International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 12, Issue 3 [March. 2023] PP: 282-285

Malware Detection System Using Machine Learning

[1]N. Nagalakshmi, *Assistant professor, Department of Information Technology, Anurag University*
[2]R. Sai Priya, [3]S. Suraj, [4]P. Meghana
UG Scholar, Department of Information Technology, Anurag University, Hyderabad
[1]nagalakshmiit@cvsr.ac.in, [2]raminisaipriya@gmail.com, [3]surajvictor7277@gmail.com,
[4]meghanareddypaga@gmail.com

ABSTRACT
In order to effectively differentiate between safe files and malicious files while minimising the number of false positives, we suggest an ideal solution in which one can use various machine learning algorithms. In this project, we examined the PE Headers of both legitimate and malicious samples (files) and presented the concepts underlying our solution by first performing exploratory data analysis (EDA) on the dataset, and then applying various types of algorithms and creating a malware classifier to find the best solution for the issue and attempt to maximise accuracy. The purpose of this model is to determine whether the provided samples or the files are valid after testing effectively on the tested dataset of legitimate and clean files.

Date of Submission: 14-03-2023 Date of acceptance: 29-03-2023

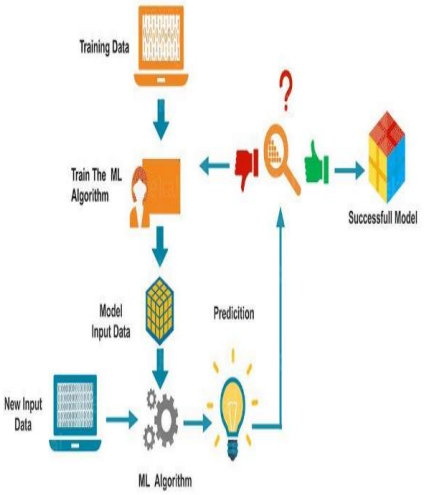
I. INTRODUCTION

System to detect malware and protect against file harm is present. (like: viruses, worms, Trojan horses, spyware, and other forms of vindictive code). Technologies for identifying malicious files and stopping them from being opened are widely available for servers, gateways, user workstations, and mobile devices. Some of these solutions also offer the capability to centrally monitor malware detection software that has been installed on a number of different computers or structures. Malware is one of the most genuine security concerns and spreads on its own through client carelessness or weakness. depending on the replication of Windows API calls in order to distinguish zero-day malware with great accuracy. Zero-day malware is a product security flaw that the manufacturer is aware of but which has not yet been addressed. Cybercriminals may take advantage of it. The Windows API is a middle-tier set of application programming interfaces (APIs) available in Microsoft Windows-based systems. In essence, the Windows API is a partner with all Windows programmes. A Windows programme can utilise a variety of restrictions and data formats to request that Windows do an action, such as opening a report or displaying a message. Essentially, invoking different API limitations is what every Windows programme does. Despite this, given the high receptivity of attacking devices in today's cyberspace, there is a decline in the dominance nearby that is necessary for malware development. High accessibility of unfavourable confirmation techniques, as well as the ability to look for malware on the black market, could end up being an aggressor for anyone, excluding those at the limit. According to recent assessments, an increasing percentage of assaults are being committed by happy youths or using machinery. Malware protection of PC systems is thus one of the most important network security duties for both individual customers and associations, since even a single attack has the potential to result in compromised data and significant setbacks. Massive challenges and everyday assaults drive the need for thorough and helpful acknowledgment methods. As a result, AI-based strategies are applicable. This study explores the main concerns and challenges of AI-based malware detection while also looking for the most effective element depiction and grouping techniques.

II. LITERATURE SURVEY

The world is vulnerable to cyberattacks because there are so many computers, smartphones, and other devices that can connect to the Internet. In response to the growth in malware activity, a multiplicity of malware detection techniques have emerged. Researchers employ a range of big data tools and machine learning strategies while attempting to find malicious code. Although they take a long time to process, traditional machine learning-based malware detection techniques may successfully spot recently discovered malware. Given the popularity of contemporary machine learning techniques like deep learning, feature engineering may become obsolete. We looked at a range of malware detection and classification strategies in this study. To check samples for malicious intent, researchers have developed methods that use machine learning and deep learning. Armaan (2021) provided examples of various models and evaluated their accuracy. No digital platform application can function properly without data. Since there are many cyber hazards, it is crucial to implement security measures to protect data.

Machine learning is a cutting-edge method that leads the way for accurate prediction, even though feature selection is challenging when creating any kind of model. The method requires a workaround that can be flexible enough to accommodate unusual data. We must evaluate malware and develop fresh guidelines and patterns in the form of new malware types, in order to successfully manage and stop upcoming attacks. Malware analysis tools are sometimes used by IT security experts to discover patterns. The cybersecurity industry benefits greatly from analyses of malware samples to identify the severity of their malignancy. By monitoring security alerts, these solutions assist guard against malware attacks. If malware is harmful, we must get rid of it right away to stop it from spreading the infection. A growing number of businesses are turning to malware analysis to assist mitigate the effects of the threats posed by malware and the more sophisticated ways in which it can be used to attack.



A practical machine learning classification technique for malware detection was suggested by Chowdhury (2018). We investigated whether improving some factors might improve the classification accuracy of malware. Our strategy integrated N-gram and API call capabilities. Experimental testing validated the effectiveness and reliability of our suggested method. In order to improve detection accuracy while lowering erroneous positives, future work will concentrate on merging a sizable number of features.

III. SYSTEM ARCHITECTURE

In the below Figure , the architecture of the malware detection system is depicted. The input and dataset collection are dealt with using the suggested system.

- Processing before.
- Using 3 Algorithms To Build The Model.
- Evaluation, testing, and prediction.

IV. METHODOLOGY

Collection of Input and Datasets:

- The input file is essentially an.exe file.
- We collected both malicious and helpful executables from various sources.
- We have gathered the safe data folders from the Google collaboration.

Jupyter Notebooks can be accessed through remote servers using Google Colab, a free cloud tool.

- Colab is a free Jupyter notebook environment that is completely cloud-based.
- The model was created in Google Colab, an online platform. Malicious and benign applications are separated out in the dataset.
- Ample sample training datasets from real antivirus software should be obtained.
- After mounting Google Drive in the notebook, data required for a project can be uploaded to Google Drive and viewed within the code.
- Removing noisy data from malware is the following procedure.

Preprocessing:

- Data cleaning is the first stage in the preprocessing process, during which duplicates, incorrectly formatted data, and corrupted data are corrected or eliminated. The data may contain numerous irrelevant and lacking pieces. Data cleansing is carried out to manage this portion. It entails managing noisy data, lacking data, etc.
- Data integration, the next stage, combines data from various sources into a single view.

• Data reduction is the third stage, during which the data are encoded, scaled, and, if necessary, sorted. Due to the fact that data mining is a technique used to handle a large quantity of data. Analysis became more difficult in these situations when dealing with a large volume of data.
• We employ the data compression method to do away with this. It seeks to lower data storage and analysis costs while improving storage efficiency.
• Data transformation, the last stage, involves transforming the data into the necessary format. This procedure is used to change the data into appropriate formats for the mining process.
Three Algorithms Are Used To Build The Model:
• Random Forest - Using trees with random features but fixed structures is a good option. Forests are collections of trees, and classifiers named Randomforests are included.
• Gradient boosting - The purpose of "gradient boosting" is to utilise a weak
Adjust the learning method in a number of ways to strengthen the learner's/hypothesis's foundation.
• Decision Tree - This is a tree structure resembling a flowchart in which each leaf node represents the result, the branch represents a decision rule, and the internal nodes represent features.

V. EXPERIMENTAL RESULTS & DISCUSSION

Apply Decision Tree Classifier

```
In [22]: from sklearn.tree import DecisionTreeClassifier
         dtc = DecisionTreeClassifier()

In [23]: dtc.fit(X_train,Y_train)
Out[23]: DecisionTreeClassifier()

In [24]: (dtc.score(X_test, Y_test))*100
Out[24]: 99.04382470119522
```

Apply Random Forest Classifier

```
In [26]: from sklearn.ensemble import RandomForestClassifier
         rcf = RandomForestClassifier(n_estimators=50,random_state=7)
```

fit the model in X_train and Y_train

```
In [27]: rcf.fit(X_train,Y_train)
Out[27]: RandomForestClassifier(n_estimators=50, random_state=7)
```

Predict the Accuracy and Score

```
In [28]: (rcf.score(X_test, Y_test))*100
Out[28]: 99.34806229626946
```

Apply Gradient Boosting Classifier

```
]: from sklearn.ensemble import GradientBoostingClassifier
   gbc = GradientBoostingClassifier(n_estimators=50)
```

fit the model in X_train and Y_train

```
]: gbc.fit(X_train,Y_train)
]: GradientBoostingClassifier(n_estimators=50)
```

Predict the Accuracy and Score

```
]: gbc.predict(X_test)
]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
]: result_1 = (gbc.score(X_test, Y_test))*100
   result_1
]: 98.62610165399009
```

VI.CONCLUSION AND FUTURE WOR

In this project we have built a system for classifying a question into High Quality or Low-Quality Question. we give input test data to generated system it gives desired output. Built a dozen different models on the dataset. Found out Classifier to be the best model. After Applying these Algorithms,
we have concluded that
• Decision Tree Model Accuracy- 98.39%

• Random Forest Classifier – 99.35%
• Gradient Boosting Classifier – 98.64%

Machine learning models have a very great capability to surpass the human potential if the data provided is sufficient. The system has provided the 99.35% accuracy in prediction of detecting the legitimate file and the malware file . We can predict the amount of time the question will take to get answered.

REFERENCES

[1]. "Malware classification using deep learning methods," in Proc. ACMSE 2018 Conf., vol. 2018-January, no. April 2018.

[2]. A. P. Namanya, A. Cullen, I. U. Awan, and J. P. Disso, "The World of Malware: An Overview," 2018 IEEE 6th International Conference on Future Internet Things Cloud, FiCloud 2018, no. September, pp. 420–427.

[3]. The rise of machine learning for malware detection and classification: Research advances, trends, and challenges. J. Netw. Comput. Appl., vol. 153, no. July 2019, p. 102526, 2020.
[Online].You can access it at: https://doi.org/10.1016/j.jnca.2019.10 2526.

[4]. M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust International Journal of Computer Applications (0975 - 8887) Volume 183 - No. 32, October 2021" 17 Knowledge-based malware detection with Deep Learning," IEEE Access, issue. 7, 2019, pp. 46 717–46 738.

[5]. Applied Sciences, vol. 8, no. 6, p. 981, June 2018. V. Menger, F. Scheepers, and M. Spruit, "Comparing deep learning and traditional machine learning approaches for predicting inpatient violence incidents from clinical text." [Online]. You can access it at https://doi.org/10.3390/app8060981.

[6]. Finding efficient classifier for malicious URL identification," in Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences - ICMSS 2018. C. Liu, L. Wang, B. Lang, and Y. Zhou. 2018; ACM Press. [Online].You can access it at: https://doi.org/10.1145/3180374.3181 352

[7]. Z. Cui, F. Xue, X. Cai, Y. Cao, G. G. Wang, and J. Chen, "Detection of Malicious Code Variants Based on Deep Learning," IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3187-3196, 2018.

[8]. Int. J. Eng. Res. Africa, vol. 24, no. February 2017, pp. 124–136, "A review of deep machine learning," by B. B. Benuwa, Y. Zhan, B. Ghansah, D. K. Wornyo, and F. B. Kataka.

[9]. Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-Aware neural language models," in Proceedings of the 30th AAAI Conference on Artificial Intelligence, 2016, pp. 2741–2749.

[10]. "Early-stage malware prediction using recurrent neural networks," Comput. Secur., vol. 77, no. December 2017, pp. 578–594, 2018. [Online]. The following link is available: 10.1016/j.cose.2018.05.010.