devices compared in 2017 and in 2018 it reaches to 52.2% which is increasing day by day [1].

A malicious activity has threatened smart phones for many years and android devices are gaining popularity with time. Seeing this most of the discovered vulnerabilities is aiming at android platform. A malicious activity could be any code which is added, removed or changed from an application in order to intentionally cause harm to the significant function of the system. The main purpose of attacker is to steal data, personal information, gaining access to user's accounts and establishing control channels. The functioning of a device also depends upon the type of vulnerability. Vulnerability writers are actively and continuously developing vulnerable programs to target Android platform. This continuous evolution and the diversity of vulnerability pose a major threat to Android applications.

Many users storing some private data such as contact list, passwords, and credit card numbers on mobile devices. Now a day's mobile banking is widely used among people since they are able to access their account's information on the go and saving the account's credential on the device is inevitable. Based on this scenario, attackers have turned their attention to smart phones, as sensitive data are available abundantly on smart phones, and the security issues are taken less seriously on such devices.

**Adware:** It generally aims to just advertising the products or websites that are annoying but doesn't cause any harm. Android dowgin is a adware which install itself on a mobile device as a bundle with the other applications. Later on it displays ads in the notification bar and cannot be easily removed. It is estimated that between 10000-50000 users are infected with this adware [2].

## 2. Features of Android in Mobile for malicious activity Detection

Researches use distinctive features available in Android for examination. Android applications contain various components, for example, permissions, Java code, certification, the behavior of the application on the gadget, and their behavior on the network. Choosing the most helpful subset of features from a huge number of accessible features changes the after effect of the entire experiments. We divide available features into two main groups' i.e. static and dynamic features.

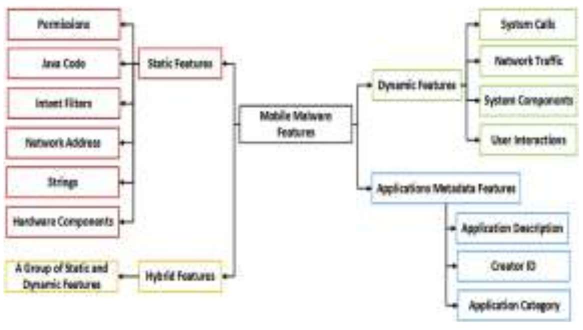

**Fig -1**: List of all Android Features

## 2.1 Static Features

Static feature includes features which are available in APK file such as java code files and AndroidManifest.xml file. Following are the list of static features.

**1. Android Permissions**

There are 130 official android permissions. Google categorizes them into four groups i.e. normal, dangerous, signature, signature or system. Researchers use various approaches in analyzing android permissions. Applications might manipulate the requested permissions and get access to secrete information without any consent of user. For example, applications which requests more permissions then they actually need, named as over privileged [17]. Later on, such applications can be transformed silently into malicious activities, whenever an application or operating system update occurs.

Droid Analytics [9], proposed solution for signature based analytic system to automatically collect, manages, analyze and extract android malware. They developed solution to scrutinize android applications at the byte code level and generated the appropriate signature which can be used by antivirus software. This tool is effective in analyzing malware repackaging and mutations.

APK Auditor [19], is permission-based android malware detection system. This system uses static analysis to characterize and classify android application as benign or malicious. This has three components such as Signature database, An Android Client and A Central server. This tool is able to detect most well known malwares only and as a result get 88% accuracy.

Permlyzer[20], analyzes applications' requested permission usage based on both static and dynamic analysis.

**2. Android Java Code**

Researcher uses various analysis approaches on java code. Several researchers use API calls to detect malware. Every android application must have API calls to interact with the device. API calls in android is sequential. Researchers believe such a sequence as a signature which is unique to that application. So, modifying the sequence of the API calls is a strategy called code obfuscation which is used by attackers to bypass the detection process. Attackers can change the sequence of API calls or rename the calls to avoid the detection system. Some researchers adopting another approach that is analyzing control flow of java code. Flow of java code does not change and researchers use it to develop stronger malware detection system.

## 2.2 Dynamic Features

Dynamic features are defined as behavior of applications in interaction with operating system or network connectivity. There are two main types of dynamic features:

**1. Android System Calls**

System call analysis is required for anomaly detection in android application's behavior. Android Applications use system calls to perform particular tasks such as read, write and open, because they cannot directly interact with the Android operating system. On issuing a system call in user mode, the Android operating system switches to kernel mode to perform the required task. System call is the most selected feature among the dynamic features, constituting more than half of the reviewed papers [17]. Research work [16] and [21], analyzes system calls to detect malicious applications.

**2. Network Traffic**

Every application needs resources and services from the operating system by issuing system calls, such as read, write and open. Network traffic is another dynamic feature used by researchers. Applications tend to connect to a network to send and receive data, receive updates, or maliciously leak personal data to attackers. Monitoring network traffic of mobile devices is a way of detecting a culprit in the act [2].

**3. Dynamic Tainting Data flow and control flow**

In paper [25], author used the approach of tracking the flow of privacy sensitive data through third-party applications.

**4. Power Consumption**

In paper [27], they have considered power consumption as the distinct feature between benign and malicious application.

In paper [6], author has considered other useful features such as Incoming and outgoing network information, loaded class through DexclassLoader, Broadcast Receivers, Activated Services, Permission Bypass, Enforced Permissions, Information leakage via the network, SMS sending, Phone calls and API calls to perform encrypted activities.

## 3. Mobile malicious activities Analysis

This section is committed to talking about analysis methods. Malware analysis is the process of examining a sample of a malicious application or a malware family in order to detect a pattern and attribute. Such attributes are then used for detection techniques. There are three types of Android malicious activity analysis: static, dynamic, and hybrid. For each analysis type, we need to examine the current research works, and point out their strengths as well as their weaknesses.

## 3.1 Static Analysis

Static analysis can be used to examine malicious activity with low computational overheads without doing malicious activity itself. It analyzes android files without installing them on a device or emulator. In static analysis, static features are used. Static analysis can be directly employed either on the source code of the application or the binary file and use reverse engineering techniques. Researchers mostly used permission feature rather than other static features. Selection of java code comes second. Malicious Applications generally request more permission then they need, which is a way to detect them from normal applications. Most of researchers are focuses on static analysis. Generally researchers extracted different information from two parts of an application i.e. AndroidManifest.xml and classes.dex [3]. Several researchers used API calls to detect malware. Control flow is another approach that is used by researchers, which is sequence of java code statements. Control flow approach is better than API calls [17], because attackers can modify the sequence of API calls to evade detection system. The flow of java code does not change and researcher uses them for detection system. Static analysis is stable and do not affect the running applications which are the major benefits of it.

There are certain problems with static analysis.

1. Dynamically loaded code is undetectable.

2. Permission based analysis is less effective [17] because of Basebridge malware which hides an updated version within the original application and as a result slips into a mobile device without user's knowledge and bypass permission system which can't be detected by static analysis.

3. Code obfuscation can't be detected by static analysis

4. Java reflection is used by attacker to evade detection which also can't be detected by static analysis.

5. Control flow obfuscation can't be detected by static analysis.

6. Static analysis can easy to be avoided through transformation techniques such as junk instruction insertion, code reordering etc [23].

7. It cannot detect real time privacy abuse behaviors.

These limitations can be solved by dynamic analysis.

### 3.2 Dynamic Analysis

The applications code is executed in virtual environment such as emulator and its interaction with the system is investigated. In device examine the behavior and react to malicious behavior accordingly. Dynamic analysis is useful for detecting the transformed malware variants because most of dynamic features are preserved. Many malware variants share common behaviors that can be selected through dynamic analysis. The strength of dynamic analysis is that it can detect real-time behavior of applications and it is accurate compared to static analysis. Drawback of this approach is that during the execution of the code a malicious path may be ignored [3]. Cost of deployment of dynamic analysis is high compared to static analysis and hard to automate the dynamic analysis as manual efforts are also required. Dynamic analysis may affect the running applications which can slow down the device performance.

Another method used by attackers to avoid detection is Java Reflection. It is defined as changing or examining the run time behavior of a class. Reflection for Android applications can also be used to get access of all API library's hidden and private classes, methods, and fields. Android malware such as Android.Obad and FakeInstaller call their methods indirectly through reflection, and they kept real method name encrypted. Moreover, the name of the target method is unknown prior to execution of the applications. Hence, by converting any method call to a reflective call with the same function, it becomes difficult for static analysis to discover exactly which method was called. Overall, static analysis has shortcomings such as code obfuscation and control flow obfuscation that can be solved by using dynamic analysis [17].

Several researchers have developed a cloud-based system that receives gathered applications activity log from smart phone and sends them to the remote server and perform analysis on that for the detection of malware but this approach is not effective since any modification on the device must be reported to the cloud and which results in large bandwidth consumption and utilization of lot of battery power [17]. Still in this research field there is an open issue is to develop effective and efficient Smartphone malware detection system to face monumental growth in mobile malware.

Dynamic analysis can solve the problems of static analysis but it has code coverage problem (While application is running there is no gurantee that the execution path in java code induce and triggers malicious behavior of malware is defined as code coverage). In dynamic analysis overheads are relatively high and kernel modification might be needed to extract dynamic features [23].

Author in [24], developed system wide dynamic taint tracking for android. Taint tracking marks any ambiguous data that originate from sensitive sources such as camera, location, microphone and phone identifiers. They have used dynamic analysis method to monitor and gain any sensitive data before they are forwarded to the network interface of a mobile phone, so this can avoid data leakages. It only focuses on data flows so cannot able to detect other vulnerabilities like network intrusions. System suffers from false positive and false negative problem.

### 3.3 Hybrid Analysis

Hybrid analysis is the optimum approach because it uses both static and dynamic analysis. In this approach combination of both static and dynamic analysis is used so that their added strengths can overcome weaknesses of both the approach while using individually.

For Example, [24] proposed a hybrid approach for mobile malware detection in android using both static and dynamic analysis. They have used static analysis method based on characteristic tree for finding API usage. They have used machine learning approach based on Bayesian classification for discovering unknown android malware via static analysis. They applied dynamic analysis in taint tracking and system call tracking. So, this approach collects application behavior data in a dynamic way and processing the data is static way and hence achieved accuracy rate as 90%. This approach is not able to detect runtime malware detection.

In [25], author has proposed DroidRanger that uses hybrid analysis. They have used static analysis to extract permissions, and matches applications' permission-based footprint with malware-specific footprint signatures. The researchers also proposed heuristics-based filtering schemes that scrutinize applications for suspicious behavior such as dynamically loaded code. The suspicious applications are detected using dynamic analysis to confirm whether they are malicious or not. In case malware is detected, the DroidRanger generates its signature and adds it to the database. The authors evaluated DroidRanger by 46 downloading applications from five different Android markets, and used their system to detect malware. The results show that they detected 171 malicious applications and two zero-day malware.

## 4. Malware Detection

This section focuses on various types of detection methods. After examining malware families, their characteristics and behavior are used for detection purpose. There are signature-based detection and behavior-based detection. For each detection type, we look into several related research works and analyze them in terms of their weaknesses and strengths.

### 4.1 Signature-based (Misuse) detection

Signature based technique is only capable of detecting attacks for which they are programmed to alert. Misuse detection is a signature-based approach which detects malware by the set of rules or policies. The strength of this approach is to precisely detect the android malware if any of the signatures is matched. This approach is utilized by the antivirus programming which depends on discovering malwares in view of their one of a kind mark. Despite of the fact that it is highly exact but it is useless against new type of malware and it requires consistent updation of signature. Misuse based IDS is used for known malwares. When new threat arises, same process has to be performed and the generated signature has to be added to the database.

### 4.2 Anomaly-based (behavior) detection

Anomaly detection is different than misuse detection because it generally applies machine learning algorithms for learning known malware behavior and predicting unknown malware. This technique is able to detect unknown malware but it sometimes causes high false positive. This approach relies upon algorithms keeping in mind the end goal is to instruct or teach framework to separate amongst benign and malicious applications.

This approach uses prior training phase to establish a normality model for the system activity. In this type of detection method, detection system is first trained on the normal behavior of the application to be monitored. By use of normality model of behavior, it becomes possible to detect anomalous activities by looking for malicious behavior or an activity that varies from the normal behavior earlier defined occurring in the system. This approach has strength of being able to detect new and unknown malware attacks.

Behavior-based detection need to use the feature vector for training the classifier before subsequent classifier can be carried out. These feature vectors are received from the features which are collected from the system. Usually researchers are using machine learning approach specifically with supervised learning for anomaly detection. Machine learning model is then train using a labeled data received from the understanding of application behaviors. Then trained classifiers are used to detect future outcomes of the test feature vectors [22].

Since behavior-based detection method enables us to detect malware based on their behavior, we choose it for our work. This way, our method is able to detect new malware with the same behavior, as opposed to misuse-based detection where we have to manually update the malware signature every time whenever a new malware is detected.

Anomaly detection can be classified in to three broad categories [28]:

1. Knowledge-based

- It tries to capture the claimed behavior.

2. Statistical-based

- In statistical approach, behavior of the system is represented from a random view point.

3. Machine Learning based

- In this approach, system will learn and recognize complex patterns automatically and made intelligent decisions based on the data.

## 5. DATASETS

Each research work needs a dataset in view of which the creators assess their proposed framework. Various specialists endeavored to collect tests through a few sites that common Android malware samples, for example, Contagio. Therefore, the shortcoming was the restriction of malware samples that made the assessment of their framework questionable. This section analyzes points of interest of the most generally utilized Android malware information tests. Some datasets of malware samples are available publicly.

### 5.1 MalGenome

The MalGenome data sample contains 1,260 Android malwares from 49 various families and this dataset is created by the author of [31]. A malware family is a collection of malware demonstrating similar a behavior. This collection was collected between August 2010 and October 2011 by the North Carolina State University. The authors examined the data samples and found that around one third (36.7%) of the gathered malware samples leverage root-level exploits to fully compromise Android security, posing the highest level of threats to users' security and privacy. Addition to that more than 90% of malware turn the compromised devices into a botnet controlled through network or short messages. Among the 49 malware families, 28 (with 571 or 45.3% samples) of them have the built-in support of sending out messages to premium-rate numbers or making phone calls without user awareness. However, this dataset is old and many of the samples are not generating traffic any more [30].

### 5.2 AndroZoo

AndroZoo is an arising collection of Android applications from a few sources, including the official Google Play. AndroZoo contains more than 5 million Android applications. It contains both android malware samples and benign applications too [32]. Analyzing different sources began in late 2011 and has proceeded with from that point onward.

### 5.3 Drebin

They change shape and infecting technique to evade detection based on the malware nature. The Drebin data sample was published in 2014 [12]. This database is a collection of 5,560 Android malware categorized into 179 different families. It was gathered between August 2010 and October 2012. The authors scanned the Drebin with antivirus applications. They report that while the best scanners identified over 90% of the malware, others identified less than 10% of the data sample.

## 6. Literature Survey

Study of different malware detection techniques and approaches are done. Different malware detection systems are proposed earlier. Authors in [3], presents a malware classification approach with true detection accuracy and evaluates the approach using artificially generated examples. This approach generates the behavior and signature profiles of each application in the data set, which is then used as input for classification task. For improving detection accuracy they have used feature fusions of features from various filter and wrapper methods are used. They have used different machine learning algorithms for classification such as J48, JRip, SMO, Naïve Bayes and IBK. They have obtained AUC and F1 up to 0.94 for both known and unknown malwares. In paper [4], author proposed a permission-based malware detection framework which uses machine learning algorithms to detect malicious applications. For improving the efficiency of permission-based malware detection framework, they introduced new method namely TF-IDF (Term Frequency – Inverse Document Frequency) to assign weight to each feature extracted from an APK file. As a result, they achieved detection rate as 95.3% using different classifier algorithms like J48, Naïve Bayes, SVM and KNN. Limitation of this system is that it's working on single feature that is permission.

Author has used static approach for analyzing the manifest file of android application for malware detection and characterization. They have extracted permission-based features by disassembling the manifest file of android application. They have used score-based feature selection method. For classification, different algorithms are used namely BayesNet(BN), Naive Bayes, Multilayer Perceptron (MLP), K-Nearest Neighbor, J48 and Random Forest. According to experimental result, Bayes network is less efficient compared to other classifier for malware detection. They have achieved detection rate as 87%. Limitation of this approach is that they have considered only one feature that is only permission and considering only permission would have difficulties to improve the current detection accuracy [5]. In paper [6], they have used static feature-based mechanism to provide a static analyst paradigm for detecting android malware. They have used static information's such as permissions, deployment of components, Intent message passing and API calls for characterizing the android behavior. They have used KNN algorithm to classify application as benign or malicious apps. They have compared their analysis result with most famous tool, AndroGuard and improve their recall rate. They have also improved analysis timing for detecting benign or malicious. They consider different parameter for evaluation of model such as Accuracy – 0.97, Recall -0.87, Precision – 0.96 and F-Measure – 0.91.Author presents an intrusion detection system for detecting anomaly behaviors in android mobile devices. The Intrusion detection system continuously monitors the network traffic of the mobile device and collects various features on NetFlows. Artificial Neural Network collects data flows and determines whether there is an intrusion or not. They have achieved accuracy as 85% and detection rate as 81.56% [7]. Author in [8], has proposed permission-based android malware detection system to enhance security and privacy of Smartphone users. The proposed system monitors different permission-based features and events obtained from the android applications and analyze these features using machine learning algorithms for classifying whether the app is benign or malicious.

In paper [9], author has proposed a framework that considers both requested and used permissions in the android applications. They have used two layered malware detections scheme and uses machine learning techniques to get high detection accuracy with detection of android malware applications based on permissions. Author in [10], proposed a framework for malware detection system that analyzes the application activity of the mobile phone. For selecting features they have used principal component analysis method and applies support vector machine to train and build classifier for malware detection.

Author has proposed client-server architecture for malware detection system based on permissions checker parts. At Client-side part, system extracts the permissions from android applications and forwards it to the server-side part. Server-side part classifies the application as benign or malware [11]. In paper [12], author proposed a light weight android malware detection approach that enables detecting malware applications in the phone directly. This approach performs broad static analysis by collecting all possible features of android APK. This approach gives advantages of time efficiency in analyzing unknown apps. This system used directly in mobile phone and enables protection of installation from untrusted sources. Author in [13], proposed machine learning based framework for malicious applications detection and security enhancement of android based smart phones. This framework monitors the different permission based features extracted from android applications and uses machine learning classifiers to tag the application as benign or malicious. They have used Information Gain method for feature selection.

**Table -1:** List of features and methods used for detection of malicious activities and their evaluation results

| **Authors** | **Features Selected** | **Detection Technique** | **Classifiers** | **Evaluation Criteria** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | **Confusion Matrix** | **Recall** | **Accuracy** | **AUC** | **Precision** | **True Positive Rate** | **F-Measure** |
| Raja Khurram Shahzad [3] | Permissions, Intents, Listed hardware components from listed files | Signature and Behavior based profile | J48, JRip, SVM, Naïve Bayes, Nearest Neighbour | yes | 0.940 | - | 0.938 | 0.947 | 0.940 | 0.937 |
| Abdirashid Ahmed Sahal, Shahid Alam and Ibrahim Sogukpinar [4] | Permission | - | SVM, J48, Naïve Bayes, KNN | - | - | 95.3% | - | - | - | - |
| Chit La Paye Myo Hein, Khin Mar Myo [5] | Permission | Anomaly | BayesNet, Naïve Bayes, MLP, K-Nearest Neighbor, J48, Random Forest | Yes | - | BN-80%, MLP-95%, J48- 87%, RF-92%, KNN – 87% | - | BN-0.78, MLP-1, J48 – 0.89, KNN – 1, RF-0.94 | BN-0.82, MLP-0.9, J48 – 0.84, KNN – 0.92, RF-0.96 | BN-0.80, MLP-0.94, J48 – 0.86, KNN – 0.95, RF-0.95 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee and Kuo-Ping Wu [6] | permissions, deployment of components, Intent messages passing and API calls | Anomaly | KNN | Yes | 0.87 % | 0.97% | - | 0.96% | - | 0.91 |
| Panagiotis I. Radoglou-Grammatikis, Panagiotis G. Sarigiannidis [7] | Network traffic | Anomaly | - | yes | - | 85% | - | - | - | - |
| Zarni Aung and Win Zaw [8] | Permission and Events | Anomaly | J48, Random Forest, CART | Yes | J48-0.88 %, Random Forest-0.91 %, CART-0.85 % | J48-88%, Random Forest-91%, CART-85% | - | J48 - 0.88%, Random Forest - 0.91%, CART-0.85% | - | - |
| X. Liu and J. Liu [9] | Permission | Signatured-based | - | yes | - | 0.98% | - | 0.89% | 0.80% | - |
| Z. Xiaoyan, F. Juan, and W. Xiujuan [10] | Permission | Anomaly | SVM | Yes | - | 90.08% | - | - | - | - |
| P. Rovelli and Ý. Vigfússon [11] | Permission | Anomaly | C4.5, K*, RIPPER, Naïve Bayes | yes | - | C4.5-94.78%, K*-95.46%, RIPPER-95.02%,Naïve Bayes – 88.31% | - | - | C4.5-92.41%, K*-92.28%, RIPPER-92.28%,Naïve Bayes – 79.66% | - |
| D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens [12] | Permission, API calls, Network Addresses, Intent Filters | Anomaly-based | SVM | - | - | 93.90% | - | - | - | - |
| Jaemin Jung, Hyunjin | API calls | Anomaly-based | Random | - | - | 99.98% | - | - | - | - |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Kim, Dongjin Shin, Myeonggeon Lee, Hyunjae Lee, Seong-je Cho, Kyongwon Suh [13] | | | Forest | | | | | | | |
| M. Zhao, F. Ge, T. Zhang, and Z. Yuan [16] | Send SMS, Access GPS, Exec Shell | Signature based | SVM | - | - | 90% | - | - | - | - |
| Min Zheng, Mingshen Sun, John C.S. Lui [18] | Permission, System Component | Signature based | - | - | - | - | | - | - | - |
| Kabakus Abdullah Talha, Dogru Ibrahim Alper, Cetin Aydin [19] | Permission | Signature-based | | | | | | | | |
| W. Xu, F. Zhang, and S. Zhu [20] | Permission | Signature-based | - | - | - | - | - | - | - | - |
| I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani [21] | Behavior related data | Anomaly | - | - | - | - | - | - | - | - |
| Joshua Abah, Waziri O. V, Abdullahi M. B, Arthur U. M and Adewale O.S [22] | In/Out SMSs, IN/Out Calls, Device Status, Running Applications/Processes | Anomaly | KNN | yes | 0.84 % | 93.75% | 0.99 % | 1.0% | 0.84% | - |
| TaeGuen Kim, BooJoong Kang, Eul Gym Im [23] | API calls | Anomaly-based | - | - | - | 0.96% | - | 0.96% | - | - |
| Fei Tong, Zheng Yan [24] | System Calls | Anomaly | - | - | - | 90% | - | - | - | - |
| Y. Zhou, Z. Wang, W. Zhou, and X. | Permission | Anomaly-based | - | - | - | - | - | - | - | - |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Jiang [26] | | | | | | | | | | |
| B. Dixon, Y. Jiang, A. Jaiantilal, and S. Mishra [27] | Power consumption | Anomaly | - | - | - | - | - | - | - | - |
| Sanjay Kumar, Ari Vinnikainen and Timo Hamalainen [30] | Network Traffic | Anomaly | Random Forest, PART | yes | - | 97.5% | - | 0.94% | 0.98% | 0.99% |
| P.K.Chan, Wen-Kai song [33] | Permission, API Calls | Anomaly | Naïve Bayes, SVM, SMO, MLP, Random Forest | yes | - | Naïve Bayes- 86.33%, SVM with SMO – 90.10%, MLP – 91.31%, Random Forest – 92.36% | - | - | Naïve Bayes- 94.41%, SVM with SMO – 95.91%, MLP – 96.88%, Random Forest – 98.91% | - |
| Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo _Alvarez [34] | Permissions | Anomaly | Naïve Bayes, SMO, J48, Random Forest | Yes | - | Naïve Bayes- 67.64%, SMO- 82.84%, J48- 81.32%, Random Forest- 85.82% | - | - | Naïve Bayes- 0.50%, SMO- 0.91%, J48- 0.87%, Random Forest- 0.91% | - |

Author in [14], proposed a behavioral detection framework instead of signature-based solutions. They detected mobile malware by observing the logical ordering of an application's actions. They discriminate malicious behavior of malware from normal behavior of applications by training a classifier based on support vector machine (SVM). In paper [15], author has proposed system to detect anomalous behavior on mobile devices based on abnormal power consumption by malware. Author in [16], focuses on the software behavior based malware detection framework called AntiMalDroid by using SVM algorithm. The proposed framework dynamically extends malware characteristics into the database. In paper [22], author has used machine learning approach for the detection of malware on android platform. The proposed detection system monitors and extracts features from the android applications while in execution and use that features to perform in-device detection using K-Nearest Neighbour classifier. They have achieved 93.75% accuracy and low error rate of 6.25% with ability of detecting new android malware.

**Table -2:** List of all the reviewed papers

| No. | Reference | Feature Type | Year | No. of tested App |
|---|---|---|---|---|
| 1 | Raja Khurram Shahzad [3] | static | 2018 | 7000 |
| 2 | Abdirashid Ahmed Sahal, Shahid Alam and Ibrahim Sogukpinar [4] | static | 2018 | 1000 |
| 3 | Chit La Paye Myo Hein, Khin Mar Myo [5] | static | 2018 | 1981 |

| | | | | |
|---|---|---|---|---|
| 4 | Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee and Kuo-Ping Wu[6] | Static | 2012 | 1738 |
| 5 | Panagiotis I. Radoglou-Grammatikis, Panagiotis G. Sarigiannidis [7] | Dynamic | 2017 | 260456 |
| 6 | Zarni Aung and Win Zaw [8] | Static | 2013 | 700 |
| 7 | X. Liu and J. Liu [9] | Static | 2014 | 28548 |
| 8 | Z. Xiaoyan, F. Juan, and W. Xiujuan [10] | Static | 2014 | 454 |
| 9 | P. Rovelli and Ý. Vigfússon [11] | Static | 2014 | 2950 |
| 10 | D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens [12] | Static | 2014 | 129013 |
| 11 | M. Zhao, F. Ge, T. Zhang, and Z. Yuan [16] | Dynamic | 2011 | - |
| 12 | Min Zheng, Mingshen Sun, John C.S. Lui [18] | Static | 2013 | 150,368 |
| 13 | Kabakus Abdullah Talha, Dogru Ibrahim Alper, Cetin Aydin [19] | Static | 2015 | 8762 |
| 14 | W. Xu, F. Zhang, and S. Zhu [20] | Static | 2013 | 110,000 |
| 15 | I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani [21] | Dynamic | 2011 | - |
| 16 | Joshua Abah, Waziri O. V, Abdullahi M. B, Arthur U. M and Adewale O.S [22] | Dynamic | 2015 | - |
| 17 | TaeGuen Kim, BooJoong Kang, Eul Gym Im [23] | dynamic | 2018 | 2293 |
| 18 | Fei Tong, Zheng Yan [24] | Static and Dynamic | 2017 | - |
| 19 | Y. Zhou, Z. Wang, W. Zhou, and X. Jiang [26] | Static | 2012 | 204, 040 |
| 20 | B. Dixon, Y. Jiang, A. Jaiantilal, and S. Mishra [27] | static | 2011 | - |
| 21 | Sanjay Kumar, Ari Vinnikainen and Timo Hamalainen [30] | Dynamic | 2016 | 600 |
| 22 | Jaemin Jung, Hyunjin Kim, Dongjin Shin, Myeonggeon Lee, Hyunjae Lee, Seong-je Cho, Kyongwon Suh [13] | Static | 2018 | 60,243 |
| 23 | P.K.Chan, Wen-Kai song [33] | Static | 2014 | 796 |
| 24 | Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo _Alvarez [34] | Static | 2012 | 1811 |

## 7. CONCLUSION

In this review paper, we surveys and groups recent research works from the perspective of feature selection, mobile malware analysis and detection. We categorized features into three groups. The first groups comprised of static features that are related to apk file itself before installation into device. The second group comprised of dynamic features that are related to the application behavior after installation. Third group comprised of combination of both static and dynamic features. We categorized detection technique into two types. First technique is based on signature-based which are mostly used to detect known malwares. Second technique is based on behavior-based which are mostly used to detect malware based on behavior and widely used by researchers for detecting new or unknown malwares. It also reviews related datasets that are well-accepted and available for researchers and reviews evaluation measures. In table 2, we have listed all the papers to have glance view of recent work.

## REFERENCES

[1] Statista. (2018). Share of mobile phone website traffic worldwide 2018 Available: https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/
[2] Aashima Malhotra, Karan Bajaj, "A Survey on Various Malware Detection Techniques on Mobile Platform", International Journal of Computer Applications (0975-8887), Volume 139 – No.5, April 2016
[3] Raja Khurram Shahzad, "Android Malware Detection using Feature Fusion and Artificial Data", 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computer on pervasive Intelligence and Computer.
[4] Abdirashid Ahmed Sahal, Shahid Alam and Ibrahim Sogukpinar, "Mining and Detection of Android Malware Based on Permissions", IEEE, 3rd International Conference on computer science and Engineering, 2018
[5] Chit La Paye Myo Hein, Khin Mar Myo, "Permission-based Feature Selection for Android Malware Detection and Analysis", International Journal of Computer Applications, Volume 181 – No. 19, September 2018
[6] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee and Kuo-Ping Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing", 2012 Seventh Asia Joint Conference on Information Security
[7] Panagiotis I. Radoglou-Grammatikis, Panagiotis G. Sarigiannidis, "Flow Anomaly based Intrusion Detection System for Android Mobile Devices", In 6th International conference on Modern Circuits and systems Technologies, May 2017.
[8] Zarni Aung and Win Zaw, "Permission-Based Android Malware Detection", International Journal of Scientific and Technology Research Volume 2, Issue 3, March 2013
[9] X. Liu and J. Liu, "A two-layered permission-based android malware detection scheme," in Mobile cloud computing, services,and engineering (mobilecloud), 2014 2nd ieee international conference on. IEEE, 2014, pp. 142–148.
[10] Z. Xiaoyan, F. Juan, and W. Xiujuan, "Android malware detection based on permissions," The Institute of Engineering and Technology, 2014
[11] P. Rovelli and Ý. Vigfússon, "Pmds: Permission-based malware detection system," in International Conference on Information Systems Security. Springer, 2014, pp. 338–357.
[12] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in Ndss, vol. 14, 2014, pp.23–26.
[13] Jaemin Jung, Hyunjin Kim, Dongjin Shin, Myeonggeon Lee, Hyunjae Lee, Seong-je Cho, Kyongwon Suh, "Android Malware Detection based on Useful API calls and Machine Learning", 2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering
[14]A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral detection of malware on mobile handsets." In Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys '08, pages 225–238, New York, NY, USA, 2008. ACM.
[15]L. Liu, G. Yan, X. Zhang, and S. Chen. Virusmeter: Preventing your cellphone from spies. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pages 244–264, Berlin, Heidelberg, 2009. Springer-Verlag.
[16]M. Zhao, F. Ge, T. Zhang, and Z. Yuan. "Antimaldroid: An efficient SVM based malware detection framework for android." In C. Liu, J. Chang, and A. Yang, editors, ICICA (1), volume 243 of Communications in Computer and Information Science, pages 158–166. Springer, 2011.
[17] Hamidreza Alimardani, Mohammed Nazeh, "A Taxonomy on Recent Mobile Malware: Features, Analysis Methods and Detection Techniques", ICEMC'18 proceedings of the 2018 International conference on E-business and Mobile Commerce, Pages 44-49, china
[18] M. Zheng, M. Sun, and J. Lui, "DroidAnalytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware," in 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia, 2013, pp. 163 - 171.
[19] Kabakus Abdullah Talha, Dogru Ibrahim Alper, Cetin Aydin, "APK Auditor: Permission-based Android malware Detection System", Digital Investigation, 2015 – Elsevier
[20] W. Xu, F. Zhang, and S. Zhu, "Permlyzer: Analyzing permission usage in android applications," in Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on, 2013, pp. 400-410.
[21] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," in 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, Chicago, Illinois, USA, 2011, pp. 15-26.
[22] Joshua Abah, Waziri O. V, Abdullahi M. B, Arthur U. M and Adewale O.S, "A machine learning approach to anomaly-based detection on android platforms", International journal of Network Security and Its Applications Vol.7, No.6, November 2015
[23] TaeGuen Kim, BooJoong Kang, Eul Gym Im,"Runtime Detection Frameork for Android Malware", Hindawi, Mobile Information Systems, Volume 2018, Article ID 8094314, March 2018
[24] Fei Tong, Zheng Yan, "A hybrid approach of mobile malware detection in android", Journal of Parallel and Distributed computing, 2017 – Elsevier
[25] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. Chun, L.P. Cox, J. Jung, P. McDaniel, A.N. Sheth, "TaintDroid: an information-flow tracking system for real-time privacy monitoring on smartphones", ACM Trans. Comput. Syst. 32 (2) (2014) article No. 5.
[26] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets," in 19th Annual Network and Distributed System Security Symposium (NDSS), San Diego, USA, 2012, pp. 5-8.

[27] B. Dixon, Y. Jiang, A. Jaiantilal, and S. Mishra. "Location based power analysis to detect malicious code in smartphones." In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, SPSM '11, pages 27–32, New York, NY, USA, 2011. ACM.
[28] Dimitrios Damopoulos, "Anomaly-based intrusion detection and prevention systems for Mobile Devices: Design and Development", Thesis, 2013
[29] Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, Ainnuddin Wahid Abdul Wahab, "A review on feature selection in mobile malware detection",Digital Investigation 13(2015), Elesvier.
[30] Sanjay Kumar, Ari Vinnikainen and Timo Hamalainen, "Machine Learning classification model for network based intrusion detection system", The 11th International conference on Internet Technology and Secured Transactions (ICITST-2016)
[31] Z. Yajin and J. Xuxian, "Dissecting Android Malware: Characterization and Evolution," in 2012 IEEE Symposium on Security and Privacy (S&P), San Fransico, USA, 2012, pp. 95-109.
[32] K. Allix, T. F. Bissyand, J. Klein, and Y. L. Traon, "AndroZoo: collecting millions of Android apps for the research community," in 13th International Conference on Mining Software Repositories, Austin, USA, 2016, pp. 468-471.
[33] Patrick P.K.Chan, Wen-Kai song, "Static Detection of Android Malware by using permissions and API calls", Proceedings of the 2014 International Conference on Machine Learning and Cybernetics, Lanzhou, 13-16 July 2014.
[34] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo _Alvarez, "PUMA: Permission Usage to detect Malware in Android", International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions, springer