Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Anomaly based Intrusion Detection in Android Mobiles: A Review

Sapna Malik
Computer Science & Engineering Department
Maharaja Surajmal Institute of Technology
Delhi, India

***Abstract*—Android Operating System is most popular operating system since its launch for the mobiles users, developers and also for Intruders. The Intrusion detection in mobiles phones is done with two methods Signature-Based and Anomaly Based Detection. The anomaly based intrusion detection technique is more promising techniques for intrusion detection in mobiles phones as it can detect the new intruders more efficiently by analyzing their behavior. This review paper discussed the various researches in Anomaly based Intrusion Detection in which the static, dynamic and hybrid features of android applications are analyzed for intrusion detection in android Mobiles.**

Keywords—Anomaly Intrusion Detection; Static Features Analysis; Dynamic features analysis; Android Permissions;Malware Families;Machine Learning Algorithms;

I. INTRODUCTION

Being an open-source operating system, the Android Operating System is more vulnerable to attacks. The pervasion of malicious applications is rampant in the Android market and it is on the rise like never before. The Android applications are downloaded by the user from the official market such as Google Play store and also from the unofficial markets. The unofficial market is full of malicious applications which lure the customers to download their apps with a heavy discount on products or payback offers. The Android Operating system is having average severity of vulnerabilities of 7.3 from 2009-2018 [1]. Even the Official Market like Google Play store has not been left untouched by the intruder. In feb ,2019 ,29 photo app are found vulnerable on Google play store which are downloaded by more than 4 million user before their removal [2].These intruders tries to steal sensitive personal data like banking detail, location detail & phone details .Two android app was found on Google infected from the Anubis Trojan which had attacked 377 different bank applications from 93 countries all over the globe, with banks like Santander, RBS, Natwest, and Citibank, as well as non-banking apps such as Amazon, eBay, and PayPal [3].

Intruders are making malicious Android applications which are difficult to detect and prevent attack because of limitations of Android Operating System over other mobile Operating Systems. Moreover, Android Operating System in different devices is available from multiple manufacturers and it requires regular updates from all the partner manufacturers in a timely manner. According to a recent report from Duo Security, only 17% of Android devices have latest security patches. The majority of Android Devices haven't applied the latest security patches whereas 68% of Android devices are eligible to receive monthly security updates [4].There are other different factors such as different CPU architectures, limited memory resources, and energy constraint etc. which make the malware detection a complex task. The Intrusion Detection System (IDS) involves broadly two methods: Signature-Based and Anomaly Based Detection. Intrusion Detection System's performance is measured in terms of Accuracy and False positive Alarm. Signature Based IDSs are the less successful solutions for mobile malware detection as it requires constant updating of signatures database and they are unable to identify the new attack. However, Anomaly Based IDSs are more suitable for detection of rapidly changing nature of malware families as it can identify the unknown attacks. Thus, the accuracy of Anomaly Intrusion Detection System for unknown malware families is more than Signature based Intrusion Detection System but have a more false positive alarm for known malware families than Signature-Based Intrusion detection Systems. Moreover, Anomaly Intrusion Detection System takes more time for intrusion detection. Therefore more accuracy and less detection time is an open research area for Intrusion Detection System in Android Mobile Phones.

The Anomaly based Intrusion Detection System makes use of static feature analysis, dynamic feature analysis and hybrid feature analysis techniques to analyze and understand the intention of a malware.

The static analysis technique is based on the reverse engineering of the .apk (Android application package) file of Android applications. It includes the exploration of AndroidManifest.xml and classes.dex files for malicious codes without installing and executing the app. In static feature analysis technique, various Android features like Android Permission Requested, Used Features, Method call, API call sequence etc. are extracted and analyzed to detect the malicious application.

It examines the entire code for detecting malicious intention of the application. However, it is unable to detect the malicious code which can be downloaded by the malicious application at the run time. Thus, Static Feature Analysis technique cannot reveal the true nature of the malicious application.

The Dynamic Analysis technique scrutinizes the behavior of the Android application during its execution by extracting and analyzing the dynamic features like process lists, system call traces, symbol table, a list of open files and network traffic. Dynamic Analysis techniques reveal the potential harms that can be caused by the malware, but as this type of

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

analysis is done in the virtualized environment, it does not have 100% code coverage.

The Hybrid Features Analysis technique combines the advantage of Static and Dynamic Features Analysis techniques, although this technique is still in its infancy as a handful of studies are available in this area. Thus, intrusion detection with hybrid feature analysis is an open area of research.

In Anomaly based Intrusion Detection System, either Static Features or Dynamic features or both are analyzed to detect the intrusion. The following section explores the Intrusion Detection Systems proposed with static, dynamic or hybrid features analysis by various researchers so far.

II. INTRUSION DETECTION SYSTEM USING STATIC FEATURES ANALYSIS

Schmidt et al. [5] proposed Intrusion Detection System for an Android device which observes the system activities through process list, a list of open files, network traffic, symbol table and system call traces of the complete system to find out any abnormality in system behavior. The kernel level malware detection had been done through static function call analysis. This analysis was done on function calls used by 105 Executable and Linking Format (ELF) Executable installed on Android Platform. They had modified the Android Platform with proposed modules like Kernel Monitoring Module, File System Monitoring Module, Log-file, Monitoring Module and Network Monitoring Module to make it more secure. This architecture has the flaw of more battery consumption and also the disadvantage of platform dependence.

Enck et al. [6] proposed Kirin, a set of nine logical rules based on Android Permissions Requested for certification of the Android application at the installation time and raised the warning alarm if an application is found malicious. In Kirin, an Android Application was declared malicious if it requests for the Android Permissions like SET_DEBUG_APP, PHONE_STATE, RECORD_AUDIO, INTERNET ,PROCESS_OUTGOING_CALL, RECORD_AUDIO, ACCESS_FINE_LOCATION, RECEIVE_BOOT_COMPLETE, ACCESS_COARSE_LOCATION, RECEIVE_SMS and WRITE_SMS, SEND_SMS, INSTALL_SHORTCUT,UNINSTALL_SHORTCUT, SET_PREFERRED_APPLICATION and CALL which appeared to be the false assumption for the current Android Applications as most of the Benign applications requested for these Android Permissions. They had taken only samples of 311 applications for testing.

Enck et al. [7] proposed an extension to the Android OS version 2.1, Taint Droid, which monitors the flow of sensitive data of a user through the installed third party applications and observes how these third party applications access and manipulate the user data. TaintDroid can identify the explicit flow of only a few tainted critical data of the user and have the limitation of not detecting the implicit flow of tainted data.

SCAndroid [8] [9] is a Java program for static analysis of Android applications by extracting security specification from a manifest file and analyzing the data flow through the components of Android apps with source and sink specification.

Shabtai et al. [10] had evaluated machine learning algorithms such as Decision Tree, Naïve Bayes, Bayesian Networks (BN), PART, Boosted Bayesian Networks, Boosted Decision Tree, Random Forest, Voting Feature Intervals for classifying two types of Android applications: tools and games. They had used static features extracted from Android's Java byte-code (.dex files) and other file types such as XML-files of the Android application for the classification. They had performed an evaluation using a collection comprising 2,850 games and tools and achieved the highest detection rate of 92% using Boosted Bayes Net classifier with 19% False Alarm rate.

A.lanzi et al. [11] proposed the malicious applications detection system for personal computers based on the analysis of system call invoked by the application and achieved the detection rate of 89%.

Luo [12] proposed a tool for the detection of malicious applications with static analysis that leaks the critical user profile information. They have developed a translator for this purpose which transformed Dalvik bytecode to Java bytecode.

Batyuk et al. [13] introduced an automated reverse engineering technique for disabling the malicious code inside the Android applications without affecting the core functionality and generates the readable report for the user about the application.

Tang et al. [14] proposed the Security Distance Model based on the combination of Android Permissions Requested by the application. They used the threat point to represent the danger level of malware. They have made the hypothesis "Permission combinations with Security Distance of high threat point means applications with this permission combination have a big chance to threat mobile phone security". They had evaluated their model on 100 applications.

Sarma et al. [15] proposed a technique named CRCP (Category-based Rare Critical Permission signal) based on the analysis of the correlation between the Android Permissions Requested and type of application that requested the Android Permission. They had used SVM for the malicious application detection and achieved the warning rate of 8.12% and detection rate of 80.99%. They had evaluated 1, 58,062 Android apps from the Android Market and 121 malicious apps. They have used less no. of malicious application samples as compared to benign Android Application samples which can bias the training of SVM machine learning algorithms.

Sanz et al. [16] used machine learning algorithm for classification of Android applications in several categories like games, tools, entertainment etc. They used KNN, Support Vector Machine, J48, Random Forest and Tree Augmented Naïve (TAN) algorithms and achieved maximum ROC 0.93 with TAN. They had used the 820 samples of Android Applications from 7 categories of Android Applications. They extracted the features like Frequency of occurrence of the Printable Strings, different Android Permissions of the application itself & Android Permissions of the application extracted from the Android Market and classified the applications based on these features.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Zhou et al. [17] performed a systematic study for the detection of malicious applications in Android market with Droid Ranger. DroidRanger is permission based behavioral footprinting and heuristics-based filtering tool for detection of malicious Android application. The authors have analyzed 2, 04,040 applications with their tool and helped in finding the new Android malicious families.

Grace et al. [18] presented a system called Woodpecker to identify the flaws in the prevalent Android-essential permission based security model. Woodpecker had implemented interprocedural data flow analysis techniques to detect unauthorized access to sensitive data or privileged actions by untrusted applications. The results of this analysis showed that out of 13 privileged permissions examined, 11 were leaked and misused by the malicious applications to send out SMS, to wipe out the user data, obtain the user's geo-location data on the affected phones and to record user conversation, or all, without asking for any permission.

Sato et al. [19] proposed the method of calculating the malignancy score of the Android application based on the static features Android Permissions, Intent filter (action), Intent filter (category), and Process for classification of Android applications and have the accuracy of 90% with J48 Algorithms.

MAMA [20] proposed a technique for malware detection based on the Android Permissions Requested and Used Features using machine learning techniques KNN, Naïve Byes, SVM, J48, and Random Forest and have the accuracy of 94.83% with Random Forest. They had taken the 333 Android application samples from 30 Android Application categories like Arcade and Action, Multimedia & Video, Books, Music & Audio, Business, News & magazines, Card Games, Personalization, Casuals, Photography, Comics, Productivity, Communication, Puzzles, Education, Races, Enterprise, Sales, Entertainment, Society, Finance, Sports, Health, Tools, Libraries & Demos, Transportation, Lifestyle, Travels, Medicine, and Weather.

Huang et al. [21] also used machine learning techniques like AdaBoost, J48, Naïve Bayes and SVM for malicious application detection using static features like number of executable and linking format (ELF) files, no. of shared objects, Android Permissions required and has achieved the maximum accuracy of 81% with J48.

DREBIN [22] used six features; Hardware features Application, Android Permissions Requested, Filtered intents, Used Permissions, Suspicious API calls and Network addresses for the malware detection using SVM and has the accuracy of 94%. DREBIN have gathered many features from Application Code and Manifest File for broad static analysis. These features like Android Permissions, API calls, and network addresses are organized in sets of strings and embedded in a joint vector space. The dataset contains 123,453 benign applications and 5,560 malicious application samples. The method requires 10 seconds for an analysis on an average.

Liu & Liu [23] proposed Two-Layered Permission-Based Android Malware Detection Scheme using machine learning techniques for classification of benign and malicious applications. They have used the features Android Permissions Requested & Used Features for malicious applications detection.

Kang et al. [24] proposed malware detection method using Static Analysis associated with creator information. They had used a Naïve Bayes machine learning algorithm for malware detection. They had used Serial Number of Certificate, Likelihood of Android Permissions and Similarity Score for malicious application detection and achieved the detection rate of 98% and classification rate 90% for 20 malware families.

Cao et al. [25] proposed static analysis tool EDGEMINER that analyses the Android framework codebase for automatically generating the API implicit control flow transitions through the Android framework.

S. Malik & K. khatter [26] proposed malicious application detection and classification system by analysis Android Permission Requested features.The authors have extracted the Android Permission Requested features from 1060 android applications from 81 malware families using AndroData tool [27] and achieved the maximum accuracy of 98.3 % using Kernel Logistic Regression for detection of malicious applications and 87.83 % accuracy with LibLinear for classification of malicious applications.

III. INTRUSION DETECTION SYSTEM USING DYNAMIC FEATURES ANALYSIS

Kolbitsch et al. [28] have done an analysis of six malware families Allaple, Bagle, Mytob, Agent, and Netsky by finding the correlation between them in terms of the system call. They monitored the system calls that unknown program issues and match these calls with nodes in the behavior graph based on the correlation of system call executed. When enough of the graph has been matched, they concluded that the running program exhibits a behavior that is similar to previously observed malicious activity. At this point, the running process can be terminated and its previous and persistent modifications to the system can be undone. They had achieved the detection rate of 0.64 for 263 samples from above said malwares families.

Wang et al. [29] proposed software theft detection system with System Call Short Sequence Birthmark and IDSCSB Input Dependent System Call Subsequence Birthmark and examined how well they reflect unique behavioral characteristics of a program. They addressed the problem of software theft where a small part of the software is theft for doing malicious activities. However, this methodology can't detect the theft in a program which does not execute any system call or doesn't have any unique system call behavior. Moreover, detection rate depends on the threshold defined by the user for the uniqueness of system call behavior.

Gomez and Neamtiu [30] explored the four malware families DroidDream, DroidDreamLight, Zsone SMS, Geinimi and done their analysis based on the resources accessed, infiltration techniques and the payload used.

MADAM [31] (Multilevel Anomaly Detector for Android Malware) is a host based tool used to detect intrusion at kernel and user level. In this method, features system calls, running processes, free RAM, CPU usage, Idle/active, keystroke, called numbers, user/applications, sent/received SMS and Bluetooth/Wi-Fi analysis are used for malware detection using

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

K-NN machine learning classifier. It has the detection rate of 84.2% for detection of 10 Malware Families like Lena.B, BootKit, Moghava, TGLoader, OpFakeA, NickySpyB, KMin, Lotoor, DroidDream, and DroidKungFu. This methodology has been evaluated with 56 applications.

PermissionWatcher [32] is an Android application that classifies malicious applications installed on the device based on the 13 rules that are used to determine if the application is to be considered as suspicious. Within 6 weeks after release, this application was downloaded and installed by over 1,000 users. During evaluation of PermissionWatcher by these users, it was found that applications like Whatsapp, Google Translate, Barcode scanner, Facebook, Skype application found to be most frequently installed suspicious applications.

Tchakount and Dayang [33] proposed the system call based methodology for analyzing only two malware families, Super History Eraser and Task Killer. They discovered the participation of click event in triggering the malicious program task initiation.

Rosen et al. [34] gives a solution for privacy related API calls and proposed a method for application profile. In this solution, they had created a knowledge base of mappings between API calls and fine-grained privacy-related behaviors. This knowledge base is used for creating the behavior profile of Android applications. They had analyzed 80,000 Android applications by creating their behavior profiles.

VetDroid [35] is a dynamic analysis platform for analysis of Android Permissions usage behavior and Android Permissions acquired by the Android applications. In VetDroid, Android Permissions based Taint analysis has been done, in which, the flow of permission grant mechanism is observed in accessing the user critical data. In this work, the 21 malware families were analyzed with reconstructing their detailed malicious behaviors. VetDroid can find the fine-grained causes of sensitive information leakage by tracing the flow of permission usage. However, it poses extra runtime overhead in terms of execution speed and memory footprint for doing the behavior analysis of malwares.

Reina et al. [36] proposed CopperDroid, a framework for characterizing low-level OS-specific and high-level Android-specific behaviors for malware detection. For characterizing OS specific behavior, they had analyzed the features of System Call logs and for characterizing high-level Android-specific behaviors; they had analyzed the application log. This framework was evaluated with 1200 samples belonging to 49 Android malware families by the researchers.

Jeong et al. [37] proposed a malware detection technique based on system call and binder analysis. The proposed technique works in two ways. First, it monitors file operation activities of the malicious application by hooking the system calls like create, read and write. Secondly, it monitors and analyzes the communication between colluding applications by hooking the binder driver for IPC messages. The proposed technique can detect even the behavior of obfuscated malware with dynamic analysis of system call and binder driver.

Canfora et al. [38] Android malware detection method is based on selecting the longest sequences of system calls for the malware detection rather than considering the individual system call invocation by the Android application. They had used the SVM machine learning algorithms for malware detection and achieved the detection rate of 97%. The system call log of Android application execution on the real device had been used for features extraction. They had evaluated this method with 1000 benign and 1000 malicious applications from 28 malware families.

Narudin et al. [39] used Network traffic generated features for malicious application detection and classification and achieved 99% detection rate and 84.57% classification rate.

Sharma [40] proposed malware detection system based on the network traffic analysis using decision tree classifiers and achieved the accuracy of 90.32% for the malicious application samples from 20 malware families.

S. Malik [41] proposed malicious application detection system by extracting System Call Invoked features from 1060 Android Applications of 81 malware families and achieved the accuracy of 85% and ROC of 0.922 for detection of malicious application with machine learning algorithms.

IV. INTRUSION DETECTION SYSTEM USING HYBRID FEATURES ANALYSIS

Dini et al. [42] implemented Analytical Hierarchy Process for classification of Android application in three categories Trusted, Untrusted and Deceptive. They had proposed multi-criteria evaluation based on the features such as Types of Market, Type of Developer, Number of Downloads, User rating and Threat point based on the Android Permissions Requested. They have tested 180 Android application and found most of Android applications downloaded from unofficial market either infected or bad applications.

DroidRanger [43] is a multi-criteria based system that uses Android Permission Requested, other information in the manifest file, structural analysis of application code and heuristics based filtering for classification of an Android application. In this system permission- based filtering and behavioral footprint matching is suggested for detecting known malware families. The behavioral footprint matching is done based on the call graph of API call and semantic information from the manifest file. In DroidRanger, the heuristics-based filtering module and dynamic execution monitoring module are made for detecting unknown malware families. However, this system can detect the malicious applications from 10 malware families only. Moreover, detection process using a no. of features reduces the detection speed.

Wei et al. [44] presented the analysis of third party application permission model and possible adverse effect of malicious applications and their state of art. Based on the analysis they suggested several approaches for securing critical data of the device.

Andromaly [45] is another host-based malware detection system which uses machine learning techniques for doing an analysis of malicious applications, Smartphone's features, and events. Andromaly relies on Machine Learning techniques monitoring both the Smartphone's and user's behaviors by observing 88 features to describe these behaviors, ranging from activities to CPU usage. Andromaly identified the best classification method out of six classifiers (DTJ48, NB, Bayesian Networks, k-Means, Histogram and Logistic Regression) and achieved 99.9% accuracy rate with the

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

decision tree (J48) classifier. Although its authors achieved great accuracy but they had used self-written malware to test their framework.

Canfora et al. [46] proposed the malware detection approach based on the analysis of System Call & Android Permission features and classified the malicious application based on the three metrics.

Holavanalli et al. [47] proposed Blue Seal for analyzing the inter application and intra application permission flow. The implicit Android Permission flow was analyzed to detect effects of permission grant mechanism on leakage of critical data by the malicious application. The explicit flow of permission was examined to detect leakage of critical information within the applications. They had used 600 popular applications and 1,200 malicious applications for evaluating this technique and claimed this technique was practical and effective in driving Flow Permissions. However, this technique fails to identify the every source and sinks of information flow. They had just focused on file, network, and output stream APIs leading to potentially many false positives.

Rastogi et al. [48] developed Droid Chameleon, a systematic framework for evaluating anti-virus against various transformation techniques opted by the Android malware for making malicious applications. In this work, they had evaluated ten popular anti-malware products for Android and explored possible ways to improve current anti-malware solutions.

Zhu et al. [49] proposed a system for classification of dangerous applications based on Android Permissions and Application Description. They had used Naive Bayes with Multinomial Event Model algorithm to build the relation between the Description and the Android Permission List of an application. They had evaluated this framework with 5,685 applications in Android Market

Aung and Zaw [50] proposed a machine learning based malware detection framework by monitoring Android Permissions based features and events executed by the Android application. They have used J48, Random Forest and CART for malware detection of over 500 Android applications and achieved the accuracy of 91% with Random Forest.

Kim et al. [51] proposed Hybrid Intrusion Detection tool which used j48 decision tree for classification of malicious and benign applications. They have designed an automatic feature extraction tool written with Java scripts which can extract two features, Permission and Method API. To evaluate the proposed framework, they had collected 893 normal applications from Android market and 110 malicious applications from the Internet site and had the detection rate of 82.7%.

Yerima et al. [52] presented the static malicious detection approach based on the features - API, Android Permissions, and Commands. They have used their own tool with a component API detector, Command Detector, and Android Permission Detector. They had analyzed 2000 Android applications consisting of 1000 malicious applications from 49 Android Malwares families. They had selected 15 to 20 features from API, Android Permission, and command categories for the malicious application classification and had the AUC of 97.1% with the Bayesian model.

Canfora et al. [53] used System Calls features for detection of malicious web pages. In this work, the individual system call invoked and sequence of the system call invoked by malicious javascript have been analyzed and had high detection accuracy of 96%.

Talha et al. [54] proposed the Signature based Intrusion Detection System with Android Permissions analysis. The evaluation result shows the accuracy of 88%.

Verma & Muttoo [55] implemented a hybrid framework with machine learning algorithms K- means, J48 and ID3 for detecting Android malicious applications. They had extracted Android Permissions and Intents features of the Android application for training and testing the classifiers and achieved the accuracy of 94% with J48.

Long Wen, and Haiyang Yu [56] proposed PCA-RELIEF algorithm foe android malware detection with machine learning algorithum. The authors have extracted features permission, intent, uses-feature, application and API,CPU consumption, the battery consumption, the number of running processes and the number of short message from 2000 android applications and analyzed them with SVM and achieved the accuracy of 95.2 % .

K. Khatter & S. Malik [57] proposed risk and ranking algorithm for malicious application detection and classification with hybrid features analysis of 1060 android applications from 81 malware families and achieved the maximum accuracy of 99.2 % for malicious application detection using SVM and 88.7% accuracy with Random Forest Algorithm for classification of malicious applications.

Table 1 different methodologies are opted by the researchers for malware detection and compared them on the basis of Techniques used, features used for analysis and their performance.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Table I: Related Work

Researches	Year	Methodology Used	Features Used	Analysis Technique	Remarks
Schmidt et al. [5]	2008	Modified the Android kernel with security modules	Process list, a list of open files, network traffic, symbol table and system call traces	Static analysis	Check for any abnormality in system behavior
Kirin [6]	2009	Proposed a set of 9 logical rules on Android Permissions Requested at the installation time of Android Application	Android Permissions Requested	Static analysis	Have more false alarms as only 9 rules are used for detecting the malicious application
Kolbitsch et al. [28]	2009	Done an analysis of six malware families Allaple, Bagle, Mytob, Agent, and Netsky by finding the correlation between them in terms of the system call.	System Calls	Dynamic analysis	Achieved 0.64 detection rate with behavior graph analysis of 263 applications
Wang et al. [29]	2009	Proposed software theft detection system with System Call Short Sequence Birthmark and IDSCSB Input Dependent System Call Subsequence Birthmark	System Calls	Dynamic Analysis	Detection rate depends on the threshold defined by the user for the uniqueness of system call behavior.
Enck et al. [7]	2010	TaintDroid, an extension to the Android Operating system	Monitors the flow of sensitive user data, its access and manipulation by third party application	Static analysis	Evaluated with only 30 Android application, monitors only explicit flow of tainted data
Shabtai et al.[10]	2010	Used the machine learning techniques for the classification of Android application into two types Games & Tools	Features are extracted from Android's Java byte-code and other file types such as XML-files.	Static analysis	Detection rate of 92% using Boosted BayesNet classifier with 19% False Alarm rate.
Tang et al. [14]	2011	Proposed the Security Distance Model based on the combination of Android Permission requested by the application	Android Permissions	Static Analysis	Analyzed combination of requested Android permission for calculating the threat point of the application
Sarma et al. [15]	2012	Proposed a technique named CRCP(Category-based Rare Critical Permission signal) based on the analysis of the correlation between the permission requested and type of application requested the permission	Android Permissions Requested	Static analysis	Warning rate of 8.12% corresponds to 80.99% detection rate for 121 malicious applications
Sanz et al. [16]	2012	Used machine learning algorithm for classifying the Android application	Frequency of occurrence of the Printable Strings, different Android permissions, and Used Features	Static Analysis	Used the algorithms KNN, SVM, J48, Random Forest and Tree Augmented Naïve (TAN) and achieved maximum ROC 0.93 with TAN
Zhou et al. [17]	2012	Proposed DroidRanger, permission based behavioral footprinting and heuristics-based filtering tool for detection of malicious Android application.	Android Permissions	Static Analysis	Analyzed 204040 applications and also discovered the new Android malicious families.
MADAM [31]	2012	Malware detection using K-NN machine learning Classifier	System calls, running processes, free RAM, CPU usage, idle/active, key-stroke, called numbers, User/Applications, sent/received SMS, Bluetooth/Wi-Fi analysis	Dynamic Analysis	Detection rate of 84.2% for detection of 10 Malware Families like Lena.B, BootKit, Moghava, TGLoader, OpFakeA, NickySpyB, KMin, Lotoor, DroidDream, Droid Kung Fu

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Researches	Year	Methodology Used	Features Used	Analysis Technique	Remarks
Grace et al. [18]	2012	Presented a system called Woodpecker employs interprocedural data flow analysis techniques to systematically expose possible capability leaks	Android Permissions	Static Analysis	Analyzed the involvement of different Android Permissions in escalation attack
PermissionWatcher [32]	2012	13 rules that are used to determine if application is to be considered as suspicious	Android Permissions	Dynamic Analysis	Installed by over 1,000 users and applications like Whatsapp, Google translate, Barcode scanner, Facebook, Skype application found to be most frequently installed suspicious applications
Dini et al. [42]	2012	Proposed Analytical Hierarchy Process, a multi-criteria evaluation based on the threat score	Types of Market, Type of Developer, Number of Downloads, User rating , threat point based on the Requested Android Permissions	Hybrid Analysis	Tested 180 Android application and found most of Android application downloaded from unofficial market either infected or bad application.
DroidRanger [43]	2012	Permission- based filtering and behavioral footprint matching are suggested for detecting malware families.	Permission requested, other information in the manifest file, structural analysis of application code and heuristics based filtering	Hybrid Analysis	Detection the malicious application from 10 malware families
Zhu et al. [49]	2012	Have used Naive Bayes with Multinomial Event Model algorithm to build the relation between the description and the permission list of an application	Description and Android Permissions Requested	Hybrid Analysis	Evaluated framework with 5,685 applications in Android Market
Andromaly [45]	2012	Another host-based malware detection system which uses machine learning techniques	88 features ranging from activities to CPU usage	Hybrid Analysis	99.9% accuracy rate with the decision tree (J48) classifier
Tchakount&Dayang[33]	2013	System call based methodology for analyzing only two malware families Super History Eraser and Task Killer	System Calls	Dynamic analysis	Analyzed only 2 malware families and discover the participation of click event in triggering the malware startup
Sato et al. [19]	2013	Method of calculating the malignancy score of the Android application based on the static features	Android Permissions, Intent filter (action), Intent filter (category), and Process	Static Analysis	Accuracy of 90% with J48 Algorithms.
MAMA [20]	2013	Technique for malware detection using machine learning techniques	Android Permissions Requested and Used Android Permissions features	Static Analysis	Accuracy of 94.83% with Random Forest
Huang et al. [21]	2013	Used machine learning technique like AdaBoost, J48, Naïve Bayes, and SVM for malicious application detection using static features	Number of executable and linking format (ELF) files, no. of shared objects, Android Permissions required	Static Analysis	Accuracy of 81% with J48
Canfora et al. [46]	2013	Used SVM for malware detection	System Call and Android Permissions feature	Hybrid Analysis	Analyzed few combinations of Android Permission & System calls features
Rosen et al. [34]	2013	Solution for privacy related API calls and proposed a method for application profile	API calls and fine-grained privacy-related behaviors	Static Analysis	Analyzed 80,000 Android applications by creating their behavior profiles.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Researches	Year	Methodology Used	Features Used	Analysis Technique	Remarks
Holavanalli et al. [47]	2013	Proposed Flow Permissions to examine the implicit and explicit flow of permission grant mechanism within the applications.	Android Permissions	Hybrid Analysis	Focused only on file, network, and output stream APIs leading to potentially many false positives
VetDroid [35]	2013	Analysis of permission usage behavior and permission acquired by the Android applications	Android Permissions	Dynamic Analysis	Analyzed 21 malware families
Aung and Zaw [50]	2013	Proposed a machine learning based malware detection framework	Android Permissions based features and events executed	Hybrid Analysis	Accuracy of 91% with Random Forest
Kim et al. [51]	2013	Proposed hybrid intrusion detection tool which uses j48 decision tree	Android Permissions and method API.	Hybrid Analysis	Evaluated the proposed tool with 893 normal applications from Android market and 110 malicious applications and have the detection rate of 82.7%.
Yerima et al. [52]	2013	Developed component API detector, Command Detector, and permission Detector for malware detection	API, Android Permissions, and command	Hybrid Analysis	Analyzed 2000 Android Applications consist of 1000 malicious applications from 49 Android Malwares families and had the AUC of 97.1% with Bayesian model
Reina et al.[36]	2013	Copper- Droid, an approach built on top of QEMU to automatically perform out-of-the-box dynamic behavioral analysis of Android malware	Application log and system call log	Dynamic Analysis	Framework was evaluated with 1200 samples belonging to 49 Android malware families by the researchers
DREBIN [22]	2014	Used SVM for malware detection	Use six features, Hardware features Application, Requested Permissions, Filtered intents, Used permissions, Suspicious API calls and Network addresses	Static Analysis	Achieved an accuracy of 94%.
Liu & Liu [23]	2014	Proposed Two-Layered Permission-Based Android Malware Detection Scheme using machine learning techniques	Requested Permissions & Used Features	Static Analysis	
Jeong et al. [37]	2014	Proposed a malware detection technique based on system call and binder analysis	System call and binder	Dynamic Analysis	Proposed technique can detect the even the behavior of obfuscated malware
Kang et al. [24]	2015	Proposed malware detection method using static analysis associated with creator information	Serial Number of Certificate, Likelihood of Permission and Similarity Score	Static Analysis	Detection rate of 98% and 90% classification rate for 20 malware families.
Talha et al.[54]	2015	Signature based Intrusion Detection System	Android Permissions	Static Analysis	88% accuracy
Calfora et al [38]	2015	Used machine learning algorithms SVM for Android malware detection	Sequence of System call	Dynamic Analysis	97% detection rate for malicious application from 28 malware families
Cao et al. [28]	2015	Proposed static analysis tool EDGEMINER for control flow analysis	API implicit control flow	Static analysis	Better tool for malware detection with control flow analysis

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Researches	Year	Methodology Used	Features Used	Analysis Technique	Remarks
Narudin et al.[39]	2016	Used machine learning algorithms Bayes network, multi-layer perceptron, decision tree (J48), K-nearest neighbor and random forest for malicious application analysis and classification 49 different families	Network traffic generated Android features from four groups: basic information, content-based, time-based and connection-based features.20 benign and 1000 malicious application	Dynamic Analysis	99% Detection Rate & 84.57% Classification Rate for malicious application of 49 malware families
Nancy & Sharma [40]	2016	Malware detection techniques based on the network traffic analysis	20 Network Traffic Features	Dynamic Analysis	90.32% accuracy with decision tree classifier for malicious application from 20 malware families
Verma & Muttoo [55]	2016	Machine learning Implementation for malicious application detection	Android permissions & Intent	Hybrid Analysis	Accuracy of 94% with J48
S. Malik [41]	2017	Malware detection techniques based on the System call Invoked features analysis	System Call Invoked features	Dynamic Analysis	Accuracy of 85 % with Random Forest
S. Malik & K.Khatter [26]	2018	Malware detection and classification techniques based on the Android Permission Requested features analysis	Android Permission Requested features	Static Analysis	Accuracy of 98.3 % with Kernel Logistic Regression
K.khatter & S. Malik [57]	2018	Malware detection and classification techniques based on the Android Permission Requested and System call invoked features analysis	Android Permission Requested and System Call Invoked features	Hybrid Analysis	Accuracy of 99.2 % with SVM

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

The static analysis techniques have been opted by researchers [5-27]. Schmidt et al. [5], Enck et al. [7], Yerima et al. [52] have modified current Android Operating System but these solutions are limited to the current version of operating system and these are unable to provide a solution for the upcoming Android Versions. Kirin [6], PermissionWatcher [32] are the rule based solutions for malware detection with Android Permissions. However, these solutions are proposed for only a few set of combinations of Android Permissions such as SEND_SMS, INTERNET, and ACCESS_COURSE_LOCATION etc. Tang et al [14] , Sarma et al. [15] ,Zhou et al. [17] ,Grace et al.[18], proposed the knowledge based solution primly with analysis of the correlation between different Android Permissions but they are also limited their research on few important Android Permissions such as SEND_SMS, INTERNET, WRITE_OWNER_DATA.

Shabtai et al. [10] ,Sanz et al. [16], MADAM [31], Andromaly [45], Sato et al. [19],MAMA [20], Haung et al.[21], Canfora et al. [38], Aung & Zaw [50], Kim et al. [51], Yerima et al. [52], DREBIN [22] , Kang et al. [24], Calfora et al [38], Narudin et al. [39], Nancy & Sharma [40], Verma & Muttoo [55] proposed methodologies for malware detection using machine learning techniques. The machine learning techniques have the power of analyzing the thousands of features with more accuracy and also have the ability to discover new malwares. In these methodologies, the many features like Java Byte Code, XML files, Android Permissions, Used Features, System Calls, Running Processes, CPU usage, called numbers, API Calls, Intents etc. were analyzed. The features analyzed in these methodologies are Static features, Dynamic Features, and Hybrid Features.

As depicted in Table II,out of static analysis methodologies proposed so far, MAMA [20] had achieved maximum accuracy of 94.83% by Static Features Analysis with Random Forest but this research can't malware families. In 2015, Kang et al. [24] proposed malware detection system with Static Features Analysis and had the good detection rate of 98% but this can classify maximum 20 malware families.

The maximum detection rate of 84.2% was achieved by the proposed methodology of MADAM [31] for 10 malware families till 2013. Calfora et al. [38] proposed malware detection system by analysis of the sequence of system calls executed using machine learning techniques in 2015 and achieved the detection rate of 97% for 28 malware families. Another remarkable research was done by Narudin et al. [39] on malware detection with network features analysis and achieved 99% detection rate & 84.57% classification rate for classification of 49 malware families.

S. Malik & K. Khatter [27] proposed malware detection and classification techniques by analyzing the static features Android Permission Requested with machine learning algorithms and achieved the 98.3 % accuracy with Kernel Logistic Regression Techniques .Authors have analyzed 1060 Android Applications from 81 malware families.

Andromaly [45] had analyzed 88 hybrid features for malware detection using machine learning algorithms and had the accuracy of 99% but could not classify malware families. However, this accuracy was achieved with extracting and analyzing more no. of features. The extraction of large no. of features of the Android Application is a cumbersome job and it can increase the detection time.

K. Khatter & S. Malik [57] had analyzed the Android Permission Requested and System Call Invoked Hybrid features of android applications and achieved the 99.2 % of accuracy for detection of malicious application and 88.7 % accuracy for classification of malicious application from 81 malware families.

Thus intrusion detection with more accuracy with less no. of Features Extraction and Analysis is an open research area. It is also revealed that the state of art works proposed so far can detect and classify the Android applications from maximum 81 malware families. Thus Intrusion Detection System for classifying more no. of malware families is a research goal.

Feature Selection for the analysis of Android applications is the crucial step in malware detection. The Static Features analyzed in the related work are Android Permissions, intents filters, network address, Java code, strings and hardware components. The related work revealed that Android Permission is the most important feature which reflects the malicious intention of the Android applications [58].. Android Permissions are the first barriers to the attacker as each malicious code needs permission to invoke API calls for accessing the critical resources of the Android Mobile Phones. Thus malicious applications can be detected by analyzing Android Permissions feature.

Dynamic Features analyzed by the researchers are system calls, network traffic features, system components and user interaction. The System Calls and Network Traffic are two main types of dynamic features used in the recent work. Tracing the System Call patterns reflects the behavior of the application with the operating system [59], and the Network Traffic Features analysis reflects the behavior of an application during network connectivity.

Every demand of application using services of mobile hardware passes through the such as read, write and open. If the malicious application controls other applications or it is sending data over the network, this behavior will be printed in system call invoked pattern. Thus, with system call pattern analysis, the run time behavior of the malicious application can be detected.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Table II Related Work with Analysis of Android Features using Machine Learning Algorithms

Authors Name & Year	Methodology Used	Features Used	Can Classify Malware Families and	No. of Malware Families can classify	Accuracy
Kolbitsch et al. (2009) [28]	Correlation between them in terms of the system call.	System calls	Yes	6	64 % with behavior graph analysis of 263 applications
Sarma et al. (2012) [15]	Proposed a technique named CRCP(Category-based Rare Critical Permission signal) based on the analysis of the correlation between the permission requested and type of application requested the Android Permissions	Android Permissions Requested	No	NA	80.99% for 121 malicious applications
MADAM (2012) [31]	Malware detection using K-NN machine learning Classifier	System calls, running processes, free RAM, CPU usage, idle/active, key-stroke, called numbers, User/Applications, sent/received SMS, Bluetooth/Wi-Fi analysis	Yes	10	84.2%
Andromaly (2012) [45]	Host-based malware detection system using machine learning techniques	88 features ranging from activities to CPU usage	No	NA	99.9% with the decision tree (J48) classifier
Sato et al. (2013) [19]	Method of calculating the malignancy score of the Android application based on the static features	Android Permissions, Intent filter (action), Intent filter (category), and Process	No	NA	90% with the decision tree (J48) classifier
MAMA (2013) [20]	Technique for malware detection using machine learning techniques	Android Permissions requested and used features	No	NA	94.83% with Random Forest
Huang et al. (2013) [21]	Used machine learning technique like AdaBoost, J48,Naïve Bayes, and SVM for malicious application detection using static features	Number of executable and linking format (ELF) files, no. of shared objects, Android Permissions required	No	NA	81% with the decision tree (J48) classifier
Aung and Zaw (2013) [50]	Proposed a machine learning based malware detection framework	Android Permissions based features and events executed	No	NA	91% with Random Forest
Kim et al. (2013) [51]	Proposed hybrid intrusion detection tool which uses j48 decision tree	Android Permissions and method API.	No	NA	82.7 % with the decision tree (J48) classifier
Yerima et al. (2013) [52]	Developed component API detector, Command Detector, and permission Detector for malware detection	API, Android Permissions, and commands	Yes	49	97.1 % for 2000 android applications
Arp et al. (2014) [25]	Used SVM for malware detection	Used features Hardware features Application, Requested permissions, Filtered intents, Used permissions, Suspicious API calls and Network addresses	No	NA	94%.
Kang et al. (2015) [24]	Proposed malware detection method using static analysis associated with creator information	Serial Number of Certificate, Likelihood of Android Permissions and Similarity Score	Yes	20	98%
Talha et al. (2015) [54]	Signature based intrusion detection system	Android Permissions	No	NA	88%
Calfora et al (2015) [38]	Used machine learning algorithms SVM for Android malware detection	Sequence of System calls	Yes	28	97%
Nancy & Sharma (2016) [40]	Malware detection techniques based on the network traffic analysis	20 Network Traffic Features	Yes	20	90.32%
Verma & Muttoo (2016) [55]	Machine learning Implementation for malicious application detection	Android permissions & Intent	No	NA	94% the decision tree (J48) classifier
S. Malik (2017)[41]	Machine learning techniques for malicious application detection	System Call Invoked	Yes	81	85 %
S. Malik & K. Khatter (2018)[26]	Machine learning techniques for malicious application detection and classification	Android Permission Requested	Yes	81	98.3 % accuracy with Karenal Logistic Regression
K.Khatter & S. Malik (2018)[57]	Machine learning techniques for malicious application detection	System Call Invoked & Android Permission Requested	Yes	81	99.2 % accuracy with SVM

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

V. CONCLUSION

In Anomaly based Intrusion Detection System, the behavior of the android application is observed by extracting and analyzing the static or dynamic features or hybrid features of the application. In Static Features Analysis technique, the features of Android applications are extracted from its Java bytecode without executing it and they are analyzed to find the malicious applications. Static Features are easy to extract as there is no need to run the code and also the malicious application under test does not cause any real harm to the device that carries the malicious applications but its analysis fails to detect the presence of malicious code that can be downloaded by the malicious application during run time. Therefore, Static Features Analysis technique has more false negative and false positive rate. In Dynamic Features Analysis technique, the dynamic features like system call invoked, API executed etc. are analyzed to detect the malicious applications. This analysis is more informative as it observes the runtime behavior of an application but it costs more as the malicious application can harm the device while running and also, the features are difficult to extract and sometimes requires rooted devices too. Furthermore, it will have more false negative rate if the malicious code inside the Android application does not execute. In Hybrid Features Analysis technique, the static and dynamic features are extracted and analyzed to detect the malicious application. Thus, this analysis has the advantage of both Static Features Analysis and Dynamic Features Analysis; consequently, this has more accuracy than Static Features Analysis or Dynamic Features Analysis. As it involves more complex features extraction process, it is the least explored area in the literature review. There is just 10% of the literature review that constitutes the research in the area of hybrid feature analysis. Thus, Hybrid Features Analysis technique is an open area of research.The template will number citations

REFERENCES

[1] Pathak,P. (December 19, 2018) Don't use antivirus on your Android phone to keep it safe, use these tips.Retrived from https://www.indiatoday.in/technology/tech-tips/story/don-t-use-antivirus-on-your-android-phone-to-keep-it-safe-use-these-tips-1412885-2018-12-19

[2] Khandelwal,S. (February 04,2019). Several Popular Beauty Camera Apps Caught Stealing Users' Photos. Retrieved from https://thehackernews.com/2019/02/beauty-camera-android-apps.html.

[3] Gatlan,S. (January 17,2019). Android Apps Steal Banking Info, Use Motion Sensor to Evade Detection.Retrieved from https://www.bleepingcomputer.com/news/security/android-apps-steal-banking-info-use-motion-sensor-to-evade-detection/

[4] Security Week News. (2016, June 29) *Overwhelming Majority of Android Devices Don't have latest Security Patches* [Online]. Available:http://www.securityweek.com/overwhelming-majority-Android-devices-dont-have-latest-security-patches

[5] A. Schmidt, H. Schmidt, J. Clausen, A. Camtepe, and S. Albayrak, "Enhancing Security of Linux-based Android Devices," *in Proc. 15th Int. Linux Kongress*, 2008.

[6] W. Enck, M. Ongtang, and P. Mcdaniel, "On Lightweight Mobile Phone Application Certification," in *Proc. of the 16th ACM conference on Computer and communications security*. ACM, 2009.

[7] W. Enck, P. Gilbert, B.G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," Osdi '10, vol. 49, pp. 1–6, 2010.

[8] A. P. Fuchs, A. Chaudhuri, J. S. Foster, "SCAndroid: Automated Security Certification of Android Applications," Department of Computer Science, University of Maryland, College Park, Tech. Rep. CS-TR-4991, Nov. 2009.

[9] *Control Flow Graph Scanning for Android* [Online] (2017,June 20). Available: https://github.com/douggard/CFGScAndroid

[10] A. Shabtai, Y. Fledel, and Y. Elovici, "Automated static code analysis for classifying Android applications using machine learning," in *Proc 2010 Int. Conf. Comput. Intell. Secur. CIS 2010*, pp. 329–333, 2010.

[11] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "AccessMiner: Using System-Centric Models for Malware Protection," in *Proc. 17th ACM Conf. Comput. Commun. Secur. CCS'10*, pp. 399–412, 2010.

[12] K. Luo. "Using static analysis on Android applications to identify private information leaks," Dept. of Computing and Information Sciences, Kansas State University, RPE Rep., 2011.

[13] L. Batyuk, M. Herpich, S. Camtepe, K. Raddatz, A.D. Schmidt, S. Albayrak, "Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications," in *Proc. of Malicious and Unwanted Software (MALWARE),6th International Conference*, pp. 66–72,2011.

[14] W. Tang, G. Jin, J. He, and X. Jiang, "Extending Android security enforcement with a security distance model," in *Proc. Internet Technology and Applications (iTAP)*, 2011.

[15] B. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-rotaru, and I. Molloy, "Android Permissions: A Perspective Combining Risks and Benefits," *Symp. Access Control Model. Technol.*, pp. 13–22, 2012.

[16] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, "On the automatic categorisation of Android applications," in *Proc. 2012 IEEE Consum. Commun. Netw. Conf. CCNC'2012*, pp. 149–153, 2012.

[17] Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in *Proc. IEEE Symp. Secur. Priv.*, no. 4, pp. 95–109, 2012.

[18] M. C. Grace, Y. Zhou, Z. Wang, and X. Jiang, "Systematic Detection of Capability Leaks in Stock Android Smartphones," in NDSS, 2012.

[19] R. Sato, D. Chiba, S. Goto. "Detecting Android malware by analyzing manifest files," in *Proc. of the Asia-Pacific Advanced Network*, pp. 23-31, 2013.

[20] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. G. Bringas, and G. Álvarez Marañón, "Mama: Manifest Analysis for Malware Detection in Android," *Cybern. Syst.*, vol. 44, no. 6–7, pp. 469–488, 2013.

[21] C. Y. Huang, Y. T. Tsai, C. H. Hsu, "Performance evaluation on permission-based detection for Android malware," *Advances in Intelligent Systems and Applications. Springer Berlin Heidelberg*. vol.2 , pp. 111-120,2013.

[22] D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," *Symp. Netw. Distrib. Syst. Secur.*, pp. 23–26, 2014.

[23] X. Liu, J. Liu. "A two-layered permission-based Android malware detection scheme," in *Proc. of Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference*, 2014.

[24] H. Kang, J. Jang, A. Mohaisen, and H. K. Kim, "Detecting and Classifying Android Malware Using Static Analysis along with Creator Information," *Int. J. Distrib. Sens. Networks*, vol. 7, 2015.

[25] Y. Cao, Y. Fratantonio, A. Bianchi, and M. Egele, "EdgeMiner : Automatically Detecting Implicit Control Flow Transitions through the Android Framework," *NDSS 2015*, pp. 8–11, 2015.

[26] Sapna Malik, Kiran Khatter,"Malicious Application Detection and Classification System for Android Mobiles," International Journal of Ambient Computing and Intelligence,Vol. 9, no. 1,pp. 95-114,2018.

[27] Sapna Malik, Kiran Khatter, "AndroData: A tool for static and dynamic features extraction of Android Applications," International Journal of Applied Engineering Research, ISSN 0973-4562, Vol. 10, no. 94, pp. 98-102, 2015.

[28] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, X. Wang, U. C. S. Barbara, and S. Antipolis, "Effective and Efficient Malware Detection at the End Host," *System*, vol. 4, no. 1, pp. 351–366, 2009.

[29] X. Wang, Y. C. Jhi, S. Zhu, and P. Liu, "Detecting software theft via system call based birthmarks," in *Proc. Annu. Comput. Secur. Appl. Conf. ACSAC*, pp. 149–158, 2009.

[30] L. Gomez and I. Neamtiu, "A Characterization of Malicious Android Applications," University of California, Riverside, Tech. Rep., 2011.

[31] G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "MADAM: A multi-level anomaly detector for Android malware," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes*

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 10, October-2019

Bioinformatics), vol. 7531 LNCS, pp. 240–253, 2012.

[32] E. Struse, J. Seifert, S. Ullenbeck, E. Rukzio, and C. Wolf, "PermissionWatcher: Creating User Awareness of Application Permissions in Mobile Systems," *in Ambient Intelligence*, pp. 65–80,2012.

[33] F. Tchakounté and P. Dayang, "System Calls Analysis of Malwares on Android," *International Journal of Science and Technology*, vol. 2, no. 9, pp. 669–674, 2013.

[34] S. Rosen, Z. Qian, and Z. M. Mao, "Appprofiler: a flexible method of exposing privacy-related behavior in Android applications to end users," in *Proc.of the third ACM conference on Data and application security and privacy*, pp. 221–232, 2013.

[35] Y. Zhang, M. Yang, B. Xu, Z. Yang, G. Gu, P. Ning, X. S. Wang, and B. Zang, "Vetting undesirable behaviors in Android apps with permission use analysis," in *Proc. of the 2013 ACM SIGSAC Confer.ence on Computer and Communications Security*, pp. 611–622, 2013

[36] A. Reina, A. Fattori, L. Cavallaro. "A system call-centric analysis and stimulation technique to automatically reconstruct Android malware behaviors," *in Proc. of Euro Sec'13*, pp. 1-6,2013.

[37] Y. S. Jeong, H. T. Lee, S. J. Cho, ,S. Han, and M. Park, "A kernel-based monitoring approach for analyzing malicious behavior on Android," In *Proc.of the 29th Annual ACM Symposium on Applied Computing*. ACM, 2014.

[38] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Detecting Android Malware Using Sequences of System Calls," in *Proc. 3rd Int. Work. Softw. Dev. Lifecycle Mob.*, pp. 13–20, 2015.

[39] F. A. Narudin, A. Feizollah, N. B. Anuar and A. Gani, " Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343-57,2009.

[40] Nancy , D. Sharma, "Android Malware Detection using Decision Trees and Network Traffic," *International Journal of Computer Science and Information Technologies*, vol. 7, no. 4, pp. 1970–1974, 2016.

[41] Sapna Malik, "Android System Call Analysis for Malicious Application Detection", International Journal of Computer Sciences and Engineering, Volume-5, Issue-11, Nov 2017.

[42] G. Dini, F. Martinelli, I. Matteucci, M. Petrocchi, A. Saracino, and D. Sgandurra, "A Multi-Criteria-Based Evaluation of Android Applications," *in Trusted Systems*, pp. 67–82, 2012.

[43] 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 NDSS, 2012

[44] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Malicious Android applications in the enterprise: What do they do and how do we fix it?," in *Data Engineering Workshops (ICDEW),IEEE 28th International Conference* , pp. 251–254, 2012.

[45] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss,"Andro- maly: A Behavioral Malware Detection Framework for Android Devices". *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161-190,2009.

[46] G. Canfora, M. Francesco ,A.V. Corrado, "A classifier of malicious Android applications." *Availability, Reliability and Security (ARES), 2013 Eighth International Conference* . IEEE, 2013.

[47] S. Holavanalli, D. Manuel, V. Nanjundaswamy, B. Rosenberg, F. Shen, S. Y. Ko, and L. Ziarek,"Flow permissions for Android," in *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference*, pp. 652–657,2013.

[48] V. Rastogi, Y. Chen, and X. Jiang, "Droidchameleon: evaluating Android anti-malware against transformation attacks," in *Proc. of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 329–334,2013.

[49] J. Zhu, Z. Guan, Y. Yang, L. Yu, H. Sun, and Z. Chen, "Permission-based abnormal application detection for Android," in *Information and Communications Security*, pp. 228–239,2012.

[50] Z. Aung and W. Zaw, "Permission-based Android malware detection," *International Journal Of Scientific and Technology Research*, vol. 2, no. 3, 2013.

[51] D. Kim, J. KIm, and S. Kim. "A malicious application detection framework using automatic feature extraction tool on Android market." in *Proc. of the 3rd International Conference on Computer Science and Information Technology, ICCSIT* , 2013.

[52] S.Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik. "A New Android Malware Detection Approach Using Bayesian Classification," *in Advanced Information Net- working and Applications (AINA), 2013 IEEE 27th International Conference*, pages 121–128, March 2013.

[53] G. Canfora, E. Medvet, F. Mercaldo, C.A.Visaggio,"Detection of malicious web pages using system calls sequences," *in Proc. of the 4th International Workshop on Security and Cognitive Informatics for Homeland Defense (SeCIHD 2014)*, pp. 226-38,2014.

[54] K. A. Talha, D. I. Alper, and C. Aydin, "APK Auditor: Permission-based Android malware detection system," *Digit. Investig.*, vol. 13, pp. 1–14, 2015.

[55] S. Verma and S. K. Muttoo, "An Android Malware Detection Framework-based on Permissions and Intents," *Defense Science Journal*, vol. 66, no. 6, pp. 618–623, 2016.

[56] Wen, L., & Yu, H. (2017, August). An Android malware detection system based on machine learning. In AIP Conference Proceedings (Vol. 1864, No. 1, p. 020136). AIP Publishing.

[57] Kiran Khatter,Sapna Malik," Ranking and Risk Factor Scheme for Malicious applications detection and Classifications" International Journal of Information System Modeling and Design, 9(3), 67-84,2018.

[58] Sapna Malik, Kiran Khatter, "Behavior Analysis of Android Application," International Journal of Control Theory and Application, Vol. 9, no. 11, pp. 5307-5322, 2016..

[59] Sapna Malik, Kiran Khatter "System Call Analysis of Android Malware Families", Indian Journal of Science and Technology, Vol. 9, no. 21,2016. DOI: 10.17485/ijst/2016/v9i21/90273.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)