IRAQI Academic Scientific Journals
العراقية المجلات الأكاديمية العلمية
University of Kut Journal
ISSN (E): 2616 - 7808 II ISSN (P): 2414 - 7419
www.kutcollegejournal.alkutcollege.edu.iq
k.u.c.j.sci@alkutcollege.edu.iq
Special Issue – The 8th Scientific Conference for Administrative and Economic Sciences – 28 – 29 June 2025

Enhancing Malware Detection through the Utilization of the Isolation Forest Algorithm

Doaa Talib Zaidan [1] , Ali kareem Abed [2] , Hasanian Ali Thuwaib [3]

Abstract

The Internet's widespread design makes it easy for malware to propagate yet protecting against it is challenging. To combat this issue, machine learning-based malware detection models may improve, however, their detection rates vary depending on the malware they find and how they classify it. In addition, the effectiveness of various machine learning algorithms for malware detection might vary depending on the adequacy of their classifiers, even when using an appropriate training dataset. A method for identifying malicious software is suggested in this research. This method combines an isolation forest with a machine learning methodology to identify dangerous software and harmless files. This study also suggests voting methods for major decisions. By using many decision trees, the isolation forest searches for outliers in the data. In other words, it doesn't need labeled data for model training. This strategy selects a parameter at random and then splits the information across extremes. The decision tree will then go through this procedure until either all possible divides in the data have been created, or a maximum number of splits has been reached. Anomalies and outliers may be more easily isolated and sorted out of the data if they are spotted early on. The KISA CISC2017 dataset is used to conduct tests on the suggested methodology. In an experiment using 96,724 out-of-the-ordinary samples, was introduced to the training, an accuracy of 0.98 was observed.

Keywords: Malware, Malicious software, Malware detection, Machine learning, Isolation Forest

Affiliation of Authors
[1, 2, 3] Department of Computer Techniques Engineering, Technical Engineering College, University of Kut, Iraq, Wasit ,52001

[1]doaa.almosawi@alkutcollege.edu.iq
[2]pad.24.022@alkutcollege.edu.iq
[3]Hasaneen.a.dweeb@alkutcollege.edu.iq

[1] **Corresponding Author**

Paper Info.
Published: Aug. 2025

تعزيز اكتشاف البرامج الضارة من خلال استخدام خوارزمية غابة العزلة

م. دعاء طالب زيدان [1] ، م. م. علي كريم عبد [2] ، م. م. حسنين علي ذويب [3]

انتساب الباحثين
[1،2] ، [3] قسم هندسة تقنيات الحاسبات، كلية الهندسة التقنية، جامعة الكوت، العراق، واسط، 52001

[1]doaa.almosawi@alkutcollege.edu.iq
[2]pad.24.022@alkutcollege.edu.iq
[3]Hasaneen.a.dweeb@alkutcollege.edu.iq

[1] **المؤلف المراسل**

المستخلص

يُسهّل التصميم الواسع للإنترنت انتشار البرامج الضارة، إلا أن الحماية منها تُشكّل تحديًا. ولمواجهة هذه المشكلة، قد تتحسن نماذج الكشف عن البرامج الضارة القائمة على التعلم الآلي، إلا أن معدلات اكتشافها تختلف باختلاف البرامج الضارة التي تعثر عليها وكيفية تصنيفها. إضافةً إلى ذلك، قد تختلف فعالية خوارزميات التعلم الآلي المختلفة للكشف عن البرامج الضارة باختلاف مدى ملاءمة مُصنّفاتها، حتى عند استخدام مجموعة بيانات تدريب مناسبة. يقترح هذا البحث طريقةً لتحديد البرامج الضارة. تجمع هذه الطريقة بين غابة عزل ومنهجية تعلم آلي لتحديد البرامج الضارة والملفات غير الضارة. كما تقترح هذه الدراسة أساليب تصويت لاتخاذ القرارات الرئيسية. باستخدام العديد من أشجار القرار، تبحث غابة العزل عن القيم المتطرفة في البيانات. بمعنى آخر، لا تحتاج إلى بيانات مُصنّفة لتدريب النموذج. تختار هذه الاستراتيجية مُعاملًا عشوائيًا، ثم تُقسّم المعلومات بين الحدود القصوى. ستمر شجرة القرار بعد ذلك بهذا الإجراء حتى يتم إنشاء جميع التقسيمات الممكنة في البيانات، أو يتم الوصول إلى أقصى عدد من التقسيمات. يمكن عزل الشذوذات والقيم المتطرفة وفرزها من البيانات بسهولة أكبر إذا تم رصدها مبكرًا. تُستخدم مجموعة بيانات KISA CISC2017 لإجراء اختبارات على المنهجية المقترحة. في تجربة استخدمت 96,724 عينة غير عادية، تم إدخالها في التدريب، لوحظت دقة قدرها 0.98.

معلومات البحث
تاريخ النشر: آب 2025

الكلمات المفتاحية : البرمجيات الخبيثة، اكتشاف البرمجيات الخبيثة، التعلم الآلي، غابة العزلة

1. Introduction

Malware assaults, which are among the most challenging attacks due to their many destructive and intrusive functions, have grown more common in recent years, having been carried out by groups hired by governments, corporations, and other such entities. In recent years, there has been a dramatic rise in the frequency of malware infections. It is estimated that in 2020 alone, more than 1.1 billion malicious programs were published, with over 89 million designed to target the Microsoft Windows operating system alone [1]. As more information becomes available online, the challenge of detecting malware is becoming more crucial. Web users are susceptible to non-executable files like Word and Hangul Word Processor files because they often open them without thinking [2]. The proliferation of malware types poses a significant new danger to cybersecurity because of the potential harm they may do to computers [3]. This is especially true when newly infected non-executables start developing. Methods for identifying new forms of malware have been offered several times. However, owing to the dynamic nature of malware types causing idea drift, reliable identification is difficult. While current malware detection systems use a mapping based on past malware attributes to categorize new malware, adversarial software may still evade detection [4]. These kinds of tactics to prevent detection are often utilized in assaults using packers, and the attacks themselves are called automated poisoning attacks. Attackers may compromise users' banker-based anonymity and utilize it in subsequent voice phishing attempts using this method. Over the last three years, the percentage of ransomware used for financial benefit has steadily and quickly grown. Both of these malware families have been linked to incidents of monetary fraud and information theft. They spread through several channels, including email and the Internet, and the amount of newly discovered viruses is staggering daily [5].

The existing issues have been addressed in a variety of ways, each with its own set of benefits; however, in this research, machine learning has been utilized using Isolation Forest Feature Selections to evaluate and identify malware in the system, thereby addressing the issues listed above.

The rest of this paper is organized as follows: Section 2 reviews the related works. Section 3 describes the proposed approach and Section 4 evaluates it. Section 5 concludes the research.

2. Related Works

By enhancing work using SNORT sub-signatures technique uses a simple method for accurately detecting malware by making use of a feature space with as few parameters as possible, and making it such that this method applies both to host-based and network-based application layers [6]. This research makes use of a method known as an intrusion detection system (IDS) to recognize malicious software on a host system.

To fully grasp the DGA domains, methods of machine learning such as multiple feature extraction, classification, clustering, and prediction algorithms were used [7]. In addition to this, it creates a model of a deep neural network (DNN) that may be used for the categorization of a huge dataset. The Domain Name System (DNS) is an essential component for the successful resolution of domain name lookups in networked applications and services. The researcher creates a DNN model and first offers a machine learning framework for recognizing and preventing DGA malware to

categorize the rising pile of DGA domain data. This allows the researcher to classify the growing pile of DGA domain data. Following this, an empirical assessment of the framework is carried out by contrasting several machine learning methods with a deep learning model. It is strongly suggested that a Hidden Markov Model (HMM) be used for DGA prediction so that DGA domains may be established with a higher level of precision.

Using a combination of feature selection and Principal Component Analysis, proposed an approach using artificial neural networks (ANN) for the detection of malicious software [8]. The PCA testing revealed that the model with 32 principal components has a training accuracy that is nearly equivalent to that of the model with 48 original features, but it requires less time to learn and accounts for a lesser number of dimensions. Additionally, it requires less time to learn than the model with 48 original features. The findings of the tests provide evidence that the model is both effective and superior by demonstrating that it can achieve a high true positive rate while maintaining a low false positive.

IN [9] proposed a machine learning-based malware detection and threat reporting methodology called MalDy. The core principle that drives MalDy's potential to expand to a great extent is the modeling of behavioral data as sequence Words using cutting-edge Natural Language Processing (NLP) and Machine Learning (ML) techniques. It is recommended that the NLP bag of words (BoW) paradigm be used while producing behavioral reports. According to the results of the study, MalDy is an efficient and adaptable tool that can be used in a wide range of analytical and deployment contexts.

In [10] a faster RCNN (Regional Convolutional Neural Network) was proposed using transfer learning as one of its methods for knowledge acquisition. Visualization methods are used both to get information about the classification of different types of malware and to change harmful code into images that have similar textural features. The RCNN model goes through some preparatory processing and is then trained on the ImageNet dataset before it is delivered to the malicious application.

While Dilhara's (2021) research using the Gaussian Nave Bayes method emphasized CNN and RNN; it was also clear that SVM had a file [11]. An undeniable benefit in comparison to the Gaussian Nave Bayes approach. In addition, the higher false positive and false negative rates that were shown by the Gaussian Nave Bayes approach suggest that it is not an appropriate choice for the classification of malware. This demonstrates how powerful Deep Learning algorithms can be when used in the appropriate context.

The concept of machine learning underpins the model proposed In [5]. A random Forest classifier is used in conjunction with a deep learning model that has 12 hidden layers to effectively differentiate between malicious and safe software. This form has certain voting rules that are encouraged to be employed, and those rules have the potential to be used to establish binding judgments. An experiment with 6395 data was conducted, and this hybrid decision model performed better than when it came to recognizing outliers.

Using a generic malware dataset and a non-public malware dataset, In [12] a basic malware detection model to evaluate the feasibility of the proposed technique. The researchers suggest using several different specialized models in conjunction with

one another to maintain coverage for the detection of malware and to ensure rapid replies to actual positive discoveries. A large number of separate categorization models were trained by this work.

In [13] was develop a model was developed that, given just the name of a non-executable file, can detect whether or not the file in question contains dangerous bytes. The researchers utilize byte streams from a variety of file formats so that the model that is used to classify file types may be trained.

With the help of MAPAS, a malware detection system with high accuracy and scalable uses of computational In[14] was recommended using a detection system for malware. To determine whether or not an application is engaging in malicious activity, MAPAS analyses API call graphs and makes use of convolutional neural networks to recognize patterns in API requests that are performed by malicious software.

These efforts refine key facets and boost the predicted accuracy of malware identification. Some studies have aimed to improve accuracy, while others have provided a more comprehensive data collection. Much of the research that has been done to improve recognition rates has used a combination of methods. Finally, the ever-increasing frequency of cyberattacks makes it very necessary to carry out actions with the utmost precision while still maintaining a high rate of speed.

3. Proposed Approach

To develop a malware detection system, this research made use of both the isolation forest (iForest) and machine learning techniques.

In terms of both its performance and accuracy, iForest surpasses the vast majority of the other outlier detection (OD) approaches that are currently available across a variety of datasets. The vast majority of alternative OD algorithms begin by attempting to construct a profile of "normal" occurrences before moving on to the process of locating examples that do not fit this definition. iForest can single out data that is anomalous because it makes use of the characteristics that define anomalies, such as values that are out of the norm for the variables that are being investigated. Since it does not depend on any density- or distance-based metrics to discover anomalies, iForest is a fast and computationally inexpensive anomaly detection method. In addition, iForest makes little use of available RAM (i.e., low overhead).

In addition, a threshold limit is used to facilitate the capability of distinguishing between a piece of dangerous software and a safe piece of software. Nevertheless, since identifying irregularities in malware is not an issue that can be reduced to zero or first, the accuracy of the system will suffer as a result. For example, a malware detection system that is based on a threshold limit will not be flexible when the degree of similarity between normal data and abnormal data is high and when the properties of both types of data are, to some extent, comparable. Because of this, the strategy of isolating forests was used in this investigation to solve this difficulty.

It is feasible to attain high performance in perturbation detection by combining the isolation forest strategy with machine learning; however, to do so, these two techniques need to be appropriately tuned. The ultimate goal of this project is to improve diagnostic accuracy while simultaneously lowering the incidence of diagnostic errors. The majority of the currently available methods for malware detection depend

on a single classifier. To properly conduct this research, it is essential to make use of a significant number of different classifiers. In this way, the chance of a classifier making an error is reduced when many classifiers are employed in a training set and the outputs of these classifiers are merged.

The vast majority of learning algorithms use random search as their primary method of data collection because, in many contexts, it provides the best training for the classifier. As a consequence, it is possible to come up with an approximation of the perfect classifier by merging several different classifiers.

The general architecture of the proposed approach is shown in Fig. 1.

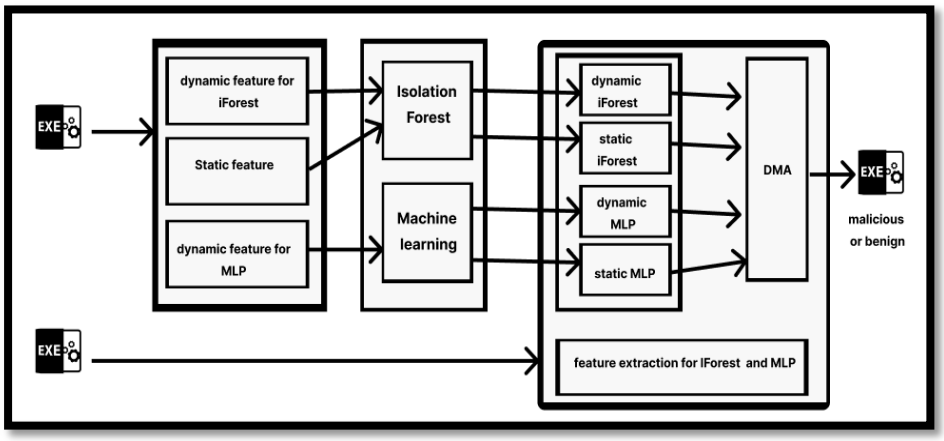

Figure (1): Architecture of the proposed approach

As can be seen in Fig. 1, the detection process is divided into three independent phases, each of which makes use of a different method, either the Multi-Layer Perceptron (MLP), the Isolation Forest (iforest), or both.

The method employs four distinct models of classification, which allows an assessment of the level of harm that has been caused to be carried out. The standard ML methodology utilizes a rating range of [0,1] to indicate the degree of damage, but the method that has been recommended makes use of a rating range of [0,1] to represent the severity of harm that has been done to the original files.

To include the results that were produced by both iforest and MLP, a voting approach is used. This method is employed for both the static and dynamic outcomes. The process of optimization makes use of these findings in many ways. At the beginning of the process of feature extraction, essential static and dynamic features from candidate files are extracted so that the work that has to be done may be finished. In the second step of classification, four distinct classification models are utilized to generate output depending on the characteristics of four distinct feature sets. The model that is being proposed uses a voting system that is based on the majority-rules principle, and it is based on rules to define the values of final preference during the phase of the decision-making process that is immediately before implementation. A simplistic method that is based on majority voting may be able to effectively detect malware; nevertheless, this method may overlook a classifier that separates safe files with dangerous behavior in an accurate manner. A voting system in which the majority always wins and in which the sample is categorized as positive

has been suggested as a means of cutting down on the number of false positives that are reported.

The next stage requires the use of multiple classifiers, and the results of this step are then delivered to the isolation forest. The next step requires the use of multiple classifiers. A heuristic engine is used by iForest to identify whether or not a certain piece of software constitutes a risk to the organization's network.

The bulk of procedures for identifying malicious software may be broken down into two stages:

Beginning The first part of the training is having an introductory conversation about the setup and the parameters of the system operators. The remaining thirty percent of the material will be used in the testing phase of the evaluation of the recommended approach that is being conducted in this inquiry. The results of the testing stage that will be used to evaluate the success of the isolation forest strategy in training the learning model with the typical data will be used. This evaluation will be based on the findings of the stage. The first part of the process is termed training, and it involves applying the isolation forest technique to around 70 percent of the data that is available.

1.1 Extraction of Features and Refinement of Data

It was necessary to do preprocessing on the data to render it usable in the way for which it was designed. This strategy makes use of a one-of-a-kind procedure for the production of data, the specifics of which will be studied in more depth at a later point in time. This is because the vast majority of previous systems relied on methods for feature selection and data preparation to enhance both speed and accuracy.

1.2 Preprocessing

To get the data ready for the analysis. These techniques include doing both static and dynamic analyses of the file that is being uploaded. A code review using static analysis is the first step in the model's process. This step requires creating several static analysis logs, including entropy, active directory entries, and header size, amongst others. The model begins with a big collection consisting of several hundred static properties, which are then whittled down to 79 really important elements. After that, these attributes are converted into quantitative values so that it is simpler to analyze them. For example, the number of times a certain API method is called may be counted, as can the number of distinct directory names that are accessed.

During the dynamic analysis, the file is executed while its behavior is monitored. This includes the activity on the file's network and registry, as well as its main API functions and crucial instructions that are often exploited by malware. This information is also analyzed and converted into numerical values, which, together with the findings of the static analysis, are used to formulate a hypothesis on the likelihood that the file in question contains harmful code.

It is essential to keep in mind that the data have not been normalized into standardized intervals. If this had been done, it might have led to a decrease in the sensitivity of the units that were being measured, which would have made it more difficult to differentiate between safe and dangerous software.

For instance, "ActiveDirectoryEntries" consists of the following: ["Characteristics," "MajorLinkerVersion," "MinorLinkerVersion," "SizeOfCode," "ISizeOfOptionalHeader," and "Characteristics"]; ["Characteristics,"

"MajorLinker Version," and "MinorLinkerVersion"]; ["SizeOfCode," and "IsizeOf"]. The number of features in this model is then lowered until it is comparable to the number of features for static analysis. For static analysis, 79 characteristics are applied, which include registry, network, core API operations, and essential instructions that are regularly exploited by malware. Upon completion of this processing, feature values, which had previously consisted of character data, are converted into numeric values. Rather than relying on symbols to describe the data, this approach uses counting as its primary means of processing it. A record is maintained of both the total number of times a particular API method is called as well as the total number of different directory names that are accessed.

The data set is combed through in search of characteristics that are unique to this model. As an example, the 79 feature values for static models comprise integers such as "feature": ["0", "0", "42929", "382206", "0", "5", "0", "0", "0", "0", "5", "0", "0", "0", "0", "5", "0", "0", "0", "0", "5", "0", "0", "0", "0", "5", "0", "0", "0", Afterwards, the feature values and trained models are used to provide a forecast regarding whether or not a file contains potentially dangerous content.

The data are not normalized into standard intervals for this particular project; rather, extra operations are done on them. This is because the normalization process would take too long. This is done to make the project easier to complete. This replaces the normalization phase that was completed in the earlier steps of the process. It is common knowledge that the process of normalizing the data may result in a reduction in the sensitivity of the units that are being examined, even though these characteristics include essential data bits. Nevertheless, this reduction may occur as a result of the process. On the other hand, it is widely recognized that this loss in sensitivity could not take place at all. Confirming that there is a suitable amount of breathing space between the feature values is of the highest relevance within the context of these circumstances, as it is of the utmost necessity to ensure that there is adequate breathing room between the feature values. During the whole of the analysis process, the feature values are compared to one another. People from different parts of the globe must work together to find a way to combat the threat posed by malicious software (malware). To achieve their goal of neutralizing potentially hazardous software features, attackers may try to trick the system into thinking that certain functions are harmless by disguising them as harmless. This will allow the attackers to achieve their goal of neutralizing potentially hazardous software features. This is done as part of an attempt to reach their aim of neutralizing potentially dangerous software features as quickly as possible. This breakthrough will make it feasible for the attackers to achieve their goal, which is to deactivate potentially harmful software features. The attackers' goal was to disable potentially dangerous software features. In the past, there was a substantial gap in quality between the two of them; however, these kinds of differences are now much more difficult to recognize.

1.3 Machine learning model optimization

It is well known that it may be a tough undertaking to successfully assign the right categorization to harmless files that exhibit suspicious behavior. There is a possibility that one of these files is infected with a virus, which might result in your computer being damaged. Because of this, it has

been determined that single machine learning-based models, which tend to generate high false positive (FP) rates and low true positive (TP) rates, are not suitable for use with this method. This conclusion was reached as a result of the fact that these models have a propensity to generate high FP rates. Since single machine learning-based models tend to yield high FP rates and low TP rates, this conclusion was made as a result of this observation. This discovery was made possible as a direct result of the fact that the models in question had a propensity to generate high FP rates, which in turn made it possible to investigate the occurrence of the phenomena in question. These models had a propensity to generate high FP rates, which led to the realization that this conclusion may be reached from the data. This discovery was made feasible as a consequence of the fact that these models tended to create high FP rates. This realization came about as a result of the fact that these models tended to produce high FP rates, which led to the realization that this conclusion may be drawn from the data. This finding was made possible as a direct result of the propensity of these models to produce high FP rates, which made it possible for this discovery to be made. A model that makes use of deep learning was given some fine-tuning with the primary goal of boosting detection efficiency to raise detection accuracy for data that does not represent a threat. This was done to achieve the primary goal of improving detection for data that does not represent a threat. This was done with the main intention of achieving the aim of increasing detection capabilities for data that does not constitute a danger. This was done to improve the operation's overall efficiency and efficacy as a whole. One of the tactics that was employed to effectively fulfill this mission was to considerably increase the overall number of training data points. This was one of the strategies that was used. A detection success rate that was as high as it was physically feasible was the single most important thing that needed to be done to be successful in this endeavor. The purpose of the study that was carried out was to determine which criteria would be the most helpful in terms of having the most significant effect on the detection rate. This was the purpose of the research that was carried out in the first place. The construction of the deep learning model could be optimized by beginning with the parameters that had been originally given as the defaults. One of the ways in which this goal was successfully achieved was through the utilization of this particular method. These default parameters included, in corresponding order, an optimizer with the name "sgd," an epoch with the value 512, a batch size with the value 2048, and a hidden layer with the value 16; the value of the hidden layer was 16. The dropout was not taken into mind in any of the network's parameters as they were being created, which is another reason why the issue was never dealt with and why it still exists today. This was one of the factors that contributed to the fact that the problem was never addressed. To determine which of the numerous possible configurations for the "node" parameter (the first one) produced the results that were the most favorable, we carried out a series of experiments in which we tested out a large number of various alternatives. The purpose of these experiments was to determine which of the many possible configurations for the "node" parameter produced the most favorable results. Our objective was to locate the most suitable environment and to identify the arrangement that provided the optimal results. It was decided that the ideal value ought to be obtained first, and then that value ought to be

reflected to the node to get the best possible value for the second parameter, which was given the name "epoch." It was decided that the ideal value ought to be obtained first, and then that value ought to be reflected to the node. It was concluded that the ideal value should be acquired first, and then that value should be reflected to the node. The order of these two steps was determined by consensus. After much consideration, it was decided that the ideal value ought to be obtained first, and only then ought that value to be reflected to the node. The sequence in which these two tasks should be completed was decided upon by agreement. This action was carried out as a direct result of the realization that determining the optimal value ought to be prioritized above all other factors to get the best possible results. To determine the optimum values that should be established for each of the hyperparameters, the performance evaluation was carried out individually on each of the hyperparameters one at a time. As a consequence of this, the optimum settings for each of the hyperparameters were able to be determined. Every single one of the configurations for the parameters, including the activation function (which was RELU for the hidden layer and softmax for the output layer), and the cost function (which was cross-entropy), were left in their default states. This included the random weight. This includes every possible setup for the layer that was hidden. To do this, the settings should be left in the majority of their default configurations whenever it is practicable to do so. Experiments were carried out with the assistance of instances obtained from the KISA database to establish which nodes, epochs, batch sizes, and hidden layers provided the most persuasive supporting evidence.

2 Evaluation

The evaluation of the proposed approach is described in the following sub-sections.

2.1 Dataset

The dataset used to evaluate the proposed approach is KISA_CISC2017[1]. It is provided by the Korea Internet & Security Agency (KISA) and consists of 138,047 normal and malicious codes. This dataset is used in a contest to present algorithms and programs that can distinguish between normal code and malicious code to detect it. It can be used for various purposes, such as measuring detection accuracy and improving algorithms.

In the study, 138,047 samples were recruited, with 96,724 used to represent dangerous software and 41323 used to represent benign software. Both static and dynamic analyses were included in the study and a 10-fold cross-check procedure was performed to check for bias. The analyses were performed six times to obtain average results, which were found to be within the norm. Standardization through machine learning compilations was performed for unexplored forests, with each column in the dataset storing a different kind of information. The columns of data are depicted in Table (1).

Table (1): The columns of the dataset

Column Name	Data type
Name	object
md5	object
Machine	int64
SizeOfOptionalHeader	int64
Characteristics	int64
MajorLinkerVersion	int64
MinorLinkerVersion	int64
SizeOfCode	int64
SizeOfInitializedData	int64
OfOptionalHeader	int64
Characteristics	int64
MajorLinkerVersion	int64
MinorLinkerVersion	int64
SizeOfCode	int64
SizeOfInitializedData	int64
OfOptionalHeader	int64
Characteristics	int64
MajorLinkerVersion	int64
MinorLinkerVersion	int64
SizeOfInitializedData	int64
SizOfOptionalHeader	int64
CharacteristicsD	int64
ImageBase	uint64
FileAlignment	int64
MajorOperatingSystemVersion	int64
MinorOperatingSystemVersion	int64
MajorImageVersion	int64
VersionInformationSiz	int64
legitimate	int64
Name	object

2.2 Evaluation Criteria

The precision, recall, and F1-measure were the well-known metrics that were used for the assessment of the proposed approach. The anticipated number of accurate positive replies is used as the retrieval criterion for a system; this statistic is also sometimes referred to as the right positive rate for the system. This metric is used to ascertain the proportion of potentially harmful

applications included within the test set that are accurately identified by the system as being dangerous. The F1-measure is used whenever a machine learning technique is being analyzed for its effectiveness.

2.3 Evaluation Results

The procedure of acquiring 96,724 samples from KISA was a part of the process of performing tests using the upgraded model. The dataset that was employed was unique in that it included harmless files that displayed dangerous behavior. These files, which are often commonly used application files, were the ones that made up the dataset.

Table 2 demonstrates that a precision of 0.98 is achieved in the identification of malware when it is trained using the data from the research. In addition, the random data was added during the training process. This resulted in an F1-measure of 0.71, which demonstrates the accuracy in identifying malware even when the data that is added during training does not exactly match the dataset that is being used.

Table (2): Evaluate results

Data Type	F1-measure	Recall	Precision
Isolation Forest	**97.9%**	**97.8%**	**98%**
Random Data	**71%**	**97.8%**	**72%**

3 Conclusion

In this paper, a method for recognizing malware is presented. Isolation forest was the method that was used to recognize data anomalies, also known as outliers, and separate them from the other data. It is not necessary to make use of labeled data to train the model using it. This approach chooses a parameter at random and then splits the information into two categories based on their extremes. Outliers and anomalies in the data will stand out instantly, making it simpler to spot them and study them independently. The procedure is maintained inside the decision tree until all conceivable splits in the data have been created, or until the maximum number of splits has been attained. In addition, the proposed method made use of hybrid classifiers and voting. The experimental evaluation of this method shows promising results.

4 Future Works

In the future, the combination of the proposed approach with other schemes can be considered. By combining static analysis and hybrid (static and runtime) disassembly schemes, it is possible to extract more data from the binary executable itself about a particular malware instance, improving the accuracy of configuring the software environment that will force the execution of more malware code. Examples of this include binary strings, statistically significant bytes, and instruction patterns.

References

[1] D. Javaheri, P. Lalbakhsh, and M. Hosseinzadeh, "A novel method for detecting future generations of targeted and metamorphic malware based on genetic

algorithm," IEEE access, vol. 9, pp. 69951-69970, 2021.

[2] L. Cavallaro, D. Gruss, G. Pellegrino, and G. Giacinto, Detection of Intrusions and Malware, and Vulnerability Assessment: 19th International Conference, DIMVA 2022, Cagliari, Italy, June 29–July 1, 2022, Proceedings. Springer Nature, 2022.

[3] M. Stamp, M. Alazab, and A. Shalaginov, Malware analysis using artificial intelligence and deep learning. Springer, 2021.

[4] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi, and A. Y. Al-Rezami, "An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning," IEEE Access, vol. 9, pp. 97180-97196, 2021.

[5] S. Yoo, S. Kim, S. Kim, and B. B. Kang, "AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification," Information Sciences, vol. 546, pp. 420-435, 2021.

[6] B. Khammas, "Malware detection using sub-signatures and machine learning technique," Journal of Information Security Research, vol. 9, no. 3, pp. 96-106, 2018.

[7] Y. Li, K. Xiong, T. Chin, and C. Hu, "A machine learning framework for domain generation algorithm-based malware detection," IEEE Access, vol. 7, pp. 32765-32782, 2019.

[8] J. Zhang, "Machine learning with feature selection using principal component analysis for malware detection: a case study," arXiv preprint arXiv:1902.03639, 2019.

[9] E. B. Karbab and M. Debbabi, "MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports," Digital Investigation, vol. 28, pp. S77-S87, 2019.

[10] Y. Zhao, W. Cui, S. Geng, B. Bo, Y. Feng, and W. Zhang, "A malware detection method of code texture visualization based on an improved faster RCNN combining transfer learning," IEEE Access, vol. 8, pp. 166630-166641, 2020.

[11] B. Dilhara, "Classification of Malware using Machine learning and Deep learning Techniques," International Journal of Computer Applications, vol. 183, pp. 12-17, 2021.

[12] Y. Gao, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Malware detection using LightGBM with a custom logistic loss function," IEEE Access, vol. 10, pp. 47792-47804, 2022.

[13] Y.-S. Jeong, S.-M. Lee, J.-H. Kim, J. Woo, and A. R. Kang, "Malware detection using byte streams of different file formats," IEEE Access, vol. 10, pp. 51041-51047, 2022.

[14] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," International Journal of Information Security, vol. 21, no. 4, pp. 725-738, 2022.