International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Malign detection based on permissions identification and pruning system using ML approach

Dr. Bharathi M[1], Harshavardhan J[2], Harshith P[3] , Koduru Siva Krishna[4], Mithun A[5]

[1]*Associate Professor, Dept. of CSE, S J C Institute of Technology, Chickballapura, Karnataka, India*
[2]*Student, Dept of CSE, S J C Institute of Technology, Chickballapura, Karnataka, India*
[3]*Student, Dept of CSE, S J C Institute of Technology, Chickballapura, Karnataka, India*
[4]*Student, Dept of CSE, S J C Institute of Technology, Chickballapura, Karnataka, India*
[5]*Student, Dept of CSE, S J C Institute of Technology, Chickballapura, Karnataka, India*

Abstract - *Android is most widely used platform by the users all around the world. Android Platform led to the immense growth of mobile apps both benign and malign apps. As we all know Google Play is the most trusted Distributor of Android apps. Even Google Play is failing to detect malign apps .With the immense use of Smartphone for accessing online services, users store general, and confidential information on mobile device. Availability and Accessibility of sensitive information has encouraged cyber criminals to use features of smartphones for cyber attacks. Numerous malware detection tools have been developed, including system-level and network level approaches. However, scaling the detection for a large bundle of apps remains a challenging task. In this paper based on permissions asked by the apps during runtime, we will detect and will allow users to disable such permissions. Based on the behavioral pattern of permissions asked by app we classify it as an benign or malign. The Proposed System initially prunes the number of permissions to be analyzed in the three levels and then classification of apps as benign or malign is done by Support Vector Machine(SVM) Algorithm. After the Classification of apps as benign or malign the results are stored in a well maintained database for future reference.*

Key Words: Benign, Malign, Support Vector Machine(SVM).

1.INTRODUCTION

Malware can simply mentioned as an software, or piece of a code which causes damage to the personal computer system which it resides on. There are five styles of malwares: viruses, worms, trojan horses, spyware. Viruses damage their target computer by corrupting data, reformatting their fixed disk, or completely shut their system down. They also steal information, affect computers and networks, create botnets, steal money, render advertisements etc. viruses copy them self and spread to other computers by attaching themselves with programs and executing code when a user runs the infected program. An virus requires human activity to spread to other computers and are usually spread through email attachments and internet downloads. worms usually are spreads through computer networks by exploiting package weaknesses. it's an standalone program or piece of code that has the capacity to duplicate itself to infect other computers, without requiring action from anyone. worms can spread fast, worms are mostly used for executing a payload – a chunk of code that has the capacity to cause damage to the system. Payloads has power to delete files on a bunch system, encrypt data for a ransomware attack, steal information, delete files, and make botnets. Trojan tries enter host system disguised form as an normal file, harmless file or program so as to trick users and make them download and install the malware, once we install a Trojan, we are giving cyber criminals access to our system, doing this permits the cyber criminal to steal data, install few more malware, modify files, monitors the user activity, destroy data, steal financial information, conduct denial of service (DoS) attacks on targeted web addresses, and etc. Trojan malware don't have the flexibility duplicate by itself, however, if its combined with a worm, the damage Trojans can cause to users and systems is limitless. Spyware is additionally an variety of malware that Installed on our computer without our own knowledge, spyware is especially designed to trace our browsing habits and internet activity. Capabilities of spying include activities like monitoring, collecting data of keystrokes, harvesting data of account information, logins, and financial data, and etc. Spywares are spread by exploiting software vulnerabilities, bundling them with an legitimate software, or within the Trojans.In our Paper we Analyse the permissions asked by the apps and classify them as benign or

malign. Generally these type permissions requested crop up the primary time an app needs access to sensitive hardware or data on your phone or tablet. If you've installed a camera app, for instance, it'll need your permission to access the camera before it can actually take photos. So, additionally to being cautious about the apps you put in from google play, its also important to grasp which permissions those apps request from you.
System permissions are divided into two groups namely normal and dangerous. Normal permissions asked by apps are allowed by default, because they don't cause risk to privacy .Dangerous permissions asked by apps, however, give apps access to things like calling history of user, private messages of user, location of user, camera of user, microphone of user, and etc. Therefore, Android will always ask us to approve dangerous permissions. we either used to allow all permissions an app needed to function before installation or we declined them all, which meant you couldn't install the app. Apps need access to content on our phone to fulfil their functionality ,a picture-editing app asks access of phone camera and media files in order to edit pictures saved in your phone. Permissions alone are harmless and are useful to produce users a decent mobile experience. But since the list of permissions required is long and doesn't explain its effect, a right away reaction is to treat it the way you'd a 'Terms and conditions' agreement accept without reading the list and move to the subsequent step.

2.LITERATURE SURVEY

Many of the Works been carried out in knowing about the Malign apps and their behavior and also different techniques have been used to detect malign apps.

In 2012, M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang utilized static analysis to discover malicious behaviors in Android apps. However, static analysis approaches generally assume more behaviors are possible than would be ,which may lead to a large number of false Positives[1].

In 2014, W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth utilized dynamic analysis approach to improve the accuracy, researchers proposed the dynamic analysis approach to capture real-time execution context. TAINTDROID Dynamically tracks multiple sensitive data source simultaneously using tainting analysis[2].

In 2014, D. Arp, M. Spreitzenbarth, M. H¨ubner, H. Gascon, K. Rieck, and C. Siemens utilized both static analysis approach as well as dynamic analysis approach and Machine learning Techniques to detect Android Malware. The Experimental Result has high detection Accuracy by incorporating as many features as Possible to help detection[3].

In 2015, S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti Proposed TextDroid, an effective and automated malware detection method combining natural language processing and machine learning. TextDroid is able to extract differential features to classify malware samples. A , malware detection model is then developed to detect mobile malware using Support Vector Machine(SVM)[4].

In 2018,Sonali Kothari Tidke, Pravin P Karde, Vilas Thakare ,provided a solution that will detect harmful permissions and will allow user to disable such permissions. To demonstrate the same , a sample malware attacks is created[5].

In 2016, Z. Li, L. Sun, Q. Yan, W. Srisa-an, and Z. Chen, in this they used an approach that uses network traffic analysis to build many or various models in an automated fashion using a supervised method over a set of labeled malware network traffic. Each and every model is constructed by extracting common identifiers from various HTTP header fields. Clustering is used for improving the level classification accuracy[6].

3. METHODOLOGY

Our Proposed System has two stages:(i)Data Pruning (ii)Machine-Learning based Malware Detection

(i)Data Pruning :The first component of our system is the data pruning process to identify only important permissions to eliminate the need of considering all available permissions in Android. This stage further has three sub stages. The complete three sub-stages procedure is illustrated in Figure below:

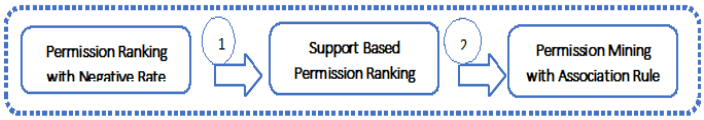

Fig.1:Data Pruning

1) Permission Ranking with Negative Rate (PRNR): This uses two matrices, M and B. M is list of permissions employed by malware and B may be a list of permissions employed by benign apps. Mij where jth permission is requested by the ith malware ,'1' indicates yes, '0' indicates no. Bij where jth permission is requested by the ith benign app . Note that the dimensions of B are often much larger than the dimensions of M. With our ranking scheme, we prefer the info assail the 2 matrices to be balanced. Training over imbalanced dataset can cause skewed models. To balance the 2 matrices, we use Equation 1 to calculate the support of every permission within the larger dataset so proportionally scales down the corresponding support to match that of the smaller dataset. just in case that the amount of rows of B is greater than that of M, we have:

$$S_B(P_j) = \frac{\sum_i B_{ij}}{size(B_j)} * size(M_j), \qquad (1)$$

Pj is that the jth permission, and SB(Pj) is that the support of jth permission in matrix B.
Permission ranking is implemented using Equation 2:

$$R(P_j) = \frac{\sum_i M_{ij} - S_B(P_j)}{\sum_i M_{ij} + S_B(P_j)}$$

This algorithm is employed to perform ranking of our datasets. within the formula above, R(Pj) represents the speed of jth permission. The results of R(Pj) contains a value ranging between [-1, 1]. If R(Pj) = 1, this suggests that permission Pj is merely utilized in malicious dataset, which may be a high risk permission. If R(Pj) = -1, this suggests that permission Pj is merely utilized in benign dataset which may be a low risk permission. If R(Pj) = 0, this suggests that Pj has little impact on malware detection effectiveness. Since both -1 and 1 are important, we simply take absolutely the value of every number and therefore the results of |R(Pj)|ranges between [0, 1]. We then evaluate malware detection by using the subsequent metrics precision, recall(true positive rate), false positive rate, accuracy, and F-measure. Next, we decide the highest three permissions in both lists to create malware detection. After This, we repeat the method with increase within the number of permissions to use for malware detection until the detection metrics plateau. the most goal is to seek out the tiniest number of permissions that yields a really similar malware detection effectiveness as that of using the whole data set.

2) Support based Permission Ranking: To further reduce the quantity of permissions, we turn our focus to the support of every permission. Typically, if the support of a permission is simply too low, it doesn't have much impact on malware detection. as an example, we find the permission INTERNET only in benign apps.

Malware (M_{ij})

	M_1	M_2
INTERNET	1	1
WRITE_SMS	1	1
READ_LOGS	1	1
CAMERA	1	0
VIBRATE	0	0
READ_SMS	1	1
RECORD_AUDIO	0	1
READ_CALENDAR	0	0

Benign (B_{ij})

	B_1	B_2
INTERNET	1	1
WRITE_SMS	0	0
READ_LOGS	0	0
CAMERA	1	0
VIBRATE	1	1
READ_SMS	0	0
RECORD_AUDIO	1	1
READ_CALENDAR	1	0

Fig 2:Matrix Represenation of Permissions

As such, we might imagine that any app that uses INTERNET is benign. However, this permissions is employed only by one app out of over 310,926 benign apps. As such, only hoping on the speed provided by Permission Ranking with Negative Rate is inaccurate. We also must prune out permissions with low support.

Apriori Algorithm:

To find out permissions that occur together, we propose a permission mining with association rules (PMAR) mechanism using association rule mining algorithm. Association rule mining has been used for discovering meaningful relations between variables in large databases. For instance, if event A and B always co-occur, it is highly likely that these two events are associated. In this paper, we only consider rules with high confidence, so that applying PMAR will only produce a small number of rules. We employ Apriori , a commonly used association mining algorithm, to generate the association rules. Apriori uses a breadth-first search strategy to count the support of item sets and uses a candidate generation process, which exploits the downward closure property of the support. Here, we only want to generate the association rules with high confidence even if the permissions have small support values.

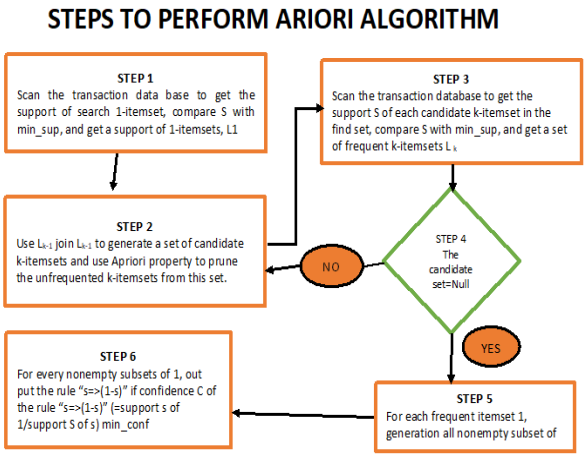

Fig.3:Apriori Algorithm

3) Permission Mining with Association Rules: After pruning permissions by using Permission Ranking with Negative Rate and Support based permission ranking, we want to further explore approaches that can reduce non-influential permissions. By inspecting the reduced permission list that contains obvious permissions, we find 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 both also belong to the "dangerous" permission list provided by Google. As such, we can associate one, which has a higher support, to its partner. In this example, we can remove permission WRITE SMS. In order to find permissions that occur together, we apply permission mining with association rules . In all, we are able to remove three additional permissions, giving us permissions that we consider as obvious.

(ii) Machine-Learning based Malware Detection: We first use SVM and a tiny low dataset to check our proposed MLDP model. SVM determines a hyperplane that separates both classes with a maximal margin supported the training dataset that has benign and malicious applications. during this case, one class is related to malware, and therefore the other class is related to benign apps. Then, we assume the testing data as unknown apps, which are classified by mapping the info to the vector space to make your mind up whether it's on the malicious or benign side of the hyperplane. Then, we are able to compare all analysis results with their original records to judge the malware detection correctness of the proposed model by using SVM.

Support Vector Machine(SVM) Algorithm:

Support Vector Machine (SVM) could be a supervised machine learning Algorithm which might be used for both classification or regression challenges. during this we use SVM and a tiny low dataset to check our pruned permission Data sets. SVM determines a hyperplane that separates both classes with a margin supported the training dataset that features non-malicious and malicious applications. during this case, one class is related to malicious, and therefore the other class is related to non-malicious application. Then, we assume the testing data as unknown apps, which are classified by mapping the information to the vector space to make a decision whether it's on the malicious or benign side of the

hyperplane. Then, we will compare all analysis results with their original records to gauge the malware detection correctness of the proposed model by using SVM.

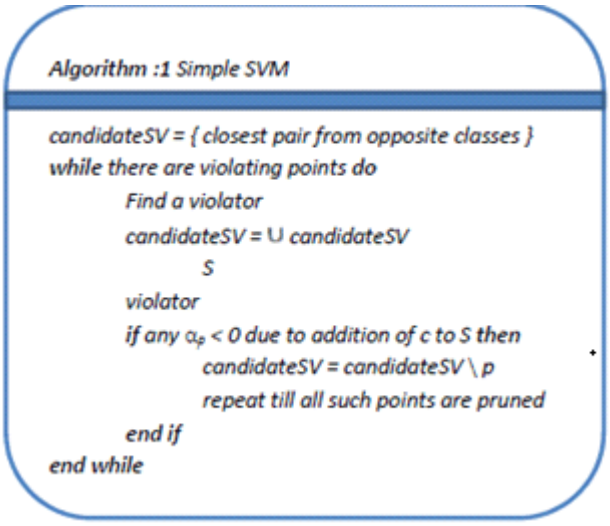

Fig.4:SVM Algorithm Pseudocode

Comparison with Other Approaches:

In this section, we compare our detection results with other state-of-the-art malware detection approaches, listed as follows: DREBIN[3] is an approach that uses static analysis to form data set supported permissions and other features from apps. After this we use Support Vector Machine(SVM) algorithm to classify malware dataset. We did not reimplement their approach since it requires significant program analysis additionally to permission analysis, we compare our results with the already reported results. PERMISSION-INDUCED RISK MALWARE DETECTION[8] is an approach that applies permission ranking, like mutual information. They use the permission ranking and choose the very best 40 risky permissions for malware detection. We reimplemented their approach for comparison. Note that in their paper, they used a special data set and thus the ratio of their malicious and benign apps in their dataset is dominated by benign apps. As such, their reported results, especially false positive rate, are significantly different than the results achieved using our data set.

The comparison results are shown in Table I. DREBIN uses more features than our approach, including API calls and network addresses. As a result, DREBIN is healthier than PERMISSIONCLASSIFIER in detection accuracy. We compared the results against 10 currently used anti-virus scanners . once we combine our approach with FT, we are ready to achieve the highest detection rate (93.62%) using only 22 permissions. Discussion: when we compared results of our work with the opposite approaches that consider only risky permissions, our approach considers a criteria that also include non-risky permissions , which are only employed in benign apps and have high support values. We deem the risky and non-risky permissions with high support values as significant permissions, allowing our approach to be simpler in distinguishing between malicious and benign apps than other existing approaches. We noticed that the permission lists employed by DREBIN contain various meaningless features, we can achieve performance improvements are by combining our approach with FT into DREBIN to reinforce both malware detection accuracy and period performance. we'll explore this integration in our future work, even though we consider less number of permissions, our approach performance is healthier than most of currently used malware scanner today, this will be because most of these techniques depend on signature matching ;so if a method of malware signatures isn't available, the system wouldn't be able to detect that specific type. We also show that our approach is easier than DREBIN once we combine our permission pruning with FT. DREBIN could also be a more complex malware detection approach that also uses static program analysis. We try to also explore a mix of using static program analysis with our approach to assess whether we are ready to achieve higher detection effectiveness.

4.CONCLUSION

In this approach, 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. Our approach has been designed to extract only significant permissions through a systematic,3-level pruning approach. Based on the dataset we took, it includes over 2,000 malware, we only need to consider 22 out of 135 permissions to improve the runtime performance by 85.6% while achieving over 90% detection accuracy. The extracted significant permissions can also be used by other commonly used supervised learning algorithms to yield the F-measure of at least 85% in

55 out of 67 tested algorithms. Our approach is highly effective, when compared to the state-of-the-art malware detection approaches as well as existing virus scanners. It can detect 93.62% of malware in the data set, and 91.4% unknown/new malware.

REFERENCES

[1] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012, pp. 281–294.

[2] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an informationflow tracking system for realtime privacy monitoring on smartphones," ACM Transactions on Computer Systems (TOCS), vol. 32, no. 2, p. 5, 2014.

[3] D. Arp, M. Spreitzenbarth, M. H¨ubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS), 2014.

[4] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Textdroid: Semantics-based detection of mobile malware using network flows."

[5] Sonali Kothari Tidke, Pravin P Karde, Vilas Thakare,"Detection and Prevention of Android Malware thru Permission Analysis.

[6] Z. Li, L. Sun, Q. Yan, W. Srisa-an, and Z. Chen, "Droidclassifier: Efficient adaptive mining of application-layer header for classifying android malware," in International Conference on Security and Privacy in Communication Systems. Springer, 2016, pp. 597–616.

[7] SigPID:Significant Permission Identification for Android Malware Detection Lichao Sun, Zhiqiang Li, Qiben Yan, Witawas Srisa-an and Yu Pan University of Nebraska–Lincoln Lincoln, NE 68588 {lsun,zli,qyan,witty,ypan}@cse.unl.edu.

[8] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-inducedriskinandroidapplicationsformaliciousapplication detection," Information Forensics and Security, IEEE Transactions on, vol. 9, no. 11, pp. 1869–1882, 2014.