

API Calls Based Malware Detection using Behavior Graphs

M.Prabhavathy¹, Dr.Valliappan Raman² and G.Thilagavathi³

¹Assistant Professor, Department of AI & DS

²Professor, Department of AI & DS, Coimbatore Institute of technology, Coimbatore

³Assistant Professor, Department of IT, Coimbatore Institute of technology, Coimbatore
E mail: prabhavathy.phd@gmail.com

Abstract— Now-a-days, the demand for malware detection has been increased to protect the system from malware crash and to keep organization's data safe and secure. IoT devices are the nonstandard computing devices that connect wirelessly to a network and have the ability to transmit data. As IoT devices becomes more functional, the security became one major factor. More attacks have been targeted towards IOT devices and the security is not guaranteed. Hence several models have been developed to overcome these malware attacks. In order to get the better of, the model for detecting malware detection using behavior graphs is developed. In this approach, Behavior-Based Framework is used to extract the behavior graphs using API Calls and the output is obtained using the feature extraction and certain classifiers. The suggested model seeks to decrease the amount of characteristics and to convey the features succinctly at the highest level. Accuracy of the model is 99.1%

I. INTRODUCTION

One of the biggest security risks on the Internet right now is malware. Every day, several virus types are created automatically. In the IoT ecosystem, traditional PCs are a major source of assaults. Computers are attacked by malware, which then leverages those compromised machines to target other connected objects in an IoT ecosystem. The malware infected windows has the possibility of stealing the confidential information from the system and transform the influenced devices into a botnet and then it will begin to launch the new distributed denial of service which is s DDoS attack . Malware assaults on traditional PCs sometimes spread to other IoT devices as well. Unfortunately, there are no effective optimal ways to protect against the dangers posed by Mirai and other IoT devices. This project uses a technique to distinguish between benign and malicious files in order to stop those assaults. In this proposed system, API calls are extracted by tracking the executable files in local machine. Utilizing extracted API calls, behaviour graphs are created to effectively communicate information about malware actions. To identify malware in executable files on a wide scale, the cloud platform has a behavior-based deep learning framework (BDLF). The use of machine learning and deep learning techniques is increasing tremendously in the society. It plays a vital role in analysing the large amount of data and gives us a better result. In this project, random forest algorithm is used to get better results. This system works as a real time model in order to give results instantly.

II. LITERATURE SURVEY

Deep Learning for Zero-day Flash Malware Detection was proposed by Wookhyun Jung, Sangwon Kim, Sangyong

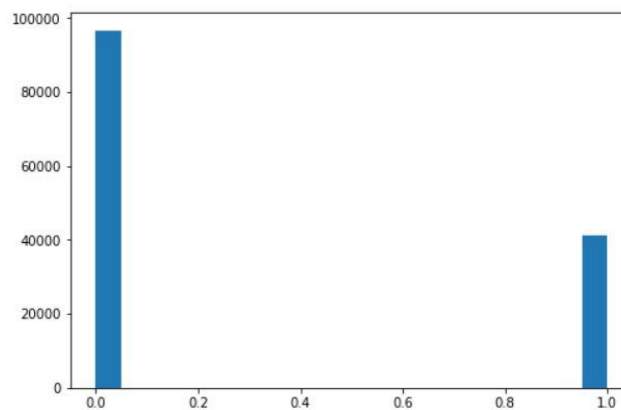
Choi's on 2016. In that paper he mainly aims at detecting zero-day Flash malwares through all of the features SWFs. This detection system can keep track of zero-day Flash malwares. It can learn new features of zero-day Flash malwares. It dynamically go through the features of zero day flash malwares and detects the new variants. That has limitations that it detects only zero day flash malwares and not any other malwares.

A research to understand malware behaviour using malware analysis was proposed in 2019 by Om Prakash Samantray, Satya Narayan Tripathy, and Susanta Kumar Das. This study focuses on the behaviour, traits, and attributes of malware as determined by various analytical tools, with the goal of determining whether to include them into behavioral-based malware signatures. They made an attempt to understand the malware behavior using a few openly available tools for malware analysis. Proposed has optimal results compared with existing systems. More dataset is needed to accommodate enough samples which is limitation for their paper.

Fei Xiao, Zhaowen Lin, Yi Sun, Yan Ma's proposed a Malware Detection Based on Deep Learning of Behavior Graphs on 2019. In this research, a method is suggested for preventing local computers from attacking IoT devices. For the purpose of identifying malware in an IoT context, an unique behavior-based deep learning framework (BDLF) is implemented within the cloud platform. The outcomes of the experiment show that the proposed BDLF can derive from behaviour graphs the semantics of higher-level harmful activities. Binary vectors are used in this paper for accurate prediction. Limitations: Incorporating more data to have large dataset will provide more accuracy and efficiency.

III. PROPOSED WORK:

For this project malware and benign dataset is collected from Kaggle where they are providing plenty of datasets for deep learning based work and several other real time project works. This dataset consists of 13807 executable files in which it contains 41323 benign files and 96724 malware files.



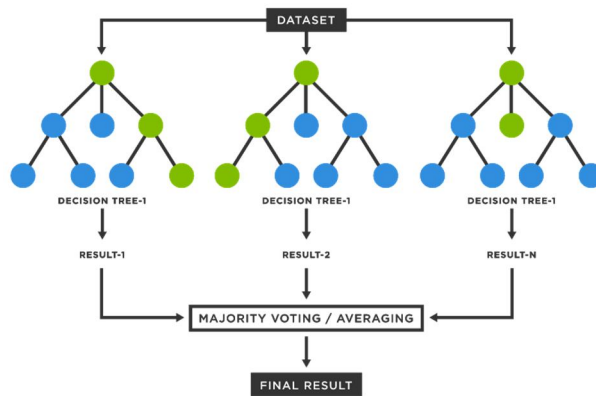
To obtain the accuracy rate several algorithms like random forest, neural network and gradient boosting algorithms were used. the dataset in this project undergoes training and testing using those algorithms with the help of deep learning.

IV. RANDOM FOREST CLASSIFIER ALGORITHM:

Random forest is a machine learning algorithm which is used for classification and regression problems. It builds decision trees on different random samples and takes their majority vote for classification and average for regression.

For this project random forest classifier is used for classifying the models into train and test the dataset. Rfe = recursive feature elimination and rfc = random forest classifier used in this project for further processing.

A random forest in term is a meta estimator which enables for fitting a number of decision tree classifiers and is on various sub-samples of the dataset and it uses averaging which is from the result set to improve the predictive accuracy and control over-fitting. The sub-sample size is also controlled within the maximum samples (max_sample) parameter if bootstrap=True which is by default otherwise the whole dataset is used to build each separate tree. Confusion matrix is used in algorithm is to evaluate the accuracy of classification.



V. IMPLEMENTATION:

Initially dataset is collected for classification. Then API calls are to be extracted from the dataset. In this project .exe file is given to API call extractions and features were extracted for sample exe files. Based on the features approximately 1 lakh dataset i.e., exe files are taken for this project. The dataset are stored in csv files. They are read and kept in dataframe. The important features from total features were taken and they are stored in another csv file. Based on api calls behavior graphs are constructed. For constructing behavior graphs igraph python package is installed.

VI. EXTRACTING API CALLS

pefile is imported and used for extracting API calls. Pe stands for portable executable , it works with exe files and features were extracted from exe files and based on those features malware files are collected.

```

import pefile
pe = pefile.PE("/content/drive/MyDrive/malware/eclipse.exe")
pe.print_info()

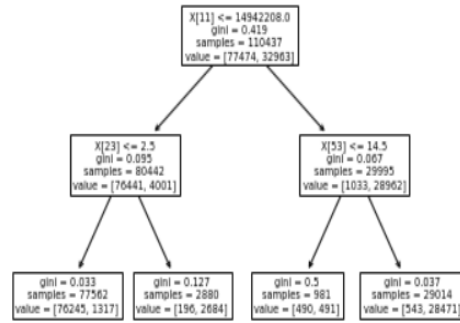
[IMAGE_OPTIONAL_HEADER64]
0x98 0x0 Magic: 0x20B
0x9A 0x2 MajorLinkerVersion: 0x2
0x9B 0x3 MinorLinkerVersion: 0x18
0x9C 0x4 SizeOfCode: 0x4AA00
0xA0 0x8 SizeOfInitializedData: 0x4AA00
0xA4 0xC SizeOfUninitializedData: 0xA00
0xA8 0x10 AddressOfEntryPoint: 0x14B0
0xAC 0x14 BaseOfCode: 0x1000
0xB0 0x18 ImageBase: 0x400000
0xB8 0x20 SectionAlignment: 0x1000
0xBC 0x24 FileAlignment: 0x200
0xC0 0x28 MajorOperatingSystemVersion: 0x4
0xC2 0x2A MinorOperatingSystemVersion: 0x0
0xC4 0x2C MajorImageVersion: 0x0
0xC6 0x2E MinorImageVersion: 0x0
0xC8 0x30 MajorSubsystemVersion: 0x5
0xCA 0x32 MinorSubsystemVersion: 0x2
0xCC 0x34 Reserved1: 0x0
0xD0 0x38 SizeOfImage: 0x5C000
0xD4 0x3C SizeOfHeaders: 0x400
0xD8 0x40 CheckSum: 0x70D3763
0xDC 0x44 Subsystem: 0x2
0xDE 0x46 DllCharacteristics: 0x0
0xE0 0x48 SizeOfStackReserve: 0x200000
0xE8 0x50 SizeOfStackCommit: 0x10000
0xF0 0x58 SizeOfHeapReserve: 0x100000
0xF8 0x60 SizeOfHeapCommit: 0x10000
0x100 0x68 LoaderFlags: 0x0

```

In the above figure the features extracted from exe files is printed. The weight values which are calculated from features values were added into matrix.

malDta=pd.DataFrame(weight_value,columns=['ResourcesMaxEntropy','SectionsMaxEntropy','DllCharacteristic s','SectionsMinEntropy','ResourcesMeanEntropy','CheckSum','SectionsMeanEntropy','ResourcesMinEntropy','M ajorSubsystemVersion']) the main features were taken as columns and weights are filled in that column values. Now binary vectors are available in matrix form and important feature 'legitimate' is used for detecting benign and malware from exe files.

These are given into the model called random classifier algorithm to find the accuracy.



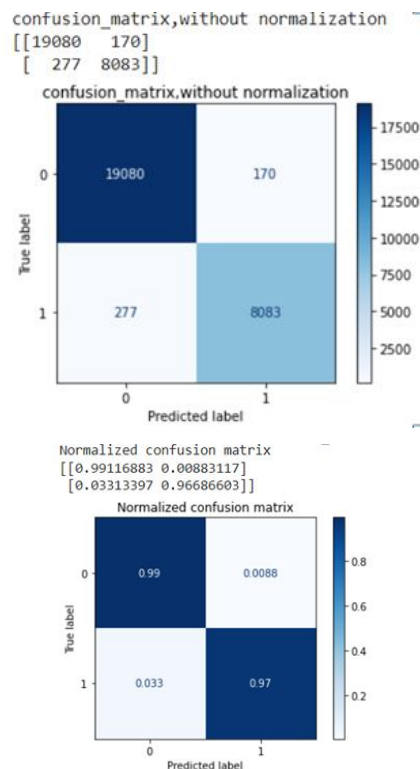
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(malData, y, test_size = 0.2, random_state = 42)
```

According to random forest, many decision trees were used. Here one of the decision tree is shown which is obtained from our dataset.

The dataset is splitted for training and test datasets. Then the features are given into ExtraTreesClassifier() to get the important features. The below figure shows the feature importances.

VI. RESULT

F1 score of a test and prediction is 0.97 in this project. Then confusion matrix is mostly looks like a table which describes the performance of the classifier on test data where truth values are known.



After finding the accuracy the confusion matrix is used to represent the performance in diagrammatic explanation.

VII. CONCLUSION

In this research paper, unknown malwares are detected in large samples using features extraction. Dynamic analysis advantages are well defined and their benefits to identify the benign files and malware were described in this paper. The random forest classifier model used in this paper gives the accuracy of 97 percent in detecting malwares. API calls are extracted from files which are in exe format and based on the API calls collected feature selection takes place the important features were extracted from total of 54 features from exe files and they are used to create behavior graphs based on the binary vector matrix with the help of adjacency matrix and igrph is used.

REFERENCES

- [1] Prabhavathy, M. & Sankareswaran, Uma Maheswari. (2020). Remote Network Injection Attack Using X-Cross API Calls. 10.1007/978-3-030-32150-5_142.
- [2] M.Prabhavathy, S. Uma Maheswari, R. Saveeth, S. Saranya Rubini: Permission and API Calls Based Hybrid Machine Learning Approach for Detecting Malicious Software in Android System. J. Multiple Valued Log. Soft Comput. 37(5-6): 553-571 (2021)
- [3] Prabhavathy, M., and S. Uma Maheswari. "Pigeonhole Malicious Attack On Big Data Using Automated Distributed Stochastic Neighbor Embedding (T-Sne)."
- [4] M. Prabhavathy and S. UmaMaheswari, "Prevention of runtime malware injection attack in cloud using unsupervised learning," Intelligent Automation & Soft Computing, vol. 32, no.1, pp. 101–114, 2022.
- [5] Fei Xiao, Zhaowen Lin, Yi Sun, Yan Ma's ". Malware Detection Based on Deep Learning of Behavior Graphs" on 2019.
- [6] Om Prakash Samantray, Satya Narayan Tripathy, Susanta Kumar Das"A study to Understand Malware Behavior through Malware Analysis" on 2019.
- [7] J. Su, V.D. Vasconcellos, S. Prasad, S. Daniele, Y. Feng, K. Sakurai's "Lightweight classification of IoT malware based on image recognition" on 2018.
- [8] Wookhyun Jung, Sangwon Kim, Sangyong Choi's "Deep Learning for Zero-day Flash Malware Detection" On 2016.
- [9] M. Fan, J. Liu, X. Luo et al., "Android malware familial classification and representative sample selection via frequent subgraph analysis," IEEE Transactions on Information Forensics and Security, vol. 13, no. 8, pp. 1890–1905, 2018.
- [10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems, pp. 91–99, 2015.
- [11] L. Vareka and P. Mautner, "Stacked autoencoders for the P300 component detection," Frontiers in Neuroscience, vol. 11, p. 302, 2017.
- [12] T. Lee, B. Choi, Y. Shin, and J. Kwak, "Automatic malware mutant detection and group classification based on the n-gram and clustering coefficient," The Journal of Supercomputing, pp. 1–15, 2015.
- [13] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in Proceedings of the Australasian Joint Conference on Artificial Intelligence, Lecture Notes in Comput. Sci., pp. 137–149, Springer, Cham, Switzerland, 2016.
- [14] Prabhavathy, M., Uma Maheswari, S., Saveeth, R., Saranya Rubini, S., Surendiran, B. (2022). A Novel Approach for Detecting Online Malware Detection LSTMRNN and GRU Based Recurrent Neural Network in Cloud Environment. In: Rathore, V.S., Sharma, S.C., Tavares, J.M.R., Moreira, C., Surendiran, B. (eds) Rising Threats in Expert Applications and Solutions. Lecture Notes in Networks and Systems, vol 434. Springer, Singapore. https://doi.org/10.1007/978-981-19-1122-4_1
- [15] Prabhavathy, M., and K Sivasankari "Federated Query Processing Service in Service Oriented Business Intelligence", International Conference on Engineering Technology and Science, Pg 266-1272.