INTENSIVE MALWARE DETECTION APPROACH BASED ON DATA MINING

Israa Ezzat Salem[1*], Karim Hashim Al-Saedi[2]
Computer Science Department, College of Science, Mustansiriyah University, Baghdad, Iraq[1]
farg.israa@uomustansiriyah.edu.iq, sraa.ezzat@baghdadcollege.edu.iq

Received : 31 July 2023, Revised: 01 November 2023, Accepted : 04 November 2023
**Corresponding Author*

ABSTRACT
Malicious software, sometimes known as malware, is software designed to harm a computer, network, or any of the connected resources. Without the user's knowledge, malware can spread throughout their computer system. Malware is typically disseminated via online connections and mobile devices. While malware has always been a problem in the digital age, its effects have gotten increasingly serious. Traditional malware detection methods seek to locate specific malware samples and families to recognize harmful codes and can be located using traditional signature- and rule-based detection methods. The research focuses on developing malware detectors using data mining techniques. The proposed method outlined below sets itself apart by emphasizing the processing of malware behaviors significantly dependent on aspects. Finding more dependable intelligent detecting techniques is a crucial component of this paper. In order to identify the cluster of the most essential malware features and use decision tree classifiers for malware detection, the study, a common methodology for creating malware detectors based on data mining, is implemented and investigated. Our approach can identify the most significant features of malware that can significantly determine and detect a malware code.
Keywords: *Malware Detection, Decision Tree, Machine Learning, Identify Malware Attack.*

1. Introduction

The most popular malware detection program clearly demonstrates how to spot malicious code. Early, the virus had basic objectives, making it easier to find. This kind of malware might be called conventional (basic) malware. Conversely, next-generation malware can now run in multiple shapes, is more harmful than classic malware, and is more challenging to detect (Alagrash *et al.*, 2019).

Malware of this kind may easily bypass firewalls, antivirus software, and other security tools running in the system. Traditional malware frequently has just one process and does not use advanced techniques for disguising itself. On the other hand, modern malware uses many active or dormant processes at once and various obfuscation techniques to hide itself and persist in the system (Glanz et al., 2017; Abdulhameed *et al.*, 2020).

This paper investigated a research problem that relies on new-generation malware to launch more destructive attacks, including never-before-seen targeted and persistent ones, and different viruses are used in the attacks (Hataba *et al.*, 2022), so the traditional solution is not more affected to detect malware, the new and smart solution is true.

Over the past ten years, data mining techniques have mainly been used to find malware. Because malware is complicated and evolving swiftly, identifying it is becoming increasingly difficult. Hence, practical approaches must be improved. Machine learning is used more frequently to identify harmful files when applying malware detection mechanisms employing data mining techniques (Aslan & Samet, 2020). A particular preparation set may contain malicious and benign examples that machine learning techniques can use. Simple illustrations can help distinguish between malicious and helpful code (Alagrash *et al.*, 2020). For distributed systems and the Internet, malware is the most considered intimidation (Pan *et al.*, 2020; Souri & Hosseini, 2018).

The most important qualities that can be used to create malware are identified and recognized using a data mining technique developed in this study. Our method can distinguish between generic activities and malware operations, emphasizing to the dynamic data mining patterns. Organization:

The rest of this essay is structured as follows. The related efforts in detecting the loss of computer accounts due to malware and insider threats are discussed in Section II. Identify the suggested model in Section III. Explain the results and discussion in Section IV. Section V finishes the study by summarizing our results.

2. Literature Review

Malicious code detection uses a variety of techniques. The majority of these migrate to the following areas:

Static detection approaches have often been the first to appear. This was made clear by the extremely low number of malware variants and the lack of thoughtfully developed defense mechanisms. The principal static detection method that quickly and accurately identifies threats and displays the apparent steps that must be taken to remove them and their effects is signature scanning. The methods for obfuscating (packing, polymorphic altering) the key program fragments are employed in static signature-based detection. This challenge prompted the creation of more complex technological judgments based on the partial or complete execution of the researched software in a separate setting (virtual machines, sandboxes, etc.) (Pan *et al.,* 2020).

Souri & Hosseini (2018) have proposed an approach to recognize viruses by data mining based on the static analysis of basic executables and have the same traditional research strategy to the point where the intense effort stays the same. However, the solution cannot detect novel malware.

Several researchers used a dynamic analysis approach to collect various types of data in order to distinguish between malicious and benign files by running the executable files in isolated environments, virtual machines (VM), or emulators and watching the behavior of the executable files during run-time which is proposed by (Choudhary & Vidyarthi, 2015).

Galal *et al.* (2016) proposed a malware detection system by using a dynamic analytic technique, many sorts of data have been obtained. It is possible to dynamically depict harmful activities by observing executable file behavior and the retention of memory images during run-time. It is possible to identify the behaviors of executable files by gathering data on the performed API calls. In (Hwang *et al.,* 2020) the author builds an approach based on machine activity to detect malware dynamically. For example, a proposed solution by Jerlin & Marimuthu (2018) relies on user activity and behaviour to detect malware in run-time terms. Many studies on machine activities in (Kim *et al.,* 2019; Fasano *et al.,* 2019; Ahmed *et al.,* 2020) an idea behind this based on file-related data such as in (Singh & Singh, 2020; Belaoued *et al.,* 2019; Norouzi *et al.,* 2016) and registry and network data (Arabo et al., 2020). It is possible to depict dangerous behavior using opcode-based memory images dynamically. Obfuscated malware can't conceal how it works when examined, yet dynamic analysis can't satisfy all dangerous conditions to probe every execution route as indicated in (Aslan & Samet, 2020).

Although dynamic detection methods are superior to static detection methods at detecting modern malware, they do have some inherent drawbacks, including (1) slow decision-making speed, (2) high resource usage, and (3) the existence of some specific tricks that can mislead a detection unit about the true functionality of the analyzed object (e.g. anti-debugging, executing malicious functionality when some specific conditions are met, including update events, e. The existing work indicates that efforts to enhance dynamic detection systems will soon encounter technological hurdles. As previously stated, it may be explained by the availability of efficient anti-debugging (anti-emulation) technologies and recent developments in applying the primary stages of the Malwa life cycle (Komashinskiy & Kotenko, 2010)

Combining data from static and dynamic analysis was developed in (Huang *et al.,* 2021), which lessens the drawbacks of each analytical technique and has increased detection rates in some earlier studies. IDA Pro Disassembler, Cuckoo Sandbox, OlleyDbg, and other tools collect dynamic and static data. Then, hybrid feature sets are created based on various data types, including text, opcode, API calls, and others. Despite combining static and dynamic analysis advantages, the hybrid analysis technique has flaws (Qin *et al.,* 2017).

In addition, (ML) machine learning is frequently used to support the required business objectives successfully and has achieved this in part because it is capable of handling enormous amounts of data, including Application Programming Interface (API) calls, Assembly code

(Opcode), and Byte code, which is unacceptable for humans to handle (Ranveer & Hiray, 2015). Because of their great generality, machine learning (ML) techniques are extensively used in malware detection.

Based on various types of data that affect the overall detection and classification performance, ML approaches such as Support Vector Machine (SVM), Naive Bayes (NB), Decision Trees (DT), etc., were utilized to locate risky applications for modelling purposes (Gardiner & Nagaraja, 2016; Muhamed, 2022; Hassan *et al.*, 2023).

Furthermore, the importance of static, dynamic, and hybrid feature extraction methods was highlighted, each connected to the most common data types as introduce in (Abusitta *et al.*, 2021).

3. Research Methods

The key goal of this article is to detect malware based on a data mining approach. This solution exploits the data mining-based malware detection method through known malware and identifies most malware members. The classifier model was trained for detection using the original dataset's inputs and the rebuilt dataset's matrices. Contrasting the detecting effects of various data processing techniques and decision tree classifiers.

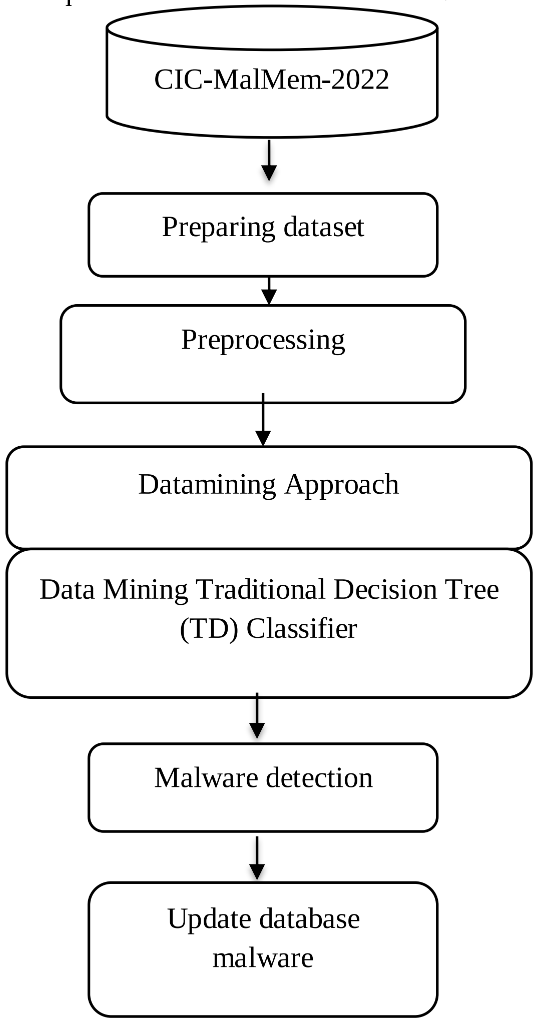


Fig. 1. Malware Detection Based on Data Mining Approach

4. Results and Discussions

In this article, we used a research approach dependent on the steps in the proposed architecture shown in Figure 1. The malware dataset to be processed is first spotted and visualized in the suggested system.CIC-MalMem-2022 is a standard dataset and an academic dataset published by the Canadian Institute for Cybersecurity aimed at malware classification research, particularly the detection of malware. 29,298 entries are benign and 29,298 records are harmful in the 58,596 record collection. To carry out training and testing for the proposed known/unknown malware, CIC-MalMem-2022 is employed. For the malware types Trojan Horse (c), Ransomware (b), and Spyware (a), Figure (2) breaks out which malware families are employed in each category.

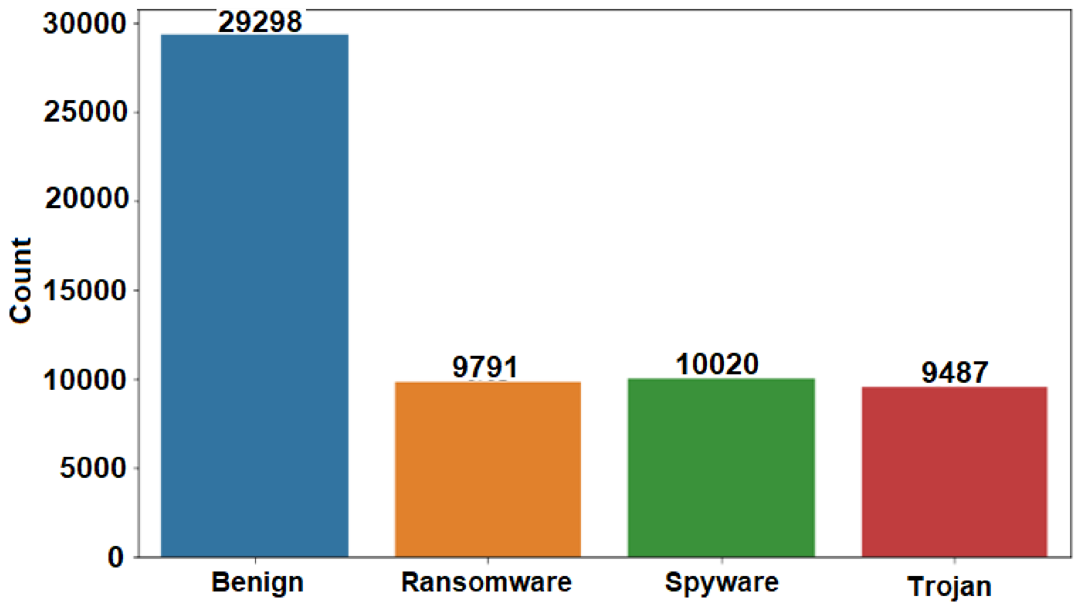


Fig. 2. CIC-MalMem-2022 Dataset Malware Families

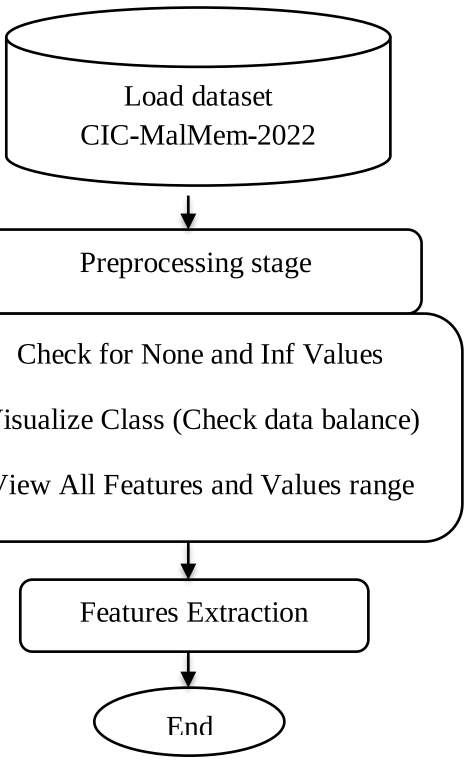
4.1 Preprocessing dataset

An important phase in the modeling process is the preparation of the input data. It involves evaluating the quality of the data intake and, eventually, optimizing the input types, selected processes, and time constraints. It directly affects the accuracy, reliability, and outcomes of categorization. The flowchart of the preprocessing processes is shown in Figure (3).

- The first step is to check for missing data; because there isn't any, our data collection is full, meaning there aren't any redundant values or null displaying label types.
- Second step of preprocessing: This stage is frequently taken before the classifier's construction since it takes precautions into account when the feature values vary across distinct dynamic ranges. Aspects with high values have a significant impact on the classifier's design if normalization is not employed.

Visualize Class (Check data balance): Since it takes multiple dynamic ranges of the feature values into consideration, this step is frequently taken before designing the classifier. The architecture of the classifier is significantly impacted by characteristics with big values if normalization is not employed. Since all values must fall inside a certain range for a feature to be normalized, this is the function of normalization. In order to check the possibility of balance in our dataset, we visualized the dataset and checked the balance. Since our dataset is labeled as regular Benign code and malware, we check the balance of each class in our dataset. Figure (4) shows that our dataset is entirely balanced.

Identification, description, and implementation of an approach to extract such characteristics are necessary for feature extraction. The main idea that illustrated the Features Extraction method is based on removed features with only one value. Feature extraction requires identifying and describing the features that are relevant to a given problem and implementing a way to extract those features.


Load dataset
CIC-MalMem-2022
Preprocessing stage
Check for None and Inf Values
Visualize Class (Check data balance)
View All Features and Values range
Features Extraction
End
Fig. 3. Preprocessing Steps

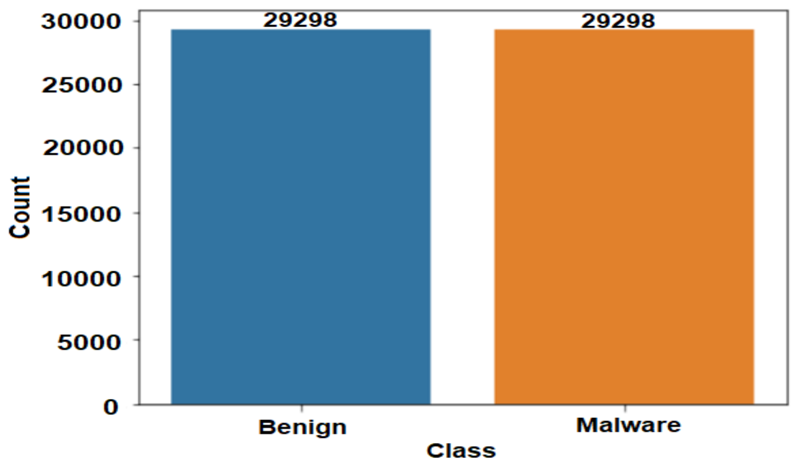


Fig.4. Balanced dataset.

4.2 Datamining Approach

In this paper, Data mining validated as a method for selecting essential variables identified through ICA will convert the large dataset to a lower-dimensional dataset. This is done by calculating and placing the attributes in ranks based on their importance by the ICA. It can do so via the following methods: Using a covariance matrix with eigenvalue and eigenvector to calculate and rank attributes based on their importance. Once this is achieved, the most important ones (k) will be selected, and the k of the attribute's dataset will be converted into the actual dataset (Sarkar & Lee, 2021).

4.2.1 ICA based on Covariance Matrix

The initial tendency in this work was to employ ICA as feature extraction rather than the more conventional ICA. This is because the proposed detection vision considers the connection session stream's time. ICA should be utilized because of the vast number of features that would take time during the training phase and real-time detection due to the (29298) features. The idea seeks to reform the authority to achieve this goal. The dataset's best subset of the features will be given in the set of features that emerges from that extraction. In order to create the classifier, both the original and extracted sets of features will be used.

4.3 Data Mining Traditional Decision Tree (TD) Classifier

Training (learning) is the initial stage in the categorization process. Any given record (pattern) can be predicted (mapped), and the associated class label can be determined. With a training node (training dataset) as the root, a costume ID3 classifier will create a decision tree. According on the "highest GR" feature value, it would then branch into numerous child nodes (sub-datasets). Instead of eliminating the feature, the values of such feature will be set to -1 in each node. The division process repeats itself utilizing the newly resulted nodes up until it chooses one of the nodes (a newly resulted node or the root node) to be divided. Sequential calculations resulted in the numbers shown below:

The *"number of classes"* in a node., Its *"initial entropy*" in a node. and, relying on the result of previous computations, computing the "*InfoGain"* and *"GR*" of the used features in a node.

The resulting child nodes from every division equal the values' number for the highest GR value features that belong to the parent node. It also takes into consideration the branch that exists between the parent and child nodes. Both are labeled with the feature's name and corresponding records' value at the child node.

The division (splitting) keeps going up to the point when all of a node's records are labeled into a single class or the node runs out of features to split according to the available values. Once the division comes to a halt, a newly formed "top-down" path is created that begins from the tree's root node, and in every leaf node, it relays a label that has the value of the branches between them. This path is made of a number of feature-value pairs. They are concluded with a class label. This path's set represents the if-the-else" rules (rules of classification) employed to classify Testing dataset records.

Every class-based dataset has some essential attributes that are highly weighted and connected. They employ the suggested feature extraction approach via ICA. The classes' important attribute subsets are combined to produce a conformed and final optimized attributes subset. Attributes of high weight depend on the training and learning datasets of ID3, which can be had via the algorithm of extracting the proposed feature. This is done so as to create a set of rules that classify connection sessions. ID3 classifier is tested alongside the testing datasets to use the accuracy of the results obtained to evaluate the ID3.

5. Experiment and Results

working machines, we evaluated our efforts identifying and learning from a variety of user tasks. The process of significant feature selection, which creates a prioritized list of features, has been applied to the generated data in this part. On the basis of computing the information gain parameter, this approach has been implemented. Highlights of the data visualization dataset dimensions are: (58596, 57). Figure (5) shows the subclass of the dataset.

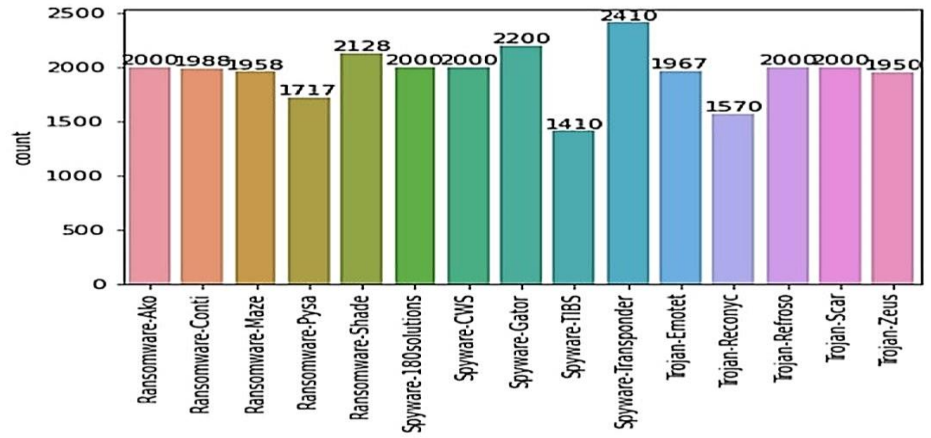


Fig. 5 Subclass of The Dataset.

To assess the effectiveness of trained classifiers, the 5-fold cross-validation approach has been used to get optimal, accurate results then we find the perfect one to divide the data into 80 percent train and 30 percent research for testing and training.

Building and preparing the malware databases. Sub-datasets based on classes are further divided into training and testing datasets. There is a sub-dataset that serves as a representation for each malware class We looked into the relationships between the fundamental components of features and the value of feature information obtained during the analysis of a few key significant features. The most important content, whose study has been found to have a considerable impact on classifier accuracy. Figure (6) shows the two types of cluster that can ICA identify to be applied for training phase in DT classifier.

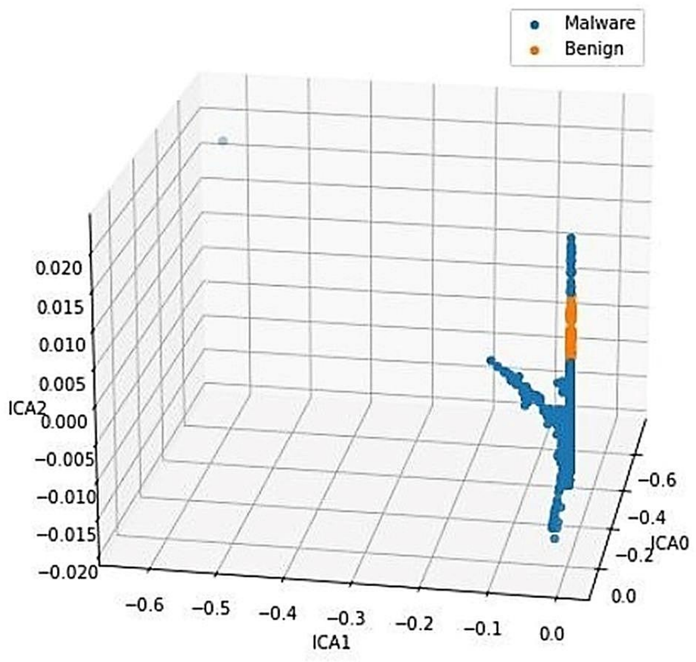
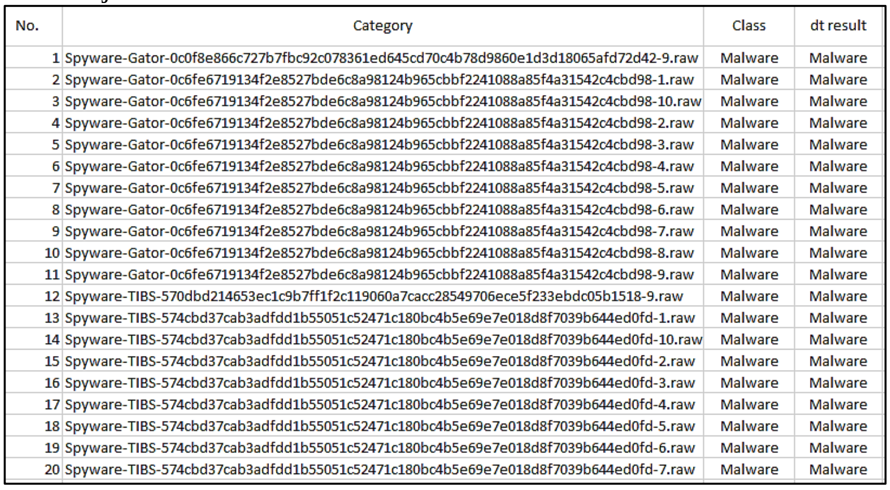


Fig. 6. ICA Clusters

Malware detection models developed using the provided methodology can be used in conjunction with other heuristic analysis techniques for preliminary, quick examination. Figure (7) shows an example of our database that can identify the malware code and features most importantly with more significant results. Our model can identify 52 malware features naming 33 spyware and 19 trojan horse.



No.	Category	Class	dt result
1	Spyware-Gator-0c0f8e866c727b7fbc92c078361ed645cd70c4b78d9860e1d3d18065afd72d42-9.raw	Malware	Malware
2	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-1.raw	Malware	Malware
3	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-10.raw	Malware	Malware
4	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-2.raw	Malware	Malware
5	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-3.raw	Malware	Malware
6	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-4.raw	Malware	Malware
7	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-5.raw	Malware	Malware
8	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-6.raw	Malware	Malware
9	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-7.raw	Malware	Malware
10	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-8.raw	Malware	Malware
11	Spyware-Gator-0c6fe6719134f2e8527bde6c8a98124b965cbbf2241088a85f4a31542c4cbd98-9.raw	Malware	Malware
12	Spyware-TIBS-570dbd214653ec1c9b7ff1f2c119060a7cacc28549706ece5f233ebdc05b1518-9.raw	Malware	Malware
13	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-1.raw	Malware	Malware
14	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-10.raw	Malware	Malware
15	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-2.raw	Malware	Malware
16	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-3.raw	Malware	Malware
17	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-4.raw	Malware	Malware
18	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-5.raw	Malware	Malware
19	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-6.raw	Malware	Malware
20	Spyware-TIBS-574cbd37cab3adfdd1b55051c52471c180bc4b5e69e7e018d8f7039b644ed0fd-7.raw	Malware	Malware

Fig. 7. An Example of Malware Features And Recognition.

We implement the confusion matrix to evaluate our model, as shown in Figure (8). Most importantly, our model works effectively to identify the malware features based on a low false positive rate.

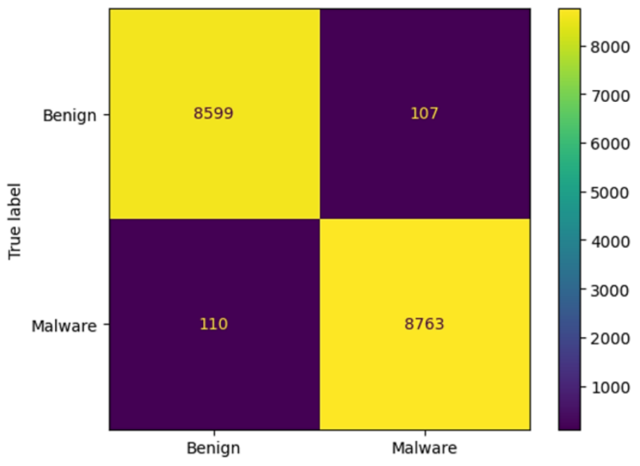


Fig 8. Confusion Matrix of Our Model.

Discussion

This section compares the effectiveness of the proposed cloud-based malware detection system to the top approaches described in the literature. The suggested solution effectively detects a variety of malware families and types, including newly emerging and unheard-of malware as introduced in pervious section.

Table (1) portrays the research for identifying malware. The accuracy ratios in this study are related to the approaches utilized in training to find the most accurate strategy. This includes the name of the author of each researcher, the name of the study, the technique or algorithm employed in each study, and the accuracy findings, along with a false positive rate. all comparison work based on the same data set are used multiple ML approaches and deep learning, then compared between them since our model integrates the data mining dt id3 to identify known malware and recognize the class of attacks.

Table 1 - A Comparison Of The Chosen Researchers

Rf	Method used	Performance	Dataset
Smith *et al.* (2023a)	Decision Tree, Random Forest, Ada Boost, KNeighbors, Stochastic Gradient Descent, Extra Trees, and Gaussian Naïve Bayes	99% with a high false positive rate	CICMalMem-2022
Talukder *et al.* (2023)	Random forest Decision Tree, CNN,ANN	99% with an 11 average for a false positive rate	CICMalMem-2022
Louk & Tama (2022)	Analysis of PE malware by using XGBoost, CatBoost, GBM, and LightGBM	99% with specific malware attacks	CICMalMem-2022
Dener *et al.* (2022)	Random Forest, Decision Tree, Gradient Boosted Tree, Logistic Regression, Naive Bayes, Linear Vector Support Machine, Multilayer Perceptron, Deep Feed Forward Neural Network, and Long Short-Term Memory algorithm	Gradient Boosted Tree follows the Logistic Regression algorithm with 99.94% accuracy.	CICMalMem-2022
Smith *et al.* (2023b)	uses three clustering algorithms for analysis, namely K-Means, Density-Based Spatial Clustering	99%	Malware-Exploratory and CICMalMem-2022
Proposed system	Decision Tree	99%	CICMalMem-2022

5. Conclusion

The study in the article demonstrates that application positionally dependent characteristics that are derived throughout the proposed model phases are adequate for malware detection. The study has provided evidence for the need for malware detection to consider the importance of certain determined and observed feature sections by using Data Mining methodologies based on decision trees. The detection of malware cannot be guaranteed to be 100% accurate with the suggested method. However, because of the qualities malware features listed in the paper as a results taken, Important malware assaults are elucidated and identified using DT classifier.

For future work it is possible to decide on a procedure of action for additional processing to make a diagnosis. The proposed method enables the provision of a precise automatic process for developing detection adaptive rules, which provides an opportunity to select a method of subsequent feature analysis more accurately.

References

Abdulhameed, A. A., Al-Azawi, R. J. & Al-Mahdawi, B. M. (2020). Modeling Web Security Analysis Attacks with CySeMoL Tool. *Al-Mustansiriyah Journal of Science*, *31*(3), 101–109. https://doi.org/10.23851/mjs.v31i3.876

Abusitta, A., Li, M. Q. & Fung, B. C. M. (2021). Malware classification and composition analysis: A survey of recent developments. *Journal of Information Security and Applications*, *59*(April), 102828. https://doi.org/10.1016/j.jisa.2021.102828

Ahmed, Y. A., Koçer, B., Huda, S., Saleh Al-rimy, B. A. & Hassan, M. M. (2020). A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *Journal of Network and Computer Applications*, *167*, 102753. https://doi.org/10.1016/j.jnca.2020.102753

Alagrash, Y., Badih, H. & Rrushi, J. (2020). Malware Detection via Machine Learning and Recognition of Non Stationary Tasks. *Proceedings - IEEE 18th International Conference on Dependable, Autonomic and Secure Computing, IEEE 18th International Conference on Pervasive Intelligence and Computing, IEEE 6th International Conference on Cloud and Big Data Computing and IEEE 5th Cybe*, 606–611. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00106

Alagrash, Y., Mohan, N., Gollapalli, S. R. & Rrushi, J. (2019). Machine learning and recognition of user tasks for malware detection. *Proceedings - 1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2019, February 2020*, 73–81. https://doi.org/10.1109/TPS-ISA48467.2019.00018

Arabo, A., Dijoux, R., Poulain, T. & Chevalier, G. (2020). Detecting ransomware using process behavior analysis. *Procedia Computer Science*, *168*(2019), 289–296. https://doi.org/10.1016/j.procs.2020.02.249

Aslan, O. & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. *IEEE Access*, *8*, 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

Belaoued, M., Boukellal, A., Koalal, M. A., Derhab, A., Mazouzi, S. & Khan, F. A. (2019). Combined dynamic multi-feature and rule-based behavior for accurate malware detection. *International Journal of Distributed Sensor Networks*, *15*(11). https://doi.org/10.1177/1550147719889907

Choudhary, S. P. & Vidyarthi, M. D. (2015). A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining. *Procedia Computer Science*, *54*, 265–270. https://doi.org/10.1016/j.procs.2015.06.031

Dener, M., Ok, G. & Orman, A. (2022). Malware Detection Using Memory Analysis Data in Big Data Environment. *Applied Sciences (Switzerland)*, *12*(17). https://doi.org/10.3390/app12178604

Fasano, F., Martinelli, F., Mercaldo, F. & Santone, A. (2019). Energy consumption metrics for mobile device dynamic malware detection. *Procedia Computer Science*, *159*, 1045–1052. https://doi.org/10.1016/j.procs.2019.09.273

Galal, H. S., Mahdy, Y. B. & Atiea, M. A. (2016). Behavior-based features model for malware detection. *Journal of Computer Virology and Hacking Techniques*, *12*(2), 59–67. https://doi.org/10.1007/s11416-015-0244-0

Gardiner, J. & Nagaraja, S. (2016). On the security of machine learning in malware C&C detection: A survey. *ACM Computing Surveys*, *49*(3), 1–38. https://doi.org/10.1145/3003816

Glanz, L., Amann, S., Eichberg, M., Reif, M., Hermann, B., Lerch, J. & Mezini, M. (2017). CodeMatch: obfuscation won't conceal your repackaged app. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *2017-Janua*, 638–648.

https://doi.org/10.1145/3106237.3106305

Hassan, F. O., Samir, N. M. & Hanapi, Z. M. (2023). Impacts of Denial-of-Service Attack on Energy Efficiency Pulse Coupled Oscillator. *Baghdad Science Journal*, *20*, 1817–1824. https://doi.org/10.21123/bsj.2023.7161

Hataba, M., Sherif, A., Mahmoud, M., Abdallah, M. & Alasmary, W. (2022). Security and Privacy Issues in Autonomous Vehicles: A Layer-Based Survey. *IEEE Open Journal of the Communications Society*, *3*(April), 811–829. https://doi.org/10.1109/OJCOMS.2022.3169500

Huang, X., Ma, L., Yang, W. & Zhong, Y. (2021). A Method for Windows Malware Detection Based on Deep Learning. *Journal of Signal Processing Systems*, *93*(2–3), 265–273. https://doi.org/10.1007/s11265-020-01588-1

Hwang, J., Kim, J., Lee, S. & Kim, K. (2020). Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques. *Wireless Personal Communications*, *112*(4), 2597–2609. https://doi.org/10.1007/s11277-020-07166-9

Jerlin, M. A. & Marimuthu, K. (2018). A New Malware Detection System Using Machine Learning Techniques for API Call Sequences. *Journal of Applied Security Research*, *13*(1), 45–62. https://doi.org/10.1080/19361610.2018.1387734

Kim, H., Kim, J., Kim, Y., Kim, I., Kim, K. J. & Kim, H. (2019). Improvement of malware detection and classification using API call sequence alignment and visualization. *Cluster Computing*, *22*, 921–929. https://doi.org/10.1007/s10586-017-1110-2

Komashinskiy, D. & Kotenko, I. (2010). Malware detection by data mining techniques based on positionally dependent features. *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, PDP 2010*, 617–623. https://doi.org/10.1109/PDP.2010.30

Louk, M. H. L. & Tama, B. A. (2022). Tree-Based Classifier Ensembles for PE Malware Analysis: A Performance Revisit. *Algorithms*, *15*(9), 1–15. https://doi.org/10.3390/a15090332

Muhamed, S. J. (2022). Detection and Prevention WEB-Service for Fraudulent E-Transaction using APRIORI and SVM. *Al-Mustansiriyah Journal of Science*, *33*(4), 72–79. https://doi.org/10.23851/mjs.v33i4.1242

Norouzi, M., Souri, A. & Samad Zamini, M. (2016). A Data Mining Classification Approach for Behavioral Malware Detection. *Journal of Computer Networks and Communications*, *2016*, 20–22. https://doi.org/10.1155/2016/8069672

Pan, Y., Ge, X., Fang, C. & Fan, Y. (2020). A Systematic Literature Review of Android Malware Detection Using Static Analysis. *IEEE Access*, *8*, 116363–116379. https://doi.org/10.1109/ACCESS.2020.3002842

Qin, X. C., Dong, C. Y., Wang, F. & Qu, X. Y. (2017). Static and dynamic analyses of isogeometric curvilinearly stiffened plates. *Applied Mathematical Modelling*, *45*, 336–364. https://doi.org/10.1016/j.apm.2016.12.035

Ranveer, S. & Hiray, S. (2015). Comparative Analysis of Feature Extraction Methods of Malware Detection. *International Journal of Computer Applications*, *120*(5), 1–7. https://doi.org/10.5120/21220-3960

Singh, J. & Singh, J. (2020). Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. *Information and Software Technology*, *121*, 106273. https://doi.org/10.1016/j.infsof.2020.106273

Smith, D., Khorsandroo, S. & Roy, K. (2023a). Supervised and Unsupervised Learning Techniques Utilizing Malware Datasets. *2023 IEEE 2nd International Conference on AI in Cybersecurity, ICAIC 2023*. https://doi.org/10.1109/ICAIC57335.2023.10044169

Smith, D., Khorsandroo, S., & Roy, K. (2023b). Leveraging feature selection to improve the accuracy for malware detection. In *Research Square*. https://doi.org/10.21203/rs.3.rs-3045391/v1

Souri, A. & Hosseini, R. (2018). A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-Centric Computing and Information Sciences*, *8*(1). https://doi.org/10.1186/s13673-018-0125-x

Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., Alharbi,

F. & Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, *72*. https://doi.org/10.1016/j.jisa.2022.103405