Active Permission Identification for Android Malware Application Detection

[1]M.Mohana, [2]Dr.S.M.Jagatheesan
[1]Research Scholar, [2]Associate Professor
Gobi Arts & Science College (Autonomous)

***Abstract* - Android malware is a term used to describe a group of malicious applications targeting Android smart-phones and tablets. Now-a-days widely applicable smart phone application adoption and rapid growth of contextually sensitive nature of smart-phone devices has lead to a renaissance in mobile application services and increased concerns over smart-phone malware. Most of the internet users are connected to internet for various purposes. Android Operating Systems are most commonly used systems in the smart-phones. Many applications are available in android play store and it is very difficult to distinguish and to discriminate between benign and malicious applications. Various malware detection tools have been developed, including system-level and network-level approaches. In this paper, we introduce Active Permission Identification (APID), a malware detection system based on permission usage analysis to cope with the rapid increase in the number of Android malware. Instead of extracting and analyzing all Android permissions, we develop three levels of pruning by mining the permission data to identify the most Active permission that can be effective in distinguishing between benign and malicious apps. APID then utilizes machine-learning-based classification methods to classify different families of malware and benign apps. This paper identifies dangerous permission list, benign permission list, shutdown list and reduce non-sensitive permissions and apply FS-SVM classification on the new data set. Our evaluation finds that only 11 permissions are active. APID is more effective by detecting 98% of malware in the dataset.**

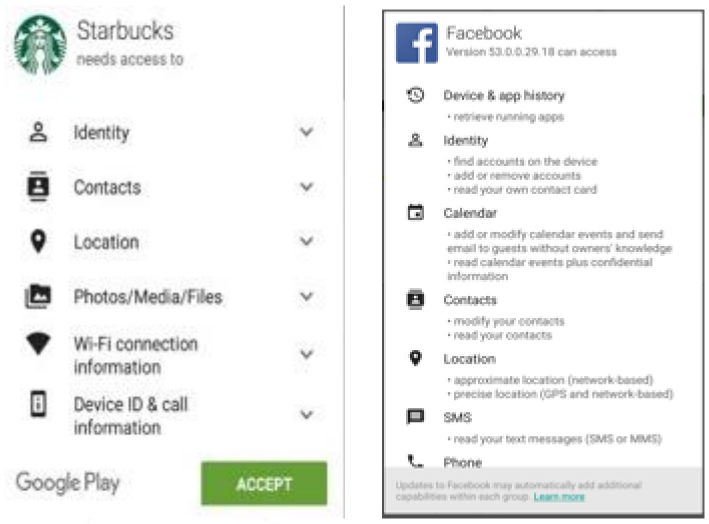
***keywords* - Android malware, Malware detection, Active permission, Classification, Machine Learning**

I. INTRODUCTION

ANDROID malware is the name used for malware infections that exclusively target Android-based devices. Android is currently the most used smart-mobile device platform in the world, occupying 86% of market share [1]. As of now, there are nearly 3 million apps available for downloading from Google Play and more than 65 billion downloads to data [2]. Android virus is an ever-evolving threat that infects millions of device for years. Google also identifies 24 permissions out of the total of more than 300 permissions as “dangerous” [3]. In 2016 alone, over 3.25 million new malicious Android apps have had been uncovered. This roughly translates to an introduction of a new malicious Android app every 10s [5]. These malicious applications are created to perform different types of attacks in the form of Trojans, worms, exploits, and viruses. Some notorious malicious apps have more than 50 variants, which makes it extremely challenging to detect them all [6]. A symptom of the android virus is surely a frustrating experience. Pushy ads signal that there’s something wrong with android devices. The official meaning of the word permission means allowing someone to do some other task. It is consent or authorization given to perform some kind of action. Android users will recognize permission from the long list of them typically presented in their product description at the play Store, and in the dialog presented to the user ensuring that “I Agree”, just before an application is installed. But other android operating systems have different methods of approaching and handling app permissions. Normal permissions are those which do not pose a risk to the user’s privacy or the device operation. The system grants this permission automatically. These includes connecting to the internet, getting Network, Bluetooth, Wi-Fi, and NFC information setting alarms & wallpapers, and modifying audio settings on a device. Dangerous permissions are permission which could potentially affect the user’s privacy or the device’s operation. The user must explicitly agree to grant that permission request. These include accessing the Camera, Contacts, Location, Microphone, Sensors, SMSs and Storage. The permission with the normal attributes is lower-risk (e.g., SET WALLPAPER) and will be automatically granted to without asking for the user’s explicit approval. The permission tagged as dangerous is more risk (e.g., READ SMS) that would access to private user data or control over the device. Most of the Android application permissions are self-explanatory, but here’s a breakdown of what each one actually means.

- *Body Sensors*-allows access to user health data and step count, from paired heart-rate monitors, fitness trackers, and other sensors.
- *Calendar*-allows apps to read, create, edit, or delete user calendar events.
- *Camera*-taking photos and recording video.
- *Contacts*-read, create, or edit user contact list, as well as access the list of all accounts used on user device.
- *Location*-access user location using GPS for high accuracy, and cellular data and Wi-Fi for approximate accuracy.
- *Microphone*-used for recording audio, including for video.
- *Phone*-access user phone number and network information. Required for making calls and VoIP, voicemail, call redirect, and editing call logs also some other activities.
- *SMS*- read, receive and send MMS and SMS messages.
- *Storage*-read and write files to user phone’s internal and external storage.

Anyone who's ever installed an application from Google Play has likely seen an app permission request. These requests pop up the first time an app needs access to sensitive hardware or data on user's phone or tablet. Here Fig 1.1 shows the some permission list during installing time.



Facebook
Version 53.0.0.29.18 can access

Device & app history
• retrieve running apps

Identity
• find accounts on the device
• add or remove accounts
• read your own contact card

Calendar
• add or modify calendar events and send email to guests without owners' knowledge
• read calendar events plus confidential information

Contacts
• modify your contacts
• read your contacts

Location
• approximate location (network-based)
• precise location (GPS and network-based)

SMS
• read your text messages (SMS or MMS)

Phone

Updates to Facebook may automatically add additional capabilities within each group. Learn more

Fig 1.1 Example of the Android Permission

Android Malware Types

While malware on Android hasn't quite reached the same scale as desktop malware, more mobile-specific malware designed to attack smart-phone features and vulnerabilities are emerging. Mobile malware on Android phones, or any devices for that matter, can be broken out into no fewer than main types. The important thing to remember about these categories is that many malware types do not fall into just one of them. When referring to a category, experts are referring to the malware's primary functionality. Here is some of the most popular malware classification today:

- *Adware* – shows frequent ads to a user in the form of pop-ups, sometimes leading to the unintended redirection of users to web pages or applications.
- *Banker malware* – attempts to steal user's bank credentials without their knowledge.
- *Ransomware* – it demands money from users and, in exchange, promises to release either the files or the functionality of the devices being 'held hostage'
- *Rooting malware* – 'roots' the devices, essentially unlocking the operating system and obtaining escalated privileges.
- *SMS malware* – manipulates devices to send and intercept text messages resulting in SMS charges. The user is usually not aware of this activity.
- *Spyware* – monitors and records information about user's actions on their devices without their knowledge or permission.
- *Trojan* –It hides itself within a piece of seemingly innocent, legitimate software.

We have designed in this system also focuses on Active Permission Identification (APID). The aim of APID systems is to achieve high malware detection accuracy and efficiency while analyzing the minimal number of permissions. It eliminates the need to analyze permissions that have little or no significant influence on malware detection effectiveness. In addition identification of dangerous, benign as well as shutdown enabled permission list is also carried out. Feature reduction is also carried out. FS-SVM classification for both all permission lists as well as feature reduced data set is included.

II. Related work

Existing research efforts have used Android permissions to detect malware apps. Huang et al. [7] explored the possibilities of detecting malicious applications based on permissions using machine learning mechanisms. They retrieved not only all the permissions, but also some easy-to-retrieve features from each .APK to help detect malware apps a. Four common machine-learning algorithms, AdaBoost, Naive Bayes, Decision Tree, and SVM [8], are employed in their system to evaluate the performance. Experimental results show that more than 80% of the malicious samples can be detected by the system. Because the precision is not high, their system can only be used as a first-level filter to help detect malware. A second stage of analysis is still needed for their system. Our proposed approach can be achieving a much higher detection rate compared to their work.

Arp *et al.* [9] has proposed DREBIN combines static analysis of permissions and APIs with machine learning to detect malware. They embedded features in a vector space, discovered patterns of malware from the vector space, and used these patterns to build the machine learning detection system. Their evaluation results indicate that their proposed work can achieve better detection accuracy. However, their analysis is performed on the devices and, therefore, requires that those smart-phones be rooted. They extracted as more features as possible to help improve high performance. However, our work only employs the significant permission identification features, which reduces the overhead of computation while retaining a satisfying accuracy result.

Wang *et al.* [10] explored the permission-induced risk in Android apps using data mining. They perform an analysis of individual permission and collaborative permissions and apply three type of ranking methods on the permission features. After the ranking step, they identify risky permission subsets using sequential forward selection and principal component analysis. They evaluate their approach using SVM, decision tree, and random forest techniques. The result shows that their strategy for identifying risky permissions can achieve a 94.62% detection rate with a 0.6% FPR. Their low FPR is brought by their use of a large benign datasets (80% of 310 926 benign apps) for model training, which incurs considerable overhead and produces skewed model.

Aung and Zaw *et al.*[11] propose a framework that intends to develop a machine learning-based malware detection system on Android to detect malware applications and to enhance security and privacy of Smartphone users. This system monitors various permission-based features and events obtained from the android apps, and analyses these features by using machine learning classifiers to classify whether the application is benign or malware.

Shabtai *et al.* [12] classifies two types of Android applications: tools and games. For their point of view, successful differentiation between games and tools is expected to provide certain indication about the ability of such methods to learn and model Android benign apps and potentially detect malware files with Machine Learning (ML) techniques on static features that are extracted from Androids app files. As such, the re-implementation of their method produces different results, our work needs less permissions compared to this work, but a high detection rate can still be achieved.

III. EXISTING SYSTEM

The existing system focuses on Significant Permission Identification (SIGPID), an associate approach that extracts important permissions from apps and uses the extracted information to effectively discover malware mistreatment supervised learning algorithms. The look objective of SIGPID is to notice malware with efficiency and accuracy. As expressed earlier, the quantity of recently introduced malware is growing at Associate in nursing frightful rate. As such, having the ability to discover malware expeditiously would permit analysts to be additional productive in distinctive and analyzing them. This approach analyzes permissions and so identifies solely those that are important in characteristic between malicious and benign apps. This includes a Multi Level Data Pruning (MLDP) approach together with Permission Ranking with Negative Rate (PRNR), Permission Mining with Association Rules (PMAR), and Support-based Permission Ranking (SPR) to extract vital permissions strategically [13].

Drawbacks

- SVM Classification isn't thought of so chance of benign/suspicious apps within the given new check knowledge isn't attainable.
- Feature reduction (based on distinctive values in permission list) before malware identification isn't administrated.
- Comparison between all the permission lists and have reduced permission list primarily based on SVM classification isn't enclosed.

IV. PROPOSED METHOD

In this section, we introduce our methodology for exploring Active Permission Identification (APID). The goal of APID systems is to achieve high malware detection accuracy and efficiency while analyzing the minimal number of permissions. It eliminates the need to analyze permissions that have little or no active influence on malware detection effectiveness. In addition identification of dangerous, benign as well as shutdown enabled permission list is also carried out. Feature reduction is also carried out. FS-SVM classification for both all permission lists as well as feature reduced data set is including. The complete system architecture of APID is shown in Figure 2.

Multilevel Data Pruning

A key component of APID is the multi-level data pruning process to reduce our permission dataset. Android apps can have up to 147 permissions in total. Most apps do not request all 142 permissions and the ones that an app requests are listed in the .APK as part of *Manifest.xml*. When we need to analyze a large number of apps, the total number of permissions requested by all apps can be overwhelmingly large, resulting in long analysis time. This high analysis overhead can skeptically affect the malware detection efficiency as it reduces analyst productivity. To address this problem, we use three levels of data pruning methods to filter out permissions that contributed little to the malware detection effectiveness. Thus, they can be safely removed without skeptically affecting malware detection accuracy.

Permission Ranking with Negative Rate (PRNR)

Every permission describes a particular operation that an app is allowed to perform. For Instance, permission INTERNET indicates whether the app has been access to the Internet. On the other hand, our focus is more on the permissions that create high-risk attack surfaces and are frequently requested by malware samples. Therefore, our pruning procedure identifies both types of highly differentiable permissions so that we can use this information to classify malicious and benign apps. At the same time, we exclude permissions that are commonly used by both benign and malware applications, as they introduce ambiguity in the malware detection process. Permission INTERNET is frequently requested by both malware and benign apps, as almost all apps will request to access the Internet. Therefore, our approach prunes permission INTERNET. To identify these two types of active permission, we design a permission ranking scheme to rank permission based on how they are used by malicious and benign apps.

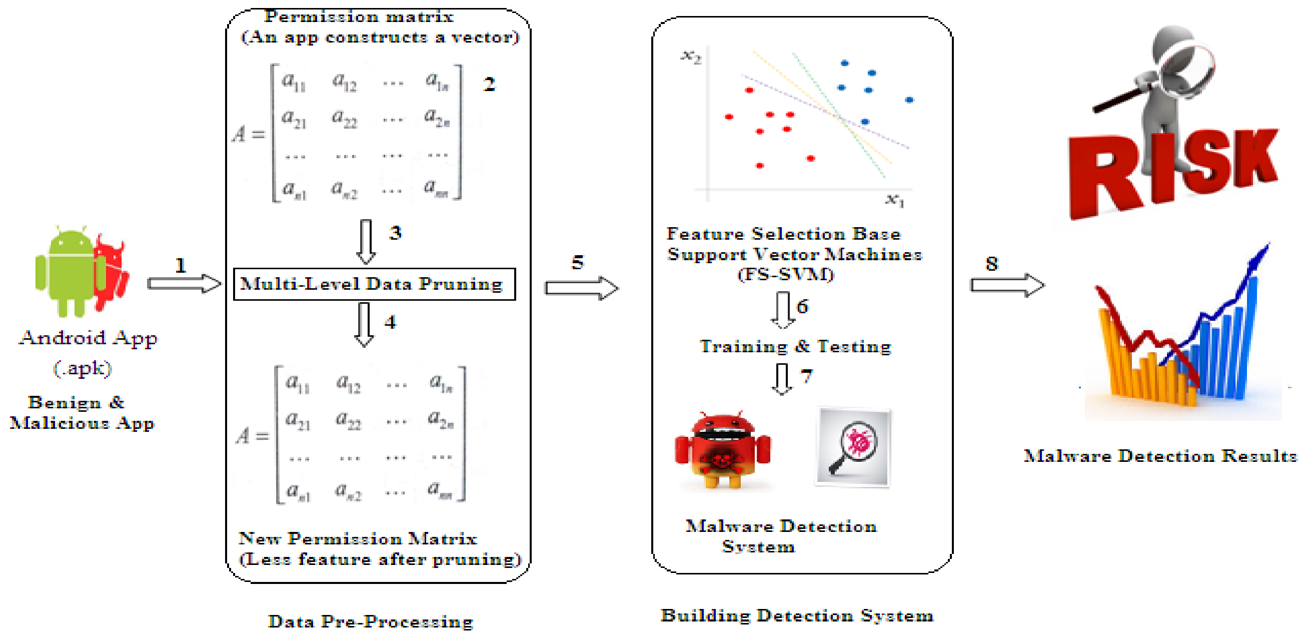


Figure 2 Architectural overview of APID

This section provides a concise ranking and comprehensible result. The approach operates on two matrices, M and B. M represents a list of permissions used by malware samples and B represents a list of permissions used by benign applications. Mij represents whether the jth permission is requested by the ith malware sample, while “1” indicates yes and“0” indicates no. Bij represents whether the jth permission is requested by the ith benign application sample. Before computing the support of permissions from matrices M and B, it first checks their sizes. Typically, the number of benign tends to be much larger than the number of malicious apps; therefore, the size of B is much larger than the size of M. With this ranking scheme, it prefers the dataset on the two matrices to be balanced. The PRNR algorithm is used to perform ranking of the datasets. In the formula above, R(Pj) represents the rate of the jth permission. The results of R(Pj) has a value ranging between [−1, 1]. If R (Pj) =1, this means that permission Pj is only used in the malicious dataset, which is a high-risk permission. If R(Pj) = −1, this means that permission Pj is only used in the benign dataset, which is a low-risk permission. If R(Pj)=0, this means that Pj has a very little impact on malware detection effectiveness.

IJEDR

Permission Mining with Association Rule (PMAR)

In this section, after pruning some permission by using PRNR and SPR (Support based Permission Ranking) with the PIS (Permission Incremental System), it can remove non-influential permissions even more. By inspecting the reduced permission list that contains some active permissions, it finds three pairs of permissions that always appear together in an app. For example, permission WRITE_SMS and permission READ_SMS are always used together. They also both belongs to the Google’s “dangerous” permission list. Yet, it is unnecessary to consider both permissions, as one of them is sufficient to characterize certain behaviors. As a result, we can associate one, which has been a higher support, to its partner. In this example, we can remove permission WRITE_SMS. In order to find permissions that occur together, it proposes a PMAR mechanism using the association rule mining algorithm. This section all the permissions are iterated in for loop and three columns are taken to find permission value ‘1’ along with next fourth column with permission value ‘1’. If the count of three columns values matched with count of fourth column then it is found out there is an association rule and printed out. The iteration continues for all 147 permissions.

Mutual Information

In this section, Mutual Information is found out as follows: Let X denotes a permission variable and C is the class variable. The relevance of X and C can be measured by mutual information of them as

$$\mathbf{I(X,C)} = \sum_{xi}\sum_{cj} P(X = xi, C = cj) \log \frac{P(X = xi, C = Cj)}{P(X = xi)P(C = Cj)}$$

Where P(C = cj) is the frequency count of class C with value cj, P(X = xi) is the frequency count of permission X with value xi, and P(X = xi,C = cj) is the frequency count of X with value xi in class cj. In this paper, the class C has binary values, c0 for benign apps and c1 for malicious apps. Each permissions X is a binary variable with value 1 or 0. I(X,

C) is nonnegative in [0, 1]. I(X, C) = 0 indicates no correlation, while I(X, C) = 1 means that C is fully inferable by knowing X.

Pearson Correlation co-efficient

In this section, Pearson Correlation co-efficient is found out as follows: Pearson Correlation Co-efficient measures the relevance of X and C by

$$\boxed{\mathbf{R(X,C)}\frac{\sum_{n=1}^{N}(Xn-\bar{X})(Cn-\bar{C})}{\sqrt{\sum_{n=1}^{N}(Xn-\bar{X})^2\ \sum_{n=1}^{N}(Cn-\bar{C})^2}}}$$

where X (resp. -C) is the average values of all the sample X (resp. C), X_n (resp. C_n), n = 1...N. R(X,C) has a value in [−1, 1], where R(X,C) = 0 indicates the independency of X and C, R(X,C) = 1 indicates the strongest positive correlation of them and R(X,C) = −1 indicates the strongest negative correlation. R(X, C) = 1 means that permission request of X makes applications higher risky, while R(X,C) = −1 means that permission request of X makes applications lower risky.

SVM Classification

In this section, SVM seeks the best hyper plane that separates data objects from one class on one side, while others on the other side. Here 70% of the data in given data set is taken as training data and 30% of the data is taken as test data. The model is trained with training data and then predicted with test data. Of which, most of the apps are classified as benign and fewer apps are classified as malicious apps.

Features Selection

In this section, each column values are taken and find the number of '1's and '0' and their percentage is calculated. If any one of the percentage is above 95%, then the column is treated as non-sensitive and can be eliminated. This section 70% of the data in given data set is taken as training data and 30% of the data is taken as test data but with the columns after feature reduction. The model is trained with training data and then predicted with test data. Of which, most of the apps are classified as benign and fewer apps are classified as malicious apps.

V. DATA SETS

We generate 10 datasets by randomly choosing 6334 benign apps from 16500 benign apps, downloading from Google play store. The requested permission list is built by extracting permission requests from each app listed in Android Manifest file. The total number of distinct permissions requested by all the apps (including benign and malicious) in our data sets is 142. However, the permissions requested by an application may be over-privileged, since 11 out of the 142 permission (e.g., permission INSTALL PACKAGES) are not for use by third-party applications [28]. The permission information is indicated into a binary format dataset, where "1" indicates that the app requests the permission, and "0" indicates the opposite side. The permission lists extracted from malicious apps and benign apps are combined to form a dataset for data analysis.

VI. EXPERIMENTAL RESULT

In this section we evaluate the malware detection effectiveness of the APID systems. We first use Multi-Level Data Tree Pruning method to rank the permissions and then secondly used to SVM and FS-SVM algorithm to evaluate the classification performance of our proposed MLDP model. We evaluate the malware detection performance by enabling these multiple levels sequentially to verify the performance improvement contributed by each level of the permission mining procedure. We use Mutual Information, MLDP methods for ranking the permissions their capability of discriminating malware apps from benign apps. As the permission ranking needs benign apps as well as malware apps, in the experiments, we use all the app data for permission ranking. The ranking results for the top 11 risky permissions are presented in Table I.

Top 11 Permission (MLDP)	Top 11 Permission (Mutual Information)
CAMERA	READ_SMS
CHANGE_NETWORK_STATE	WRITE_SMS
READ_CALL_LOG	RECEIVE_SMS
READ_CONTACTS	READ_PHONE_STATE
READ_EXTERNAL_STORAGE	READ_EXTERNAL_STORAGE
READ_PHONE_STATE	WRITE_CONTACTS
READ_SMS	VIBRATE
SEND_SMS	CHANGE_NETWORK_STATE
SYSTEM_ALERT_WINDOW	DEVICE_POWER
WRITE_CONTACTS	ADD_SYSTEM_SERVICE
WRITE_SETTINGS	ACCESS_NETWORK_STATE

Table 1 Ranking by MLDP & Mutual Information

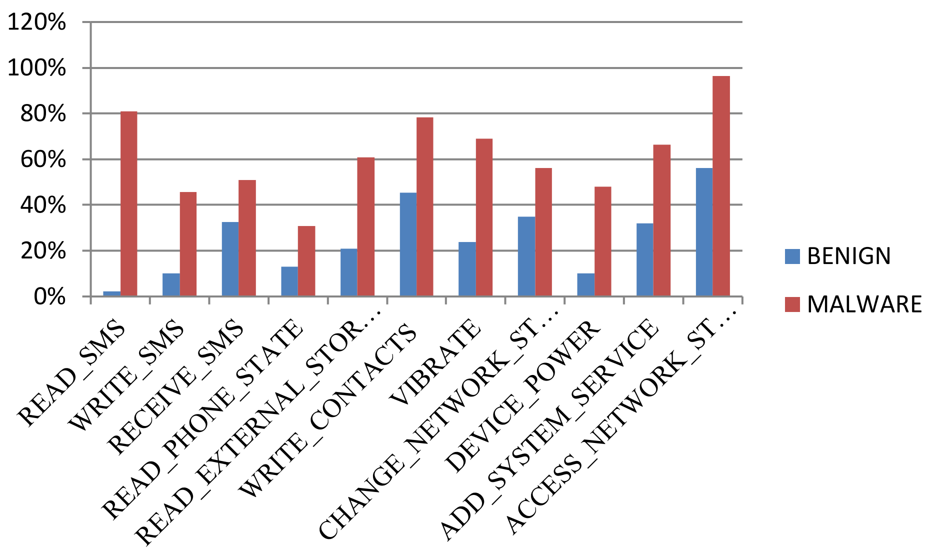


Figure 3 Top 11 Ranked Risky Permissions in Benign apps and Malware apps

The classification results on these different permission sets are compared on Accuracy (ACC) and F-score. Accuracy is the proportion of correctly classified data object (both True Positives (TP) and True Negatives (TN)) in the population (that is, the sum of TP, TN, as well as False Positive (FP) and False Negative (FN)):

$$ACC = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

F-score is a compromised measure between Precision and Recall, where Precision can be referred as Positive predictive value (defined as Precision = TP/TP+FP), and Recall is also referred to as the TPR (calculated by Recall = TP / TP+FN). It is defined as

$$F_{\_Score} = 2 \text{ x} \frac{\text{Precision . Recall}}{\text{Precision} + \text{Recall}}$$

F-score is usually used to evaluate the performance of unbalanced binary classification problem, like our malware apps detection problem involving a relatively small number of malware apps comparing to that of benign apps. It can be interpreted as a weighted average of the precision and recall. A F-score close to 1 indicates the good performance on correctly classifying the minority class, which in our case is the malware app.

Table 2 describes a confusion matrix for APL dataset recommendation mistreatment Malware Detection method mistreatment FS-SVM. The table contains a range of iteration, total APL dataset, results in frequent APL dataset count and non-predicted Malware App dataset details are shown.

Iteration	**Training APL Dataset (n)**		**Predicted Malware App Dataset (CR_{app}) (n)**		**Non-Predicted Malware App Dataset (n)**	
1	300		272		28	
2	600		583		17	
3	900		873		27	
4	1200		1173		27	
5	1500		1484		16	
6	1800		1790		10	
7	2100		2086		14	
8	2400		2382		18	
9	2700		2680		20	
10	3000		2989		11	
	TN or TR_{app}	**16500**	**TP (or) R_{app}**	**16312**	**FP**	**188**

Table 2 FS-SVM-Confusion Matrix for APL Dataset

Table 3 describes the Comparison Metrics Analysis for APL dataset using SVM & FS-SVM model. The table contains number of instances, number attribute selection with precision, recall, F-score and accuracy performances details are shown

Techniques	**Malware App Recommendation Dataset**		**Precision**	**Recall**	**F-measure**
	No. of Instances	No of Permission Level			
SVM	16500	22	0.974	0.978	0.975
FS-SVM	16500	11	0.975	0.988	0.980

Table 3 Comparison Metrics Analysis for SVM & FS-SVM Model Using APL Dataset

Figure 4 describes the Comparison Metrics Analysis for APL dataset mistreatment SVM & FS-SVM model. The figure contains a variety of instances, variety permission level choice with preciseness, recall and F-score performance details are shown.

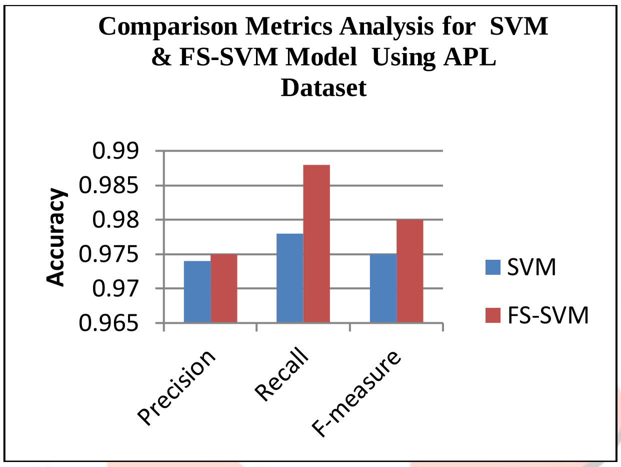


Figure 4 Comparison Metrics Analysis for SVM & FS-SVM Model Using APL Dataset

Table 4 contains a range of instances; range attributes choice accuracy performance details are shown.

Techniques	**Malware App Recommendation Dataset**		**Accuracy (%)**
	No. of Instances	No of Attributes	
SVM	16500	22	0.978
FS-SVM	16500	11	0.988

Table 4 Accuracy Analysis for SVM & FS-SVM Model Using APL Dataset

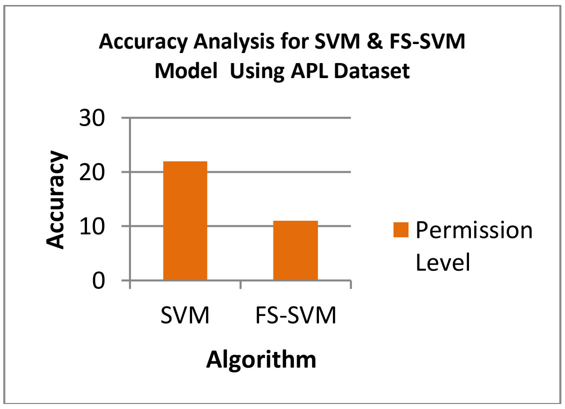


Figure 5 Accuracy Analyses for SVM & FS-SVM Model Using APL Dataset

Our evaluation finds 11 active permissions to the list of 24 harmful permissions identified by Google. [3] Therefore we believe our algorithm can retain more active permissions by pruning less important permissions compared with other permissions ranking methods. This allows our approach to identify more malicious applications in the dataset. APID is more effective by detecting 98% of malware in the dataset.

VII. CONCLUSION

In this paper, we have shown that it is possible to reduce the number of permissions to be analyzed for mobile malware detection, while maintaining high effectiveness and accuracy. APID has been designed to extract only Active permissions through a systematic three-level pruning approach. This proposed framework demonstrated how it is possible to reduce the number of permissions to be analyzed for mobile malware detection, while maintaining high effectiveness and accuracy. It has been designed to extract only significant permissions through a systematic three-level pruning approach. The existing system considers 22 permissions for malware apps but the proposed system analyzes 11 permissions are malware apps for the given data set. The difference is due to the non-sensitive permission features reduction. By adjusting the unique percentage in values of particular permission, the malware surety would be raised or lowered. There are several directions for future research. The current investigation of classification is still preliminary. Furthermore, the algorithm consistently outperformed all the tested classification and methods under different conditions. The future enhancements can be made with still less permission set and also use other techniques in future works.

REFERENCES

[1] IDC, “Smartphone OS market share, 2019 q1,” 2019.[Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[2] Statista, “Cumulative number of apps downloaded from the Google Play as of May 2019,” May 6, 2019. [Online]. Available: https://www. statista.com /statistics/266210/number-of-available-applications-in-the-google-play-store/

[3] G.Android, “Requestiong permissions.” 2017. [Online].Available: https://developer.android.com/guide /topics/permissions/ requestiong.html

[4] Wu, S. Schwab, and R. L. Peckham, “Signature based network intrusion detection system and method,” 2008, US Patent 7,424,744.

[5] G.DATA,“8,400 new android malware samples every day.”2017. [Online].Available: https:/ / www.gdatasoftware.com /blog/2017/04/29712-8-400-new-android malware-samples-every-day

[6] Symantec, “Latest intelligence for March 2016,” in *Symantec Official Blog*, 2016.

[7] C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, “Performance evaluation on permission-based detection for android malware,” in *Advances in Intelligent Systems and Applications*, vol. 2. New York, NY, USA: Springer, 2013, pp. 111–120.

[8] Jindal, A. Dua, K. Kaur,M. Singh, N. Kumar, and S.Mishra, “Decision tree and SVM-based data analytics for theft detection in smart grid,” *IEEE Trans. Ind. Informat.*, vol. 12, no. 3, pp. 1005–1016, Jun. 2016.

[9] Arp, M. Spreitzenbarth, M. H¨ubner, H. Gascon, K. Rieck, and C. Siemens, “DREBIN: Effective and explainable detection of android malware in your pocket,” presented at Annu. Symp. Netw. Distrib. Syst. Security, 2014.

[10] Wang. W, Wang. X, Feng. D, Liu. J, Han. Z, and X. Zhang, “Exploring permission-induced risk in android applications for malicious application detection,” *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 11, pp. 1869–1882, Nov. 2014.

[11] Z. Aung and W.Zaw, “Permission-based android malware detection”, international journal of Scientific & Technology Research vol. 2, no. 3, 2013.

[12] Y. Shabtai ,Fledel and Y.Elovici, Automated static code analysis for classifying Android application using machine learning”, in computational Intelligence and security (CIS), 2010 International Conference on app,329-333, IEEE-2010.

[13] J. Li., L. Sun, Q. Yan, Z. Li, Srisa-an, W. and Ye, H. “*Significant Permission Identification for Machine Learning Based Android Malware Detection*”. IEEE Transactions on Industrial Informatics, 2018.