IOP Conference Series: Materials Science and Engineering

PAPER • OPEN ACCESS

Analyzing Various Machine Learning Algorithms for the Classification of Malwares

To cite this article: J Pavithra and J S Femilda Josephin 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **993** 012099

ANALYZING VARIOUS MACHINE LEARNING ALGORITHMS FOR THE CLASSIFICATION OF MALWARES

J Pavithra[1] and Femilda Josephin J S[2]

[1] School of Computing, Dept of Computer Science and Engineering, SRM Institute of Science and Technology, SRM University, Kattankulathur-603203, Chennai, India.

[2] School of Computing, Dept of Software Engineering, SRM Institute of Science and Technology, SRM University, Kattankulathur-603203, Chennai, India.

E-mail: pavithrj@srmist.edu.in

Abstract. Malware, brief for Malicious Software, is increasing continuously in amounts and sophistication as our digital world continues to develop. Lately, tools for forming malware have been increasing rapidly on the internet, making it more accessible for people without expertise to create malware. Towards the end, the number of malware is growing fast. To deal with the problem, it is necessary to classify malware instantly and accurately. These malware programs play a role such as encrypting or destroying sensible data, stealing, changing or capturing main computing functions, controlling users and perform computer activity without their consent. In this paper, six different algorithms for classification like Linear Regression, RandomForest, Adaboost, Gaussian, Gradient Boosting, Decision Tree has been used for classifying a file as malicious or benign. Based on the results, the Random forest attained 99.43% accuracy and infers that it is suitable for malware prediction.

Keywords: Machine learning, Malware classification, malicious software, Prediction, Accuracy.

1. INTRODUCTION:

The current world is quickly running towards digitization. The technology has increased a great deal of significance in our daily life to manage numerous perspectives like business reason, training, and so forth. The term “Malware” denotes a malicious program. It defines opposed software programs and applications [1], it has formed into the most critical danger to the digital framework, earliest in the 1960s. Malware is a program that has intended to hold unwanted or impacts on a computer, and it has enhanced a critical cause to the computer security systems. With the innovative progression, the malware creators have generated destructive programs that are difficult to decrypt and distinguish by the researchers. At the same time, malware journalists made destructive code by executing a new procedure transformation trademark on that malware causes a tremendous development on malware.

Since the quantity of malware programming quickly increasing [2], antivirus organizations are consistently seeking methods that are best in recognizing malware. An antivirus organization utilized the most mainstream technique is Signature-based recognition. Not opposing, but the conventional malware identification methodologies are not suitable to report the hidden malware, distinguishes variations in malware that is recently recognized.

Several attempts have made to recognize malware. A few techniques have been utilized in many review papers [3]. There are various sorts of malware recognition and characterization strategies like static, dynamic, and hybrid characteristics [1]. The static review involves code examination [4] without executing malware by looking at and viewed programming code to accumulate data on how malware's positions work. There is an extraordinary method for without using the codes yet as per the runtime conduct by following its performance, framework connection, and impacts on a framework called dynamic review [5]. The hybrid review [6] is a blend of static and dynamic analysis.

2. LITERATURE SURVEY:

The antivirus association is facing the initial problem as detecting malicious code which is unseen at an earlier stage. To overcome this problem, many detection approaches have existed. Many detection approaches use data mining and machine learning algorithms. Because it self-learn patterns from the training data, which contains both benign and malware applications.

In paper[7], A proposed method is to analyze text hidden in images on spam emails rather in general emails through the OCR. This problem has been rectified by examining spam messages.

By using API calls, [8] a binary feature technique for malware location and arrangement has been proposed, researchers additionally examined the recurrence on the same dataset with the same information, but no improvement over on a binary feature. In comparative methodology, [9] Malware location dependent on Application programming interface calls and their contentions had been proposed. This procedure is used as a component by the researchers and dissected their outputs on characterizations. The feature selection method reduces the number of features. The result from the test assessment displays a precision of 98.4% in the fittest by using a random forest.

A **parallel ML-based classification method** was proposed for the initial stage detection of android malware. Based on real malware and benign applications, the parallel composition of diverse classifiers designed the composite model[10][11]. [12] researchers implied that deep learning models are performing better for the long series of system call analysis.

[13] proposed an actual PE malware identification framework for the review of data saved in the portable executable-optional header files. [14] author recommended another procedure for the classification of malware utilizing static analysis dependent on control observation. The authors applied Phi-coefficient and chi-square from feature selection for training and testing with chosen features random classifier. The authors attained 97% precision. In their work, 10,260 malware cases used from the dataset, and authors attained up to 99.21% exactness by utilizing stripped malware binaries as excerpted features.

In[15], researchers compared a few machine learning algorithms for malware detection. Decision Tree (DT), K-Nearest Neighbors (K-NN), and SVM algorithms were used for malware prediction by utilizing portable executable file data. The authors confirmed that decision tress performs better in ML algorithms to identify and classify malware from original files with 99% accuracy.

In this paper, there are two sections. In the first section, a correlation of malware detection methods utilizing different algorithms in machine learning such as Gradient boosting(GB), Random forest(RF), Linear Regression(LR), GNB, Adaboost, and Decision Tree (DT) for detecting malware. The PE file data is used as feature extraction. The portable executable file format consists of an optional header, major and minor link version, size of initialized and uninitialized, and its characteristics accompanied by the information. The experimental results have shown that the Random Forest algorithm provides better results to identify malware and achieved 99.43% accuracy in detection. In the following section, the classifier has been executed for the classification of benign and malware executables. The outcomes observed that .exe is a malware or benign app.

3. EXPERIMENTAL METHOD:

The proposed detection approach demonstrated with design in Figure 1. Static analysis is used for analyzing the malware sample. Static analysis tends to feature extraction. In this experiment, the header files explained with detailed values[16]. To increase the predictive accuracy and to control over-fitting, an extra-tree classifier used, it outfits the number of extra-trees on various sub-samples. It helps in finding the analysis of malware and benign applications. A list of features displayed which is identified by the extra tree classifier. After testing, the built model has shown the results with accuracy for all the algorithms like Gradient boosting, Decisiontree, Linear regression, Adaboost, random forest and GNB. The last process is the testing of malware or benign application.

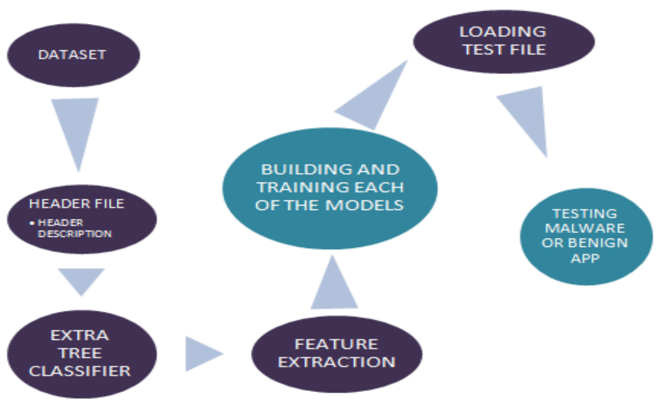

Figure 1 Detection model

3.1 DATA DESCRIPTION:

The Dataset contains the collection of malware and benign samples. The dataset is collected on both the websites and self-windows and program files. The dataset contains nearly1,30,000 data samples. The dataset contains nearly 41,000 data records are benign samples which are taken from windows, program files, encryption files, appstore execution. The records for malware samples are collected from virus share[17], virus total[18] and it contains nearly 90,000 records. The data set contains header information in figure2 and description as shown in figure 3. The binary method provides the count of malicious and legitimate files. In the results, it shows that 96724 files as malicious and 41323 files as legitimate.

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedData	...	ResourcesNb	ResourcesMeanEntropy	ResourcesMinEntropy	ResourcesMaxEntropy	ResourcesMaxSize	LoadConfigurationSize	VersionInformationSize	legitimate
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984	115712	0	...	4	3.262823	2.568844	3.537939	18032	0	16	1
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	19968	0	...	2	4.250461	3.420744	5.080177	1156	72	18	1
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120	621568	0	...	11	4.426324	2.846449	5.271813	270376	72	18	1
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728	369152	0	...	10	4.364291	2.669314	6.40072	4264	72	18	1

Figure 2: Header Information

	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedData
count	138047.000000	138047.000000	138047.000000	138047.000000	138047.000000	1.380470e+05	1.380470e+05	1.380470e+05
mean	4259.069274	225.845632	4444.145994	8.619774	3.819286	2.425956e+05	4.504867e+05	1.009525e+05
std	10880.347245	5.121399	8186.782524	4.088757	11.862675	5.754485e+06	2.101599e+07	1.635288e+07
min	332.000000	224.000000	2.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00
25%	332.000000	224.000000	258.000000	8.000000	0.000000	3.020800e+04	2.457600e+04	0.000000e+00
50%	332.000000	224.000000	258.000000	9.000000	0.000000	1.136640e+05	2.631680e+05	0.000000e+00
75%	332.000000	224.000000	8226.000000	10.000000	0.000000	1.203200e+05	3.850240e+05	0.000000e+00
max	34404.000000	352.000000	49551.000000	255.000000	255.000000	1.818587e+09	4.294966e+09	4.294941e+09

AddressOfEntryPoint	BaseOfCode	...	ResourcesNb	ResourcesMeanEntropy	ResourcesMinEntropy	ResourcesMaxEntropy	ResourcesMeanSize
1.380470e+05	1.380470e+05	...	138047.000000	138047.000000	138047.000000	138047.000000	1.380470e+05
1.719561e+05	5.779845e+04	...	22.050700	4.000127	2.434541	5.521610	5.545093e+04
3.430553e+06	5.527658e+06	...	136.494244	1.112981	0.815577	1.597403	7.799163e+06
0.000000e+00	0.000000e+00	...	0.000000	0.000000	0.000000	0.000000	0.000000e+00
1.272100e+04	4.096000e+03	...	5.000000	3.458505	2.178748	4.828706	9.560000e+02
5.288300e+04	4.096000e+03	...	6.000000	3.729824	2.458492	5.317552	2.708154e+03
6.157800e+04	4.096000e+03	...	13.000000	4.233051	2.696833	6.502239	6.558429e+03
1.074484e+09	2.028711e+09	...	7694.000000	7.999723	7.999723	8.000000	2.415919e+09

ResourcesMinSize	ResourcesMaxSize	LoadConfigurationSize	VersionInformationSize	legitimate
1.380470e+05	1.380470e+05	1.380470e+05	138047.000000	138047.000000
1.818082e+04	2.465903e+05	4.656750e+05	12.363115	0.299340
6.502369e+06	2.124860e+07	2.608987e+07	6.798878	0.457971
0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000
4.800000e+01	2.216000e+03	0.000000e+00	13.000000	0.000000
4.800000e+01	9.640000e+03	7.200000e+01	15.000000	0.000000
1.320000e+02	2.378000e+04	7.200000e+01	16.000000	1.000000
2.415919e+09	4.294903e+09	4.294967e+09	26.000000	1.000000

Figure 3: Data Description

4. FEATURE SELECTION AND EXTRACTION:

Extraction refers to the process of converting raw data into numerical features that can be prepared while storing the information in the primary data set[19]. Feature extraction is the process of converting the huge, obscure group of data into the collection of characteristics [20]. Superior prediction depends on feature extraction and selection of the malware being analyzed [21]. The features contain Name, size of code, header information, size of optional header, major and minor link versions, size of initialized and uninitialized data, disassembled files, dlls, and so on. These features are given as inputs to the machine learning algorithm. In this study, an extra tree classifier is used to increase accuracy prediction and to restrict overfitting. It fits subsamples with the number of extra tree classifiers and uses averaging. Based on the result of feature extraction by extra tree classifiers, 14 features are identified. It helps to select the features required for the classification of benign and malware files. In figure 4, the overall information and extracting features are listed.

FEATUERS
Name|md5|Machine|SizeOfOptionalHeader|Characteristics|MajorLinkerVersion|MinorLinkerVersion|SizeOfCode|SizeOfInitializedData|SizeOfUninitializedData|AddressOfEntryPoint|BaseOfCode|BaseOfData|ImageBase|SectionAlignment|FileAlignment|MajorOperatingSystemVersion|MinorOperatingSystemVersion|MajorImageVersion|MinorImageVersion|MajorSubsystemVerSion|MinorSubsystemVersion|SizeOfImage|SizeOfHeaders|CheckSum|Subsystem|DllCharacteristics|SizeOfStackReserve|SizeOfStackCommit|SizeOfHeapReserve|SizeOfHeapCommit|LoaderFlags|NumberOfRvaAndSizes|SectionsNb|SectionsMeanEntropy|SectionsMinEntropy|SectionsMaxEntropy|SectionsMeanRawsize|SectionsMinRawsize|SectionMaxRawsize|SectionsMeanVirtualsize|SectionsMinVirtualsize|SectionMaxVirtualsize|ImportsNbDLL|ImportsNb|ImportsNbOrdinal|ExportNb|ResourcesNb|ResourcesMeanEntropy|ResourcesMinEntropy|ResourcesMaxEntropy|ResourcesMeanSize|ResourcesMinSize|ResourcesMaxSize|LoadConfigurationSize|VersionInformationSize|legitimate

Figure 4: Feature Extraction

4.1 FEATURE SELECTION:

As shown in figure 4,the information about the features are listed, extracted and identified 14 features are required by extra tree classifier. The feature are extracted from the PE information. In this study, the data for training and testing is separated with the help of cross validation and for testing 0.2 proportion is included for the testing. The 14 selected features are listed in figure 5.

Features selected
1.feature DllCharacteristics (0.141259)
2. feature Characteristics (0.136174)
3. feature Machine (0.102237)
4. feature SectionsMaxEntropy (0.093866)
5. feature MajorSubsystemVersion (0.076185)
6. feature ResourcesMinEntropy (0.054568)
7. feature ResourcesMaxEntropy (0.048843)
8. feature ImageBase (0.047034)
9. feature VersionInformationSize (0.046712)
10. feature SizeOfOptionalHeader (0.041392)
11. feature SectionsMeanEntropy (0.025279)
12. feature Subsystem (0.022657)
13. feature MajorOperatingSystemVersion (0.019587)
14. feature CheckSum (0.019544)

Figure 5 Feature Selection

5. MODEL DESCRIPTION:

In this section, The model is proposed for analyzing six different machine learning techniques such as Decision Tree, GNB, Gradient Boosting, Linear regression, Random forest and Adaboost. The selection of algorithms or classifiers is based on the size of the dataset, feature type, and also the problem solution. Once the Irrelevant features are reduced, the classifier performs the classification. To perform classification, feature selection for training and testing process on every classifier.

6. EXPERIMENTAL OUTCOMES:

In Figure 6, the proposed model were displayed, this model shown the best results with the accuracy for the detection of malware and benign files from the dataset samples. The training and testing had been done on every algorithm on the model with the X_Train and X_Test. The best accuracy model will be denoted as winner. In the winner, the best accuracy results shown.

```
model = { "DecisionTree":tree.DecisionTreeClassifier(max_depth=10),
         "RandomForest":ek.RandomForestClassifier(n_estimators=50),
         "Adaboost":ek.AdaBoostClassifier(n_estimators=50),
         "GradientBoosting":ek.GradientBoostingClassifier(n_estimators=50),
         "GNB":GaussianNB(),
         "LinearRegression":LinearRegression()
}
```

Figure 6: model using machine learning Techniques.

In figure 7, The results clearly shown that random forest got the best high accuracy 99.4%. In [15] decision tree attained 99% accuracy for the detection of malware or benign samples. When compare to that, Randomforest is more appropriate for the classification of malware or benign sample. The figure 6 is the graph for accuracy results.

```
RandomForest : 0.994386091996
GradientBoosting : 0.988373777617
GNB : 0.702897500905
DecisionTree : 0.990981528432
LinearRegression : 0.54036008649
Adaboost : 0.986381745744
```

Figure 7: Accuracy Results

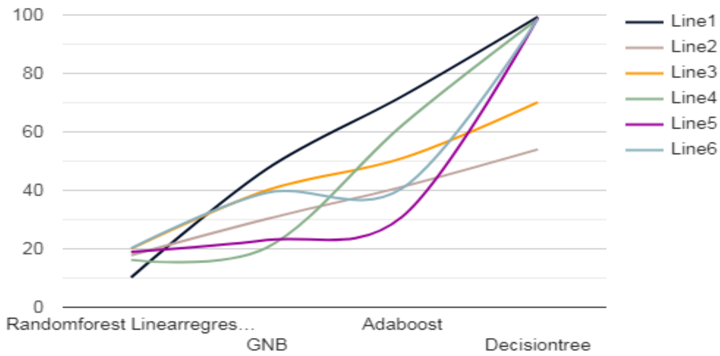

Figure 8: Graph for accuracy results

The Graph results has been shown in the figure 8 and the performance metrics are calculated based on the confusion matrix. Here the false positive and false negative rate is calculated. The False Positive Rates (FPR) provides the rate of wrongly recognized as malware. The FN (False negative) provides the rate of wrongly recognized as benign. Here we attained False positive rate as 0.099251 % and False negative rate as 0.1476%.

7. CLASSIFICATION AND TESTING:

For the classification, the classifier file is used. After the accuracy results, the classifier python file is dumped into the model. The classifier can extract the features and saved. Once the classifier and features are dumped, the next process is whether it detects the file as a benign or malware.

The main role is to distinguish the file whether it is benign or malware from the given any unseen execution file.

To test the model on an invisible file, that is necessary to remove the features of the given execution file. For construction and building the feature vector and an ML model, the python's pefile.PE library is applied to find a class for the attached file based on the previously trained model. The next step is to run after the extraction and prediction.

```
%run malware_test.py "/home/surajr/Downloads/Tweetdeck.exe"
The file Tweetdeck.exe is legitimate
To test for the malicious file, an application has been downloaded from malwr.com

%run malware_test.py "/home/surajr/Downloads/IAn12ui49643.exe"
The file IAn12ui49643.exe is malicious
```

Figure 9 Testing an unseen file (malicious or benign)

From the figure 9, it is clear that the test python file predicted the file as malware or malicious. The Tweetdeck.exe is legitimate and shows that the downloaded location of that file and found that IAN12i49643,exe is malicious.

8. CONCLUSION:

Every day malware is growing broad and more complicated. In this analysis, the main goal focus on analyzing and estimating the prediction accuracy of the classifier in ML. The latent analysis is used for the features to be extracted based on the extracted PE file and library information by analyzing six diverse classifiers on ML techniques. The training and testing of ML algorithms were proposed to classify benign and malware files. The experimental outcomes showed that the Random Forest method is better to classify the data and attained 99.4% accuracy. Based on these results, it is transparent that the PE library supports static analysis, and the selection of related features provides better accuracy for detection and perfectly describes malware. The main advantage is there is an option to test the file whether it is malware or legitimate before execution.

9. REFERENCES:

1. Imran, M., Afzal, M.T. and Qadir, M.A 2017 *A comparison of feature extraction techniques for malware analysis Turkish Journal of Electrical Engineering & Computer Sciences 25*(2) pp.1173-1183.
2. Krishna, G.B., Radha, V. and Rao, K.V 2016 *Review of Contemporary Literature on Machine Learning based Malware Analysis and Detection Strategies Global Journal of Computer Science and Technology*.
3. Mohammad DK, Mohd TS, Rafia A, Mahenoor S,& Sonalii S September (2017) *Malware detection using Machine Learning Algorithms" IJARCCE,* Vol. 6, Issue 9 available online: https://ijarcce.com/upload/2017/september17/IJARCCE%2035.pdf, last visit:20/10/2018.
4. Landage, J. and Wankhade, M.P. 2013 *Malware and malware detection techniques: A survey. International Journal of Engineering Research and Technology (IJERT), 2*(12), pp.2278-0181.
5. S.Najari & I.Lotfi 2014 *Malware detection using data mining techniques International Journal of Intelligent Information Systems* Volume 3 Issue 6-1 December Pages: 33-37.
6. Rao, V. and Hande, K. 2017 *A comparative study of static, dynamic and hybrid analysis techniques for android malware detection Int. J. Eng. Dev. Res (IJEDR) 5* pp.1433-1436.
7. Bratko, A., Cormack, G.V., Filipič, B., Lynam, T.R. and Zupan, B. 2006 *Spam filtering using statistical data compression models. Journal of machine learning research 7*(Dec) pp.2673-2698.
8. Tian, R., Islam, R., Batten, L. and Versteeg, S. 2010 *Differentiating malware from cleanware using behavioural analysis* In *2010 5th international conference on malicious and unwanted software* (pp. 23-30). Ieee.
9. Salehi, Z., Ghiasi, M. and Sami, A. *2012 A miner for malware detection based on API function calls and their arguments*. In *The 16th CSI international symposium on artificial intelligence and signal processing (AISP 2012)* (pp. 563-568). IEEE.
10. Chan, P.K. and Lippmann, R.P. 2006 *Machine learning for computer security Journal of Machine Learning Research 7*(Dec) pp.2669-2672.
11. Kolter, J.Z. and Maloof, M.A. 2004 August *Learning to detect malicious executables in the wild*. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 470-478).
12. Raff, E., Sylvester, J. and Nicholas, C. 2017 November *Learning the pe header, malware detection with minimal domain knowledge* In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (pp. 121-132).
13. Belaoued, M. and Mazouzi, S. 2015 May. *A real-time pe-malware detection system based on chi-square test and pe-file features* In *IFIP International Conference on Computer Science and its Applications* (pp. 416-425) Springer Cham.
14. Hassen, M., Carvalho, M.M. and Chan, P.K. 2017 November *Malware classification using static analysis based features* In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7) IEEE.

15. Selamat, N. and Ali, F 2019 Comparison of malware detection techniques using machine learning algorithm *Indones. J. Electr. Eng. Comput. Sci 16* p.435.
16. Yuxin, Ding, and Zhu Siyi 2019 *Malware detection based on deep learning algorithm Neural Computing and Applications* 31.2 (2019) 461-472.
17. https://virusshare.com/
18. https://www.virustotal.com/
19. https://www.mathworks.com/discovery/feature-extraction.html
20. Ranveer, S. and Hiray, S 2015 *Comparative analysis of feature extraction methods of malware detection International Journal of Computer Applications 120*(5).
21. Qu, Y. and Hughes, K 2013 December *Detecting metamorphic malware by using behavior-based aggregated signature* In *World Congress on Internet Security (WorldCIS-2013)* (pp. 13-18) IEEE.
22. Ranveer, S. and Hiray, S. 2015 *Comparative analysis of feature extraction methods of malware detection International Journal of Computer Applications 120*(5).
23. Yuan, M.Y. 2014 *Data mining and machine learning* WEKA application technology and practice.
24. Zhao, Z., Wang, J. and Bai, J. 2014 *Malware detection method based on the control-flow construct feature of software IET Information Security 8*(1) pp.18-24.
25. Salton, G. and Buckley, C. 1988 Term-weighting approaches in automatic text retrieval *Information processing & management 24*(5) pp.513-523.
26. Santos, I., Brezo, F., Ugarte-Pedrero, X. and Bringas, P.G. 2013 Opcode sequences as representation of executables for data-mining-based unknown malware detection *Information Sciences 231* pp.64-82.
27. Sarikaya, R., Hinton, G.E. and Deoras. A 2014 *Application of deep belief networks for natural language understanding IEEE/ACM Transactions on Audio, Speech, and Language Processing 22*(4) pp.778-784.
28. Jerlin, M.A. and Marimuthu, K 2018 *A new malware detection system using machine learning techniques for API call sequences Journal of Applied Security Research*, *13*(1) pp.45-62.
29. Qu, Y. and Hughes, K., 2013 December Detecting metamorphic malware by using behavior-based aggregated signature In *World Congress on Internet Security (WorldCIS-2013)* (pp. 13-18) IEEE.
30. Jung, J., Kim, H., Shin, D., Lee, M., Lee, H., Cho, S.J. and Suh, K 2018 September *Android malware detection based on useful API calls and machine learning* In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)* (pp. 175-178) IEEE.
31. Sami, A., Yadegari, B., Rahimi, H., Peiravian, N., Hashemi, S. and Hamze, A 2010 March Malware *detection based on mining API calls* In *Proceedings of the 2010 ACM symposium on applied computing* (pp. 1020-1025)