International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

A Hybrid Model for Android Malware Detection

Vinisha Malik, Sandip Kumar Goyal, Naveen Malik

Abstract: ***Android malware have risen exponentially over the past few years, posing several serious threats such as system damage, financial loss, and mobile botnets. Various detection techniques have been proposed in the literature for Android malware detection. Some of the techniques analyze static parameters such as permissions, or intents, whereas, others focus on dynamic parameters such as network traffic or system calls. Static techniques are relatively easier to implement, however, stealthy recent malware evade static detection by virtue of update attacks. Dynamic detection can be used to detect such stealthy malware, however, it increases the computation overhead. Hence, both kinds of techniques have their own advantages and disadvantages. In this paper, we have proposed an innovative hybrid detection model that uses both static and dynamic features for malware analysis and detection. We first rank the static and dynamic parameters according to the information gain and then apply machine learning algorithms in the testing phase. The results indicate that hybrid approach is better than both static and dynamic approaches and the proposed model achieves 94.2% detection accuracy with Decision Tree classifier.***

Keywords: Android Security, Malware Detection, Permissions, Intents, Network Traffic.

I. INTRODUCTION

Smartphones are continuously stepping up the ladder of popularity and leaving behind the laptops and desktops in this regard. Globally, 95% of the users owned smartphones as compared to 75% of the users with desktops/laptops in 2018. This number was 87 percent and 88 percent for the smartphones and PC/Laptops in 2015. Moreover, 65% of the users consider smartphones to be relatively important as compared to 15% in favor of the desktops/laptops [1]. Global smartphone shipments touched 345 million units in Q1 of 2018, and 330 million units in Q1 of 2019 [2]. On the other hand, in 2019, global desktop shipments have been forecasted to touch 89.1 million units [3]. This huge difference between the shipments of the smartphones and the desktops reflects the smartphones' popularity amongst the users.

Revised Manuscript Received on October 30, 2019.
***** Correspondence Author
Vinisha Malik*, PhD Research Scholar Department of Computer Science & Engineering, Maharishi Markandeshwar (Deemed To Be University) Mullana, Ambala , India. Email: Vinishamalik317@gmail.com
Sandip Kumar Goyal, Professor, Department of Computer Science & Engineering, Maharishi Markandeshwar (Deemed To Be University) Mullana, Ambala, India. Email: skgmmec@gmail.com
Naveen Malik, Assistant Professor Computer Science & Engineering, Dcrust Govt University, Murthal, Sonepat, ndia. Email: Naveenmalik317@gmail.com

According to a forecast projection [4], the shipment volume of Android is about 1191 million units with a market share of 86.7%, whereas, for iOS, the total shipments are around 183.5 million units with a market share of 13.3%. This trend clearly reflects that amongst the mobile OS, Android has been the best seller compared to others like iOS, Windows or Blackberry. Moreover, Android smartphones offer a huge list of applications through the Google Play Store such as music applications, games applications, chat platforms, online banking and various others which makes these smartphones highly popular.

Such increasing popularity of Android has come hand-in-hand with the rise of malware towards the Android platform. Various techniques have been used by malware developers to spread malicious applications such as repackaging, update attacks and drive-by-downloads. Omnipresent Internet connectivity and procurable personal details such as contacts, messages, and banking credentials have attracted the malware developers towards the smartphones. Malware attacks are incessantly leaping up as visible from the Table I. Malware attacks on Android devices have further escalated in the first half of 2019 with key reasons owing to an increased usage of mobile banking applications [6].

Table I: Number of Android Malware Sample Recorded Every Year

Year	Malware Samples
2012	214,327
2013	1,192,035
2014	1,548,129
2015	2,333,777
2016	3,246,284
2017	3,002,482
2018	3,189,480(Q1-Q3)

Motivation: With such an increase in the malware attacks towards Android, many research works have been proposed in the literature to detect Android malware. The detection mechanisms can broadly be categorized into two classes: Static Analysis and Dynamic Analysis. Static Analysis aims to investigate static components such as the Java code or the Manifest file segments of the application, without executing it. Ease of extraction of static features like permissions makes the static detection quite simple and inexpensive as the permissions can be simply extracted from the Android application which does not require execution of the application. However, stealthy malware samples have the capability to evade static detection by downloading malicious components at run-time.

For instance, Anserver Bot and Plankton are two such malware families which download malicious files at update time.

Therefore, researchers also focus on Dynamic Analysis [9], i.e., executing the application on an emulator or smartphone, to analyze its run time behavior. Features such as network traffic, system calls, and API calls are used in dynamic solutions [7]. However, dynamic solutions also suffer from additional computation overhead. Moreover, considering network traffic as the feature, not all malware samples are remotely controlled or connected by the server. Thus, network traffic feature cannot be used to detect all types of malware families. For instance, HippoSMS malware sends SMS in the background without any network connectivity. Hence, both types of techniques have their advantages and disadvantages.

Keeping the advantages of both the types of solutions in mind, in this paper, we have combined both the static and the dynamic features and proposed a hybrid detection model. We have used permissions and intents [8] as static features and network traffic as the dynamic one.

Several related works have been proposed in the literature that have used permissions, intents or network traffic features for Android malware detection. However, to the best of our knowledge, none of the existing works have combined all three features in a single hybrid model.

Contributions*:* The major contributions of the research are summarized below:

- We separately ranked the permissions, intents and the network traffic features according to the "Information Gain" (read as I.G.) which is based upon the concept of Entropy.
- We proposed an algorithm to find the best set of permissions that gives better detection results by applying Decision Trees and Naïve Bayes' classifiers.
- On similar lines, we used the same algorithm to find the best set of intents and the traffic features that gives better detection results.
- We then merged permissions with intents, permission with traffic features, and intents with traffic features to find the detection results.
- We further merged all the three features together and found that the hybrid approach of combining all the three features is best than any other combination of the features.

Organization: The rest of the research paper is structured as follows. Section II throws light on the related works in the field of Android malware detection. Section III depicts the methodology used in our research work. Section IV puts forward the experimental results of the Work. Finally, Section V concludes the proposed approach with furnishing the future work scope.

II. RELATED WORK

In this section, we review the studies that have been proposed for Android malware detection. We discuss the related works in four subsections: Permissions-Based Solutions, Intents-Based Solutions, Network Traffic-Based Solutions, and Hybrid Solutions.

A. Permissions-Based Solutions

Several related works exist that analyze the permissions to detect malicious samples. We further consider these related studies under two subsections: Permissions-Based Assessment and Permission-Based Detection.

Permissions-Based Assessment: Rahul et al. [11] proposed a model called WHYPER to analyze the set of permissions present in normal applications and to ascertain why that app utilizes that particular permission. The authors in [12] assessed the in-app ad libraries for the availability of precarious permissions. Their results depicted that perilous permissions were present in many in-app ad libraries which could summon the malicious code from the Internet. In [13], the authors developed a model for detection of over-privileged permissions in Android applications. Their model indicated that around 22% of the permissions are unnecessarily present and hence over-privileged. Enck et al. [14] designed a model called Kirin which provided permission rules to discern whether a normal application is behaving maliciously or not.

All these studies have analyzed the permissions within the apps to look for signs of suspicious behavior. However, none of them have focused to detect malicious samples.

Permissions Based Detection: Wang et al. [15] utilized the mechanisms of mutual information, T-Test and Correlation coefficient for ranking the dangerous and risky permissions. The authors in [16] and [17] proposed two detection models: PUMA and MAMA respectively. The PUMA model established the top permissions that had been defined in normal and malicious applications. Thereafter, machine learning algorithms have been used for the detection process. In MAMA model, the authors have extracted permissions and other features from the manifest file such as hardware components, created a feature vector of all the features, and then the machine learning algorithms were used in detection. The research work in [18] shows the development of the model named DroidRanger, that used permissions to detect malware samples. The authors inferred that few permission combinations were dangerous and thus can help detect malicious behavior such as (INTERNET, RECEIVE_SMS) and (SEND_SMS, RECEIVE_SMS).

All of these solutions used permissions alone for the detection purpose. Stealthy recent malware samples can evade permissions-based detection. Hence, compared to all these works, we have used both static and dynamic features in our detection model.

B. Intents-Based Solutions

The authors in [19] proposed a model named Droidmat, that analyzed intents and hardware components, in addition to the permissions, and further used k-means clustering algorithm for detection. Arp et al. [20] further extended the features-list by adding API calls and network addresses to the permissions and intents, to detect Android malware. On the similar lines the authors in [21] and [22] used permissions, intents, and strings to detect malicious samples.

All of these studies are static in nature and hence cannot detect all stealthy malware samples. On the contrary, our proposed work is hybrid in nature that can detect stealthy malware samples.

C. Network Traffic Based Solutions

In this section, we highlight some related studies that have used network traffic either for analysis or for detection of Android malware. We divide the related studies under two subsections: Network Traffic Induced Analysis and Network Traffic Induced Detection.

Network Traffic Induced Analysis:

This subsection scrutinizes the research works that have analysed the network traffic of Android malware samples and the normal mobile samples but have not discussed the detection approach for malicious Android samples.

Aresu et al. [23] used HTTP traffic analysis to cluster malware families into sets. The authors analyzed HTTP attributes such as the number of GET and POST requests, the quantity of data to be sent in POST communication and request method utilized in the network. The authors in [24] captured network traffic of apps and then assessed it for the first five minutes of capture, and thereafter they uncovered that nearly 70 percent of the samples produced malware traffic within the initial five minutes. They also unearthed some features to detect malicious samples. As per their research, HTTP request and DNS query are such features that can be used in this process of detection. The authors in [25] assessed the encrypted network traffic but they did not consider malicious samples in their assessment. Moreover, their work did not propound any detection approach for malicious Android apps.

Network Traffic Induced Detection:

This subsection highlights the research work that lay stress on detection of malicious android samples on the basis of network traffic features.

The researchers in [26] analyzed the precision of several machine learning techniques on network traffic for detection of malicious applications in Android devices. The authors delineated that kNN classifiers and Random Forest provide better detection precision than Naïve Bayes and Decision Tree Classifier when used with TCP-based attributes, HTTP-based attributes, DNS-based features, and Origin-Destination based attributes. The authors in [27] propounded an algorithm to sequence the network traffic features with an objective to curtail the number of features to be assessed to provide improved detection precision with lessened training and testing duration.

Wang et al. [28] accomplished assorted Network Traffic analysis with detection at TCP and HTTP levels. They considered TCP flow rooted attributes and HTTP Request attributes. The authors implemented the work using the Decision Tree algorithm and outlined the detection rate of 99% with HTTP based attributes. The detection rate with TCP based attributes was 98%. The researchers in [29] extricated features from HTTP header fields of malicious traffic in Android devices and further implemented clustering on those features. Their work showed the detection precision of around 90 percent. The authors illustrated the families of malware on the basis of the HTTP traffic data. Shabtai et al. [30] detected that few malicious applications had the potential to self-update and the authors used TCP traffic analysis to detect such samples. They also utilized machine learning techniques for the generation of traffic patterns and then discovered malicious applications. The traffic behavior of malicious apps from normal ones leads to their detection. Arora et al. [28] found 7 distinguishing features amongst 16 features but they captured the malicious traffic on the emulators rather than the actual smartphones.

All these network-based solutions suffer from one drawback, i.e., not every malware is remotely controlled and hence not all samples may produce network traffic. Therefore, such solutions can be used only for the subset of the malware samples.

D. Hybrid Solutions

Few solutions exist in the literature that have used hybrid attributes to detect Android malware. Kabakus et al. [32] analyzed permissions and the network traffic of the samples and observed that malware samples generate the lesser number of API calls with permissions as compared to the normal ones. The authors in [33] analyzed manifest file components, system calls, and the network traffic to detect malicious activity on smartphones. Riskranker model [31] analyzed static features like Java Code and permissions in addition to the dynamic ones like run-time Dalvik code loading to detect malware samples. The authors in [35] and [36] used permissions and network traffic to detect malicious samples.

To the best of our knowledge, no research work has focused on all the three parameters i.e. permissions, intents, and network traffic features to detect malware on Android smartphones. Therefore, in this research paper, we propose an adaptive malware detection mechanism using both the static and dynamic attributes simultaneously. This kind of solution will provide a better detection precision in Android smartphones.

III. METHODOLOGY

This section discusses the proposed hybrid approach that detects malicious samples based on permissions, intents, and network traffic. The proposed model consists of two phases: Training phase and Testing phase, as shown in Figure 1. We discuss both the phases in the subsequent sections.

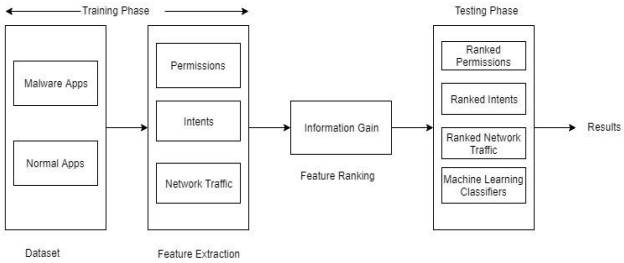

Figure 1: The Proposed Hybrid Model

A. Training Phase

This phase aims to find the distinguishing permissions, intents, and the traffic features by ranking them. We follow the below-mentioned steps to complete our training phase:

Permissions and Intents Extraction: We used "apk tool" to extract the manifest file from the applications. Further, we wrote Python scripts to extract the permissions and the intents defined in the manifest files.

Network Traffic Extraction: To capture the network traffic of the samples, we used an app named "tPacket Capture" available in the Google Play Store.

This app can capture the traffic of the Android smartphone in the "pcap" format that can be easily analyzed with the Wireshark software on the desktop machine. We have installed each application (normal or malware) on the Samsung Galaxy SIV smartphones for the period of three days to capture their corresponding traffic file. From the captured traffic files, we have extracted 20 traffic features as summarized in Table II for our experiments.

Table II: Traffic features used in Experiments

List of Traffic Features		
Average Packet Size	First Packet Size Sent	First Packet Size Received
Maximum Packet Size	Ratio of Incoming to Outgoing Bytes	Bytes Sent per Flow
Bytes Received per Flow	Bytes Sent per Second	Bytes Received per Second
Packets Sent per Flow	Packets Received per Flow	Packets Sent per Second
Packets Received per Second	Ratio of Incoming to Outgoing Packets	Average Time Interval Between Packets Sent
Average Time Interval Between Packets Received	Minimum Time Interval Between Packets Sent	Maximum Time Interval Between Packets Sent
Minimum Time Interval Between Packets Received	Maximum Time Interval Between Packets Received	

Features Ranking: We have used the concept of Information Gain to rank all the features separately, i.e., we have ranked the permissions, intents and the traffic features separately from each other and hence generated three different rankings. Information Gain finds its usage in the selection of the most suitable feature among a feature set according to their ranks. It provides a measure of curtailment of uncertainty. Suppose we have a training set T with features $f_1, f_2, \ldots . f_n$ and a class C. Then the uncertainty or entropy can be defined as:

$$H(C) = - \sum_I P(C_i)\log_2(P(C_i)). \quad (1)$$

Where $P(C_i)$ indicates the prior probability for all values of class C. The uncertainty of class C after focusing on feature f is calculated by the conditional entropy which is as follows:

$$H(C|f) = - \sum_j P(f_j) \sum_I P(C_i|f_j)\log_2(P(C_i|f_j)). \quad (2)$$

Where $P(C_i|f_j)$ shows the posterior probabilities of class C provided the value of feature f. Now the information Gain of feature f is as follows:

$$IG(C|f) = H(C) – H(C|f) \quad (3)$$

Higher the value of the Information Gain, it is more probable for the feature to be distinguishing between the malware and normal samples, and hence will have a higher rank. So at the end of this phase, we have got three different rankings, one each for permissions, intents, and the traffic features according to the Information Gain.

B. Testing Phase

In this phase, we explain our detection algorithm. From the previous phase, we get three ranked lists. The detection algorithm works on these lists as the input and gives the detection results. Algorithm 1 defines the working of the detection approach. We rename the list of ranked permissions be "P_List", list of ranked intents be "I_List", and list of ranked traffic features be "T_List". Let the number of features in "P_List", "I_List" and "T_List" be "P", "I" and "T" respectively. Then our aim is to find the best set of hybrid features, which we name as Best_Set, that gives the best detection results, which we name as Det_max. To achieve this, we have three loops working. To begin with, we select the top-ranked (first rank) permission, top-ranked (first rank) intent and top-ranked (first rank) traffic feature in our Best_Set. Then we run several machine learning classifiers like Decision Trees and Naïve Bayes algorithms on the testing data using the features in the set Best_Set and find the detection accuracy. Initially, the detection accuracy is initialized to zero. If the detection accuracy achieved is higher than the previous detection accuracy, then we update Det_max with the new detection accuracy. The procedure is repeated by adding the features sequentially in the Best_Set. Our aim is to find the maximum detection accuracy Det_max. The algorithm returns both, the Det_max and the Best_Set, i.e., the set of hybrid features on which are getting the best detection results. In the next section, we discuss the detection results obtained from the proposed approach.

Algorithm 1

1. Input: Three ranked lists of Permissions (P_List), Intents(I_List) and Traffic Features(T_List).
2. Output: Detection Accuracy
3. Det_max=0;
4. Best_Set=NIL;
5. Let number of permissions in P_List be denoted by P, number of intents in I_List be denoted by I, and number of traffic features in T_List be denoted by T.
6. for(i=1 -> P)
    a. for(j= 1-> I)
        i. for(k=1 -> T)
            1. Best_Set=P[i] U I[i]U T[i]
            2. Run machine learning classifiers on features in Best_Set and find the detection results Det_Acc.
            3. If Det_Acc > Det_max then Det_max <- Det_Acc.
7. Return Det_max and the Best_Set where we get the Det_max.

IV. RESULTS AND DISCUSSION

This section describes the results obtained from the proposed approach. We performed the experiments on the Windows machine with 12 GB RAM. We wrote Python scripts to extract the features and for machine learning classifiers. We have used malware datasets from Genome and Drebin. Further, we downloaded normal apps from the Google Play Store. We tested all the normal aps from the Virus Total, just to make sure that none of the normal apps downloaded from the Play Store is malicious in nature. Table III summarizes the number of malware and normal apps used in the experiments. We have used different set of applications in Training and Testing phase of our experiments.

Table III: Datasets for Experiments

	Malware Apps	Normal Apps
Training Phase	1000	800
Testing Phase	500	450
TOTAL	1500	1250

We analyze the results in different subsections. First we discuss the ranking obtained for the permissions, intents and the traffic features using Information Gain. Further, before discussing the results of the hybrid approach, we also analyze the results from the static and the dynamic approaches, i.e., we discuss the results obtained from the permissions alone, intents alone, and the traffic features alone. Then we discuss the results obtained by combining all the features.

Features Ranking Analysis: First we discuss the rankings obtained from the Information Gain. Table IV summarizes the ranking of permissions, intents and the traffic features. In total, we had 240 permissions, 801 intents and 20 traffic features. However, we have shown top 20 features only in the table below.

Table IV: Ranked Features

Ranked Permissions	Ranked Intents	Ranked Traffic Features
WRITE_SMS	DEFAULT	Maximum Packet Size
READ_SMS	BOOT_COMPLETED	Minimum Time Interval Between Packets Received
SEND_SMS	UMS_CONNECTED	Ratio Of Incoming To Outgoing Bytes
READ_PHONE_ STATE	BROWSABLE	Minimum Time Interval Between Packets Sent
WAKE_LOCK	VIEW	Average Packet Size
READ_EXTERNAL_ STORAGE	SIG_STR	Bytes Received Per Flow
CAMERA	UMS_ DISCONNECTED	Bytes Received Per Second
RECEIVE_SMS	SEND	Bytes Sent Per Second
WRITE_APN_ SETTINGS	SEARCH	Packets Received Per Flow
RESTART_ PACKAGES	QUICKBOOT_ POWERON	Packets Sent Per Second
USE_CREDENTIALS	ACTION_POWER_ DISCONNECTED	Packets Received Per Second
SYSTEM_ALERT_ WINDOW	TIME_SET	Packets Sent Per Flow
FOREGROUND_ SERVICE	BATTERY_ CHANGED ACTION	Ratio of Incoming to Outgoing Packets
CALL_PHONE	MY_PACKAGE_ REPLACED	Bytes Sent Per Flow
WRITE_CONTACTS	INPUT_METHOD_ CHANGED	Average Time Interval Between Packets Sent
GET_ACCOUNTS	TIME_ZONE_ CHANGED	Maximum Time Interval Between Packets Sent
MANAGE_ACCOUNTS	MEDIA_BUTTON	First Packet Size Received
BLUETOOTH	USER_ PRESENT	Maximum Time Interval Between Packets Received
MODIFY_AUDIO_ SETTINGS	LOCALE_ CHANGED	Average Time Interval Between Packets Received
RECORD_AUDIO	LEANBACK_ LAUNCHER	First Packet Size Sent

As can be seen from the table, WRITE_SMS is the highest ranked permission, which means that WRITE_SMS is the most distinguishing permission between malware and normal samples. Many malware samples try to send SMS in the background without the users' knowledge, hence permissions related to SMS are frequently found in the malware samples rather than normal ones. Similarly, we can interpret the rankings of other parameters. Higher the rank, more probable is the feature to be distinguishing between malware and normal samples.

Static Detection Results: Firstly, we analyze the results obtained only from the static features which are permissions and intents. Table V summarizes the results when we use permissions, intents and their combinations to get the detection results.

Table V: Static Detection Results

	Decision Tree		Naïve Bayes	
	Det_Max	**Best_Set**	**Det_Max**	**Best_Set**
Permissions Alone	90.2%	Top 120 permissions	89.3%	Top 50 permissions
Intents Alone	82.4%	Top 105 intents	81.2%	Top 65 intents
Permissions and Intents Combined	90.6%	Top 108 permissions and 90 intents	89.4%	Top 45 permissions and 30 intents

With permissions alone, we get the maximum accuracy of 90.2% with Decision Tree classifier with Best_Set consisting of 120 permissions. A similar analysis is given for intents. Further, we notice that we get the better accuracy of 90.6% when we combine the permissions and the intents. On analyzing the results, we found that certain malware samples evade this static detection because they download their malicious component at run time, for instance, AnserverBot. Hence, static detection is not sufficient alone to detect stealthy malware samples.

Dynamic Detection Results: On the similar lines, we performed the experiments only on the dynamic traffic features and observed the detection results. Table VI summarizes the detection results on network traffic feature alone.

Table VI: Dynamic Detection Results

	Decision Tree		Naïve Bayes	
	Det_Max	**Best_Set**	**Det_Max**	**Best_Set**
Traffic Features Alone	92.4%	Top 12 traffic features	91.2%	Top 10 traffic features

Again Decision Tree classifier was better than Naïve Bayes with dynamic features. However, with dynamic features we get relatively lesser accuracy than static results because some of the malware samples are not remotely controlled by any server and hence do not generate any network traffic. Therefore, we cannot extract network traffic features for such samples and hence these samples cannot be detected by traffic feature alone. Both of the solutions, static and dynamic, have some disadvantages. Therefore, we focus our approach towards hybrid detection.

Hybrid Detection Results: Finally, we combine all the static and dynamic features to form a hybrid set. Table VII summarizes the detection results with hybrid features.

Table VII: Hybrid Detection Results (Part-1)

	Decision Tree		Naïve Bayes	
	Det_Max	**Best_Set**	**Det_Max**	**Best_Set**
Permissions and Traffic Features	92.1%	Top 115 permissions and 12 Traffic features	91.6%	Top 50 Permissions and 12 traffic features
Intents and Traffic Features	91.2%	Top 102 intents and 12 traffic features	90.3%	Top 64 Intents and 12 traffic features
Permissions and Intents and Traffic Features	**93.6%**	Top 11 permissions, 86 intents and 12 Traffic features	93.1%	Top 46 Permissions, 50 intents and 12 traffic features

Table VIII: Hybrid Detection Results (Part-2)

	Neural Networks		SVM	
	Det_Max	**Best_Set**	**Det_Max**	**Best_Set**
Permissions and Traffic Features	93.1%	Top 112 permissions and 12 Traffic features	93.3%	Top 45 Permissions And 12 traffic Features
Intents and Traffic Features	92.4%	Top 98 intents and 12 traffic features	92.8%	Top 62 Intents and 12 traffic features
Permissions and Intents and Traffic Features	94.1%	Top 15 permissions, 80 intents and 12 Traffic Features	**94.2%**	Top 40 Permissions, 52 intents and 12 traffic features

Initially, we combine permissions with traffic features, and intents with traffic features, to form a hybrid set and find the detection results. The highest accuracy which we could get from any of this combination is 93.3% when we apply SVM on the combination of permissions and traffic features. However, on combining all the three features together, we observe the highest detection accuracy, i.e., 94.2% when we use SVM on all the three features together. Hybrid approach gives better accuracy than static and dynamic because it removes disadvantages of both the approaches. It can detect stealthy malware samples, undetected by the static approach. Moreover, it can also detect the malware samples that do not generate any network traffic, undetected by the dynamic approach. Hence, we can conclude two remarks: a) Hybrid approach is better than static and dynamic, and b) SVM classifier is giving better accuracy than other similar machine learning classifiers for our experiments.

V. CONCLUSION

This paper proposed a novel hybrid approach to detect Android malware by combining the static features of permissions and intents and the dynamic feature of network traffic. Initially, we ranked the features separately using the concept of Information Gain to obtain three different rankings: one each for permissions, intents, and traffic features. Further, we propose a novel detection algorithm that aims to combine all the features in such a way so that we can get the best detection results. We compared the detection results obtained with the hybrid features against the static and dynamic ones. The results indicated that the hybrid approach of combining all three features gives relatively better accuracy as compared to any other combination of the features. In our future work, we will try to integrate other static features such as hardware components, and dynamic API calls as well with the existing model to give a further extended hybrid approach.

REFERENCES

1. https://blog.globalwebindex.com/trends/device-usage 2019.
2. https://news.strategyanalytics.com/press-release/devices/strategy-analytics-global-smartphone-shipments-dip-4-percent-q1-2019.
3. https://www.statista.com/statistics/269044/worldwide-desktop-pc-shipments-forecast.
4. https://www.idc.com/getdoc.jsp?containerId=prUS45115119.
5. https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise
6. https://www.zdnet.com/article/mobile-malware-attacks-are-booming-in-2019-these-are-the-most-common-threats.
7. A. Feizollah, N. Anuar, R. Salleh, and A. Wahab, "A review on feature selection in mobile malware detection," Digital Investigation, vol. 13, pp. 22-37, 2015.
8. A. Feizollah, N. Anuar, R.Salleh, G. Suarez-Tangil, S. Furnell, "AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection," vol. 65, pp. 121-134, 2017.
9. P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. Gaur, M. Conti, and M. Rajarajan, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," IEEE Communications Surveys & Tutorials, vol. 17, pp. 998-1022, 2015.
10. Y. Zhou, and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, 2012.
11. M. Grace, W. Zhou, X. Jiang, and A. Sadeghi, "Unsafe exposure analysis of mobile in-app advertisements," Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks, 2012.
12. K. Au, Y. Zhou, Z. Huang, and D. Lie, "PScout: Analyzing the Android Permission Specification ," Proceedings of the 19th ACM conference on Computer and communications security, 2012.
13. W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," 16th ACM Conference on Computer and Communications Security, 2009.

14. W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and Zang, "Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection," IEEE Transactions on Information Forensics and Security, vol. 9 , pp. 1869-1882, 2014.
15. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. Bringas, and G. lvarez, "Puma: Permission usage to detect malware in android," International Joint Conference CISIS12-ICEUTE 12-SOCO 12 Special Sessions, Springer Berlin Heidelberg, 2013.
16. B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. Bringas, and G. Maran, "MAMA: manifest analysis for malware detection in android," Cybernetics and Systems, vol. 44, pp. 469-488, 2013.
17. D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, "Droidmat: Android malware detection through manifest and api calls tracing", in Seventh Asia Joint Conference on Information Security (Asia JCIS), 2012.
18. F. Idrees, M. Rajarajan, M. Conti, T. Chen, and Y. Rahulamathavan, "PIndroid: A novel Android malware detection system using ensemble learning methods", Computers & Security, 68, 36-46, 2017.
19. T. Kim, B. Kang, M. Rho, S. Sezer, and E. Im, A Multimodal Deep Learning Method for Android Malware Detection Using Various Features, IEEE Transactions on Information Forensics and Security, 14(3), 2019.
20. M. Aresu, D. Ariu, M. Ahmadi, D. Maiorca, G. Giacinto, Clustering android malware families by http traffic, in: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015, pp. 128–135.
21. Z. Chen, H. Han, Q. Yan, B. Yang, L. Peng, L. Zhang, J. Li, A first look at android malware traffic in first few minutes, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, 2015, pp. 206–213. doi:10.1109/Trustcom.2015.376.
22. M. Conti, L. V. Mancini, R. Spolaor, N. V. Verde, Analyzing android encrypted network traffic to identify user actions, IEEE Transactions on Information Forensics and Security 11 (1) (2016) 114–125. doi:10.1109/ TIFS.2015.2478741.
23. S. Garg, S. Peddoju, A. Sarje, Network-based detection of android malicious apps, International Journal of Information Security 16 (4) (2017)385–400. doi:10.1007/s10207-016-0343-z.
24. A. Arora, S. Peddoju, Minimizing Network Traffic Features for Android Malware detection, ICDCN, 2017.
25. S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, Z. Jia, Trafficav: An effective and explainable detection of mobile malware behavior using network traffic, in: 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), 2016.
26. L. Zhiqiang, S. Lichao, Y. Qiben, S. Witawas, C. Zhenxiang, DroidClassifier: Efficient Adaptive Mining of Application-Layer Header for Classifying Android Malware, Springer International Publishing, 2017.
27. A.Shabtai, L.T.Chekina, D.Mimran, L.Rokach, B.Shapira, Y.Elovici, Mobile malware detection through analysis of deviations in application network behavior, Computers & Security 43 (2014) 1–18.
28. A. Arora, S.Garg, S.K.Peddoju, "Malware detection using network traffic analysis in Android based mobile devices", 8th IEEE NGMAST, 2014.
29. A. Kabakus A. Talha, and I. Dogru, "An in-depth analysis of Android malware using hybrid techniques," Digital Investigation vol. 24, 2018.
30. M. Lindorfer et al, "Andrubis–1,000,000 apps later: A view on current Android malware behaviors," IEEE Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), Wroclaw, Poland, 2014
31. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, Riskranker: scalable and accurate zero-day android malware detection, 10th international conference on Mobile systems, applications, and services, 2012.
32. A. Arora, and S. Peddoju, NTPDroid: "A Hybrid Android Malware Detector Using Network Traffic and System Permissions" , 17th IEEE TrustCom, 2018.
33. V.Malik, S.Kumar Goyal, and N.malik, " Analysis of Android Malwares and Their Detection Techniques", Fourth IEEE International Conference On Parallel, Distributed and Grid Computing", 978-1-5090-3669-1/16/31 2016 IEEE.
34. V.Malik, S.Kumar Goyal, and N.Malik, " Mobile Malware Attacks And Security attributes", JETIR UGC Approved Journal, Vol.-6, Issue-6, June 2019.
35. V.Malik, and S.Kumar Goyal, " An Effective Detection Of Mobile Malware Behavior Using Network Traffic:TrafficAV", IJSDR Journal, Volume-4, Issue-6, June 2019.
36. V.Malik, and S. Kumar Goyal, Malware Detection in Smartphones Using Network Traffic Features, International Journal on Future Revolution In Computer Science & Communication Engineering, Vol. 4, Issue 3, March 2018.
37. A. Arora, S. Peddoju, V. Chauhan, and A. Chaudhary, Poster: Hybrid Android Malware Detection by Combining Supervised and Unsupervised Learning, 24th ACM MobiCom, 2018.

AUTHORS PROFILE

Vinisha Malik is a PhD Research Scholar in Department of Computer Science & Engineering, M.M Engineering College, Maharishi Markandeshwar (Deemed to be University) Mullana, Ambala, Haryana, India. Vinisha received M.Tech Degree in 2013 with Goldmedal from DCRUST Govt. University Murthal, Haryana. Vinisha is in Teaching and Research Development since 2013. She has published about 5 research papers in International Journals and refereed Conferences. She has also Industry work experience of two years in TCS Company. Her current research interests are in Mobile Malwares In Smartphones.

Dr. Sandip Kumar Goyal is working as Professor & Head, Computer Science & engineering Department , M.M Engineering college, Maharishi Markandeshwar (Deemed To Be University) Mullana, Ambala, Hrayana, India.Dr. Goyal received PhD from M.M University Mullana and M.Tech from Kurukshetra University, Kurukshetra. Dr. Goyal is in teaching and Research & Development from 2002. He has published about 45 Research papers in International, national Journals and Refereed International Conferences. His Current area of research is Load Balancing in Distributed system and simulation & Modeling.

Naveen Malik is an Assistant Professor, Computer Science & engineering Department , Dcrust Murthal, Sonepat, Haryana, India. Mr Malik Received M.Tech Degree from Kurukshetra University, Kurukshetra. Mr. Malik is in teaching and Research & Development from 2014. He has published about 14 Research papers in International, national Journals and Refereed International Conferences. His Current area of research is Semantic Web, Ontology & Software Engineering.