Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048883

ARTICLE

Robust Malicious Executable Detection Using Host-Based Machine Learning Classifier

Khaled Soliman[1,*], **Mohamed Sobh**[2] **and Ayman M. Bahaa-Eldin**[2]

[1]Department of Computer and Systems Engineering, Ain Shams University, Cairo, 11517, Egypt

[2]Department of Computer Engineering Technology, ElSewedy University of Technology, Cairo, 44629, Egypt

*Corresponding Author: Khaled Soliman. Email: khaled.solimaan@gmail.com

Received: 21 December 2023 Accepted: 22 February 2024 Published: 25 April 2024

ABSTRACT

The continuous development of cyberattacks is threatening digital transformation endeavors worldwide and leads to wide losses for various organizations. These dangers have proven that signature-based approaches are insufficient to prevent emerging and polymorphic attacks. Therefore, this paper is proposing a Robust Malicious Executable Detection (RMED) using Host-based Machine Learning Classifier to discover malicious Portable Executable (PE) files in hosts using Windows operating systems through collecting PE headers and applying machine learning mechanisms to detect unknown infected files. The authors have collected a novel reliable dataset containing 116,031 benign files and 179,071 malware samples from diverse sources to ensure the efficiency of RMED approach. The most effective PE headers that can highly differentiate between benign and malware files were selected to train the model on 15 PE features to speed up the classification process and achieve real-time detection for malicious executables. The evaluation results showed that RMED succeeded in shrinking the classification time to 91 milliseconds for each file while reaching an accuracy of 98.42% with a false positive rate equal to 1.58. In conclusion, this paper contributes to the field of cybersecurity by presenting a comprehensive framework that leverages Artificial Intelligence (AI) methods to proactively detect and prevent cyber-attacks.

KEYWORDS

Portable executable; malware; intrusion detection; cybersecurity; zero-day threats; Host Intrusion Detection System (HIDS); machine learning; Anomaly-based Intrusion Detection System (AIDS); deep learning

1 Introduction

The recent years have witnessed unprecedented progress in emerging technologies which led to more dependency on technology. This progress was accompanied by new threats such as AI-based cyber-attacks, side-channel attacks, and IoT security threats as discussed in Shaukat et al. [1] and Tariq et al. [2]. New technologies such as quantum computers are also threatening the current cryptographic systems and the replacement of current encryption algorithms to Post-Quantum Cryptography (PQC) would affect various blockchain and smartphone applications as described in Canto et al. [3].

These new threats have driven researchers to find alternative solutions to develop traditional mitigation mechanisms such as Signature-based Intrusion Detection Systems (SIDS). This SIDS approach is no longer sufficient to detect malware programs as files can be easily modified to bypass cybersecurity solutions using unknown signatures. In addition, these infected files can be delivered to devices without being scanned at the network level through various means, such as connecting removable devices or receiving encrypted files through HTTPS and FTPS protocols.

Therefore, there is a great need to develop a Host Intrusion Detection System (HIDS) to discover the threats at the host level and establish a technique to detect any previously unknown malware files. The most efficient technique to detect these Zero-day threats is the ability to monitor the behavior of computers' processes to detect any abnormal behavior generated by malware files.

Abnormal behavior can be tracked through a static or dynamic approach. The dynamic approach is more powerful than the static counterpart as it can detect the impact of running files on the operating system. It does so by executing files and monitoring their changes to the system to decide if it is safe or unsafe behavior on the computer. However, the dynamic approach is always considered a challenge, as it is time-consuming and requires huge resources to execute all running files before being executed in the live environment. The network and email sandboxes are known examples of executing each file in a virtual machine before delivering files to the users. Therefore, the static technique is the only suitable mechanism to detect abnormal activities that could happen due to running these files in the future.

This static approach is called the Anomaly-based Intrusion Detection System (AIDS) which differentiates between the normal and abnormal expected behavior of files as clarified in the survey by Khraisat et al. [4] and Shaukat et al. [5]. The AIDS mechanisms can be used at the host level with static scanning only since there are limited resources at the host level which would prevent dynamic scanning.

One of these major threats requiring behavioral detection is malware programs concealed in packed executable files in the Windows Operating System which prevents malware detection tools from recognizing their malicious content. These files are called Packed Executable (PE) which represents the file types of .exe, .dll, mui, .sys, .efi, and other file extensions. As per VirusTotal statistics [6] shown in Fig. 1, the PE files constitute the highest percentage with 38% of files submitted for investigation among all the other files' extensions from 26th November 2022–11th December 2022. According to Statista reports [7], the executable files represent 59% of malicious file types in 2020.

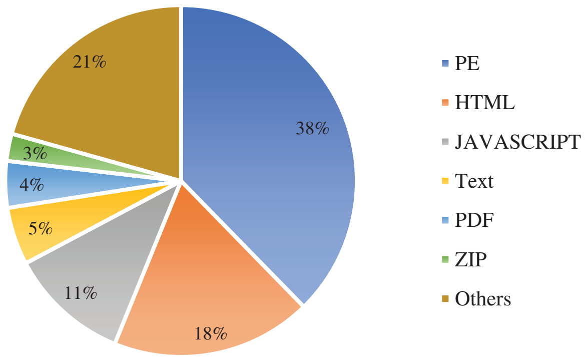

Figure 1: VirusTotal file types statistics

The PE files include a group of headers and sections to map the files inside the memory. As per Microsoft documentation [8], the structure of the PE file header structure includes MS-DOS stub, PE signature, COFF File Header, Optional Header, and Section Headers. The PE headers determine the actions expected from these files while running in the Operating System (OS).

The behavior of these executables can be collected by analyzing the PE headers which can act as an indicator of unknown malware detection at the host level. By tracking the behavior of PE files, this approach can determine whether they are acting normally, or malicious activities are taking place even if these activities are generated from trusted executables in Windows as they can be hijacked by attackers.

This approach can prevent any type of injection for trusted executable files as it can learn the behavior of each executable in Windows files. This behavior can be classified as normal or abnormal by AIDS methods which are statistical-based, knowledge-based, and machine learning mechanisms.

Researchers focused on this threat and discovered that machine learning is the most efficient technique in AIDS to detect such threats using the behavior collected from PE headers as in Soliman et al. [9]. However, there were challenges related to determining the features selected in PE headers and finding an available and viable dataset used to strengthen the machine learning model to prevent advanced cyber-attacks with maximum accuracy and minimum False Positive Rate (FPR).

According to the literature review in this paper, researchers face the following challenges:

– The large number of selected PE headers increases the time of scanning files because the system takes a long time to process all the features required from each file.
– The process of collecting tens or hundreds of PE headers can consume the computers' resources as it requires high processing overheads. As a result, most of the existing research techniques are not practical in a live environment and are not suitable for real-time detection.
– The datasets available are either too limited in the number of normal files or malware samples to train the machine learning models efficiently and test their performance on actual threat scenarios.

Consequently, this paper proposes an innovative approach called RMED to empower the machine learning techniques used at the host level to detect malicious executables efficiently in real-time using a novel dataset of PE headers. It aims to detect unknown malware files and malicious running processes from the behavior of PE headers collected according to their impact on the Windows 10 operating system.

This paper's four major contributions are as follows:

– Proposed a new approach based on a novel combination of PE headers to detect abnormal behavior of executables, showing that the selection of 13 PE headers and 2 other values for Entropy can provide the best classification accuracy which has reached 98.42%.
– Developed a real-time scanning code for Windows 10 endpoints which can scan each file in 91 milliseconds using the proposed machine learning approach based on a random forest classifier.
– Built a new dataset for normal and abnormal executables using the PE headers collected from 116,031 benign files and 179,071 malware samples to test the machine learning algorithms efficiently and improve their accuracies.
– Analyzed the performance of 12 machine learning models and validated the accuracy and false positive rate of RMED with 4 other machine learning and deep learning approaches.

In Section 2, the paper describes the proposed machine learning mechanisms to detect malware files using non-signature-based techniques. Following this, Section 3 explains the steps to build a new dataset of PE headers with selected features and introduces the proposed machine learning approach RMED to detect unknown malware files in real-time. After this, Section 4 evaluates the proposed solution and compares it with the previously proposed approaches according to accuracy, detection time, and FPR. Section 5 discusses the paper's achievements, limitations, and future work. Finally, Section 6 summarizes the paper and highlights the key challenges facing researchers.

2 Literature Review

The role of Artificial Intelligence is increasing tremendously in all fields of work as mentioned in Kamran et al. [10] and cybersecurity is one of the core fields that highly requires intelligent approaches to detect advanced cyber-attacks. The AI-based cybersecurity techniques have been developed exponentially recently to protect against Zero-Day Threats and side-channel attacks as reviewed in Hettwer et al. [11] and Shaukat et al. [12].

This section presents the research approach of this paper and discusses the papers reviewed to highlight the challenges and areas of development. The reviewed studies are divided into two main categories: Machine learning approaches and deep learning approaches.

2.1 Threats to Validity

The research approach of this paper was based on the following criteria (1) cybersecurity and machine learning techniques to detect Zero-Day Attacks, (2) HIDS machine learning and deep learning approaches, (3) PE headers structure and datasets, (4) evaluation for machine learning performance.

The authors have used Google Scholar to search for these topics using various strings such as 'Artificial Intelligence techniques to detect Zero-Day Attacks', 'AIDS techniques using machine learning to detect malicious PE files', and 'PE dataset generation on Windows 10'. The main databases used to review papers were Scopus, ScienceDirect, SpringerLink, and IEEE Xplore.

2.2 Machine Learning Models

Hubballi et al. [13] proposed semi-supervised and anomaly detection approaches to flag packed and non-packed files using certain features. They used a dataset containing 2598 malware samples and 2900 benign samples of Windows in addition to the dataset used in Nappa et al. [14] containing 1591 malware samples. They used 9 features in the PE headers, then tested the approach using Euclidean and Mahalanobis distance measurement, and they reached a 96% classifier accuracy.

Bai et al. [15] extracted 197 features from PE headers and Windows Application Programming Interface (API) and then applied features selection called CfsSubsetEval (filter approach) and WrapperSubsetEval (wrapper approach). They collected 8,592 benign samples from Windows files, program files, and known software applications in addition to 10,521 malware samples. They used 80% as a training dataset and 20% as a testing dataset, which led to an accuracy of 99.1% and an Area Under the Curve (AUC) value of 0.998, and the detection rate for new malware files was 97.6% with 1.3% FPR.

The anomaly detection technique in Ugarte-Pedrero et al. [16] used 13,173 malware samples from VxHeavens, 1,548 from Zeus, and 1,645 benign files. They used 28 features of heuristics and structural features which were based on distance-based anomaly detection techniques with a clustering method proposed. The main advantage of this approach is the ability to build an efficient model without packed samples to detect abnormal behavior, as it does not require gathering a huge number of malware samples. Euclidean and Manhattan distance were used to test the proposed system which achieved 0.9574 as the highest AUC rate from all measures applied.

A different approach was proposed by Yan et al. [17] in which the authors used ReliefF, Chi-square, and F-statistics to select the features of PE headers. The dataset collected included 597 benign files from Windows and 526,179 unique malware files. They built their machine learning model using Naive Bayes, kNN, SVM, and the decision tree. After testing the performance of each technique, they discovered that the decision tree algorithm was the best approach as it could detect all the malware families with high accuracy, excluding the malware types of Rbot and Sdbot.

There was another paper by Syuhada Selamat et al. [18] suggested a machine-learning mechanism based on a decision tree algorithm. The authors extracted 78 features from PE headers and then selected 28 features as per the analysis done by Ranveer et al. [19]. The dataset collected contains 305 types of malware and 236 benign files from Windows system and program files. The best approach used had a detection accuracy of 99% and 0.021% FPR.

An integrated feature set method was proposed by Kumar et al. [20] as a development for Markel et al. [21]. The authors gathered 2488 benign types from freshly installed Windows in addition to 2,722 malware samples from VirusShare. This dataset was divided into 70% for training and 30% for testing. They selected 15 features and the accuracy reached 97.47% while using Random Forest and 98.78% in the case of an integrated feature set. The accuracy for detecting novel malicious PE reached 89.23%.

A new features selection technique was introduced by Belaoued et al. [22] which provides a variable number of PE header features using the Chi-square (KHI^2) score and the Phi (φ) coefficient. The research evaluated the proposed system by building a model using the Rotation Forest classifier. The dataset collected contained 214 benign Windows 10 files and 338 malware samples from Vxheavens. The model results reached 97.25% accuracy and took 0.077 s to classify each checked file.

Manavi et al. [23] proposed an approach to detect ransomware attacks using the header bytes sequence of the PE header. It extracted 256 bytes of features from 7,000 normal files and 7,879 ransomware samples. The authors used a weighted model depending on the features collected then they converted the graphs to feature vectors with the concept of eigenvector and eigenvalues. The eigenvectors indicate the scatter of data and eigenvalues show the variance of eigenvector direction. The extracted features from the last step were used to train a machine-learning model to detect ransomware attacks using a random forest classifier. The model was evaluated using 3 different datasets and the best accuracy was 96.8% with an F1 Score of 0.9676.

There was another approach to select the best-collected features of PE headers by Varma et al. [24] which used a hybrid Rough Set Feature Selection using Cuckoo Search Optimization (RSFSCSO) to find highly effective features to detect malware files. The authors used the Random Forest algorithm to differentiate between benign and malware files using 5 selected features from the CLAMP dataset and the accuracy reached 94.71%.

A new dataset was generated by Kattamuri et al. [25] called SOMLAP (Swarm Optimization and Machine Learning Applied to PE Malware Detection). SOMLAP contained 51,409 samples divided into 31,600 benign and 19,809 malware files. In addition, the paper collected 108 features and reduced them to 12 features using swarm optimization techniques named Cuckoo Search Optimization (CSO), Ant Colony Optimization (ACO), and Grey Wolf Optimization (GWO). The analysis showed that ACO selection with the Decision Tree algorithm was the best approach with an accuracy of 99.37%.

2.3 Deep Learning Mechanisms

The Deep Neural Network technique was used in Divakarla et al. [26]. The model consisted of 9 layers which included one for input, one for output, and 7 hidden layers. A rectified linear unit (ReLU) was used as the activation function for all hidden layers. It also used two regularization methods to reduce overfitting with batch normalization and dropout. The dataset used for this model included 300,000 benign samples, 300,000 malicious, and 300,000 unlabeled. The features collected from the PE header were 7 in addition to the entropy calculated by Shannon's formula. The model used GAN and it was tested against 100,000 benign samples and 100,000 malicious files and showed that the detection accuracy reached 97.42%.

Rezaei et al. [27] proposed a new method for malware detection using a deep neural network supported by K-means and clustering techniques. The authors used 2 datasets to test the classification performance on different data. The first dataset included 2,000 benign samples and 2,000 malicious PE files, while the second dataset contained 4,500 normal Windows files and 4,500 malware files from other sources.

After training the model on 324 bytes of each file, the classification accuracy for the first dataset was 92.41% and the second dataset reached 97.75%. They also tested the approach on metamorphic malware samples with an accuracy of 91.13%.

As shown in Table 1, the results of papers discussed earlier show that the maximum accuracy was 99.37% with a limited dataset including 31,600 benign files and 19,809 malware samples. The other approaches had lower accuracy, and the majority used small datasets in addition to using a large number of features which resulted in a long scan time.

Table 1: Related work performance summary

Ref.	Classification technique	Benign PE dataset size	Malware PE dataset size	Features selected	Model accuracy	AUC	F1 Score
[13]	Semi supervised using Euclidean distance	2,900	2,598	9	96%	NA	NA
[15]	AdaBoost	8,592	10,521	19	97.6%	0.998	NA
[16]	Anomaly detection using Euclidean and Manhattan	1,645	14,721	28	NA	0.9574	NA
[17]	Decision tree	597	526,179	100	NA	NA	0.85
[18]	Decision tree	236	305	28	99%	NA	NA
[20]	Random forest	2,488	2,722	68	98.78%	1	0.99
[22]	Rotation forest	214	338	Variable	97%	NA	NA
[23]	Random forest	7,000	7,879	256	96.8%	NA	0.9676
[24]	DNN	300,000	300,000	8	97.42%	NA	NA
[25]	DNN	6,500	6,500	Variable	97.75%	NA	0.9773
[26]	Random forest	2501	2,683	5	94.71%	NA	NA
[27]	Decision tree	31,600	19,809	12	99.37%	0.993	NA

3 RMED

This section describes the steps taken to generate the new dataset for PE headers and clarifies the selection criteria for each feature. Afterward, it presents the classification approach of RMED and discusses a detailed performance analysis against other approaches.

3.1 Dataset Generation

The cornerstone of any machine learning model is the dataset which proves the quality of any security approach. The dataset required to build a powerful model should include sufficient behavior of PE headers for both normal and malware files. This behavior is used to train the machine learning model to classify any executable file as a benign or malware file according to a set of features as per the clarification discussed in David et al. [28].

The dataset richness is defined by its diversity and quantity to learn the different characteristics of normal and malware samples. The diversity can be achieved by collecting information from diverse desktop and laptop vendors because each hardware vendor has its executable files for the system and drivers. It can also be done by learning the behavior of computers used in different sectors as each sector is using a wide range of trusted applications that would be considered normal behavior to prevent false positive incidents. For malware files, the diversity of malware types can play a significant role in strengthening the machine learning model to discover diverse types of malware samples.

On the other hand, the quantity of the dataset can be enriched by the number of Windows devices used for data collection. The malware volume can highly affect the dataset efficiency to detect the likelihood of those malware executables or any abnormal activity from all running processes on local machines even if their sources are trusted by Windows SmartScreen.

To these requirements, this paper has implemented a program as a development for the code in Git Hub [29] which was designed by Python 3.9 to extract PE headers for the sake of building benign and malware datasets from PE files running on Windows 10. The code uses a well-known Python module named pefile which is used to capture the header of PE files.

As shown in Fig. 2, the code is executed on normal and malware PE files to extract their PE headers which include PE Header, DOS Header, Optional Header, and Section Table. Each variable collected from headers is considered as a running feature for the tested executable which is needed to build the proposed machine learning model of this paper. The values of all features extracted from each file are added in the third phase in a consolidated dataset either for benign or malware datasets. The features collected were 78 variables extracted from each PE file which represent the expected behavior of each executable file while running in the operating system such as size, characteristics, and patterns.

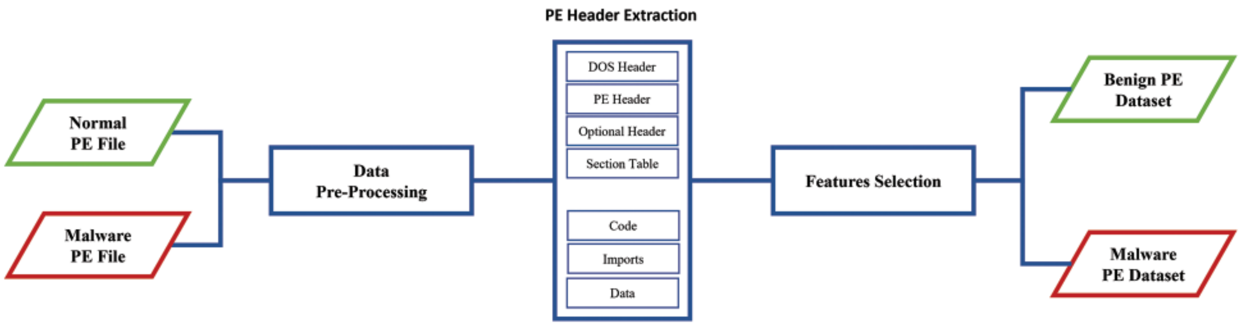

Figure 2: Dataset generation structure

The benign data was collected from 15 computers using freshly installed Windows 10 and selected different vendors to get more varieties of trusted PE files representing drivers' files and operating system internal executables. These machines were also used for different working fields as each device used diverse trusted applications. After collecting the data from those devices, 116,031 unique benign PE files were collected which present normal PE behavior in the dataset (Supplementary Table S1).

For the malware dataset, the same Python code was used on a dedicated Windows 10 machine to generate the PE headers of malware samples collected from VirusShare [30] and the Zoo [31] malware repository. The total number of malware files collected after removing duplicates was 179,071 representing malicious behavior in the dataset (Supplementary Table S2).

3.2 Features Selection

Efficiency is improved by decreasing the number of features to the most important that can act as an indicator of malware behavior. The number of selected features was selected by understanding the role of each feature and testing their impact on the machine learning approach.

The optimization of collected PE features is also applied to simplify the implementation of the proposed dataset which is responsible for classifying files as normal or malware executable according to their headers collected. If those features remain at the same excessive number, the machine learning model will take a long time on Windows 10 devices to collect the features from each executable and classify them.

After testing the classification quality using several machine learning models with diverse features, the selected 15 features had the best results. Those features include default PE headers and a major indicator called Entropy which is being calculated by Claude Shannon's rule below to measure the uncertainty in a set of information.

Table 2 shows the 15 features selected, which include 13 PE headers and 2 other values for Sections Mean Entropy and Resources Mean Entropy. In the analysis and trials conducted, these features are considered highly effective values to differentiate between normal PE files and malware samples.

Table 2: Selected PE features

No.	Selected feature	Source type	Importance
1	Characteristics	Section table	Considered as a sign for a change in the section's flag value to make a specific section as executable
2	Size of code	Optional header standard fields	Indicator for an additional code added by a malware which will reveal that the size is not compliant with expected value
3	Size of initialized data	Optional header standard fields	Pointer for the nonexistence of initialized data which increase malware probability
4	Address of entry point	Optional header standard fields	Discover entry point redirection done by malware files which leads to incompatible flow with their characteristics

(Continued)

Table 2 (continued)

No.	Selected feature	Source type	Importance
5	Base of data	Optional header standard fields	Reveal abnormal behavior for miss match between the beginning-of-data section address and image base
6	Image base	Optional header windows-specific fields	The value of Image Base field must be a multiple of 64 K. Otherwise, there could be a malware impact
7	Number of sections	COFF file header	Check if there are few or excessive number of sections compared to normal behavior of PE files
8	Section alignment	Optional header windows-specific fields	Indicator for abnormal alignment of sections in memory which can be different than the default page size of architecture
9	Size of image	Optional header windows-specific fields	Size of image value must be a multiple of the Section alignment to confirm that there is no adjustment in normal file
10	Size of headers	Optional header windows-specific fields	The total size of headers shows any modifications happened to headers
11	DLL characteristics	Optional header windows-specific fields	Refer to multiple DLL indicators such as relocation at runtime or WDL driver. Normal files are rarely having 0 value
12	Checksum	Optional header windows-specific fields	It is a refence for DLLs calls at boot time or into critical Windows process. Malware files are usually having checksum as 0
13	Sections mean entropy	Non-PE header	Threat indicator for a suspicious data randomness in sections
14	Sections mean rawsize	Section table	Size of initialized data on disk which should be multiple of File Alignment
15	Resources mean entropy	Non-PE header	Threat indicator for a suspicious data randomness in resources

The table shows the source type of each feature and highlights the location of each header within the PE header format. The selected headers included a header from the COFF file header, 2 headers from the Section table, 4 headers from Optional Headers Standard fields, and 6 headers from Optional Header Windows-Specific Fields. Furthermore, it clarifies the importance of each selected feature according to its detected behavior explained in Zatloukal et al. [32].

The following equation Eq. (1) shows that Entropy H(X) is the expected amount of information required to identify a random sample from probability distribution, where X is the set of all possible outcomes and p(x) is the probability of each element of X.

$$H(X) = \sum_{x} p(x) \log p(x) \tag{1}$$

The equation of entropy is used to calculate the values of Sections' Mean Entropy and Resources' Mean Entropy. For Sections, p(x) represents the probability of each possible value occurring within a given X section. While the Resources, p(x) shows the probability of each possible value occurring with a given X resource. The result can be in bytes or characters depending on the type of resource.

Entropy plays an essential role in identifying abnormal behavior because it represents randomness within the data collected from the PE header. It provides a value between 0 and 8 where the high values refer to high randomness and high probability of having a packed and encrypted PE file, while low entropy values refer to regular distribution of information which is considered as a normal file.

3.3 Classification

The proposed system named RMED aims to discover Zero-Day malicious executables in real time using a machine learning mechanism. RMED is an AI-based software developed in Python code (Supplementary Table S3) to detect malicious PE files and running processes using a built-in trained model stored in a pickle file on Windows 10 endpoints.

This study has used the new PE dataset for both normal and malware files to train the machine-learning models after selecting the most effective features. The algorithms used were supervised learning techniques that were applied to PE headers to measure the accuracy and performance metrics of each mechanism.

Both normal and malware datasets were divided into 70% for training the model and 30% to test the model's efficiency for both benign and malware files. Afterward, 10 different machine learning algorithms were used: Logistic Regression (LR), Decision Tree (DT), Linear Discriminant Analysis (LDA), Random Forest (RF), Naive Bayes Gaussian (NB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Extreme Gradient Boosting (XGBoost), Light Gradient-Boosting Machine (LightGBM), and Voting Classifier.

After testing and analyzing the output of Machine Learning algorithms, it has been found that the recommended method is to use the Random Forest algorithm on the trained dataset including the 15 features. As shown in Fig. 3, the RMED approach contains four phases to classify malicious executables. The first phase is inserting the executable sample, and the second phase extracts the PE header content.

Afterward, phase three selects the 15 features mentioned earlier and then inserts them in the machine learning model in phase four to classify the behavior of the executable file inserted. The proposed machine learning model is the RF algorithm as it showed the highest accuracy compared to other Machine Learning and Deep Learning algorithms.

RMED suggested flow can ensure that all EXE, DLL, and other executable files can be scanned using their expected behavior without executing these files in a sandbox environment which can be bypassed as well. The limited number of selected features has simplified the scan process and provided real-time prevention against malware execution through new malicious files or process hijacks that can be used to conceal malware behavior within Windows legitimate processes or trusted applications running in Windows. This improvement can enable cybersecurity providers to run this system in the background to proactively check all the running processes without affecting the performance of Windows.

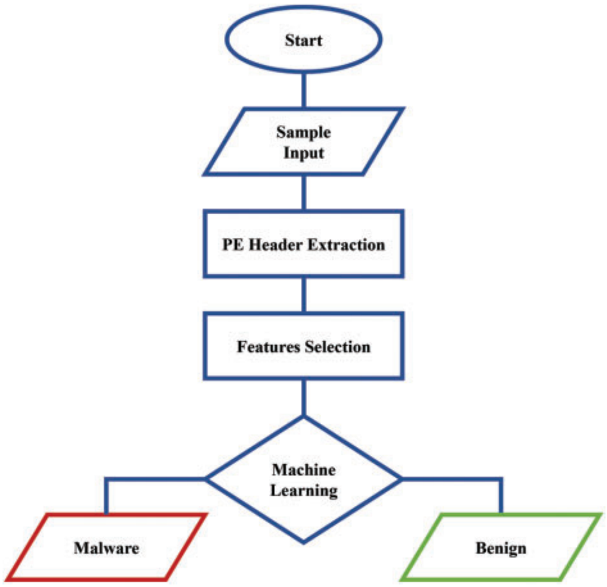

Figure 3: RMED flowchart

4 Evaluation

Several machine learning and deep learning techniques have been used recently in intrusion detection systems to discover sophisticated cyber threats that conventional mechanisms failed to detect. However, these techniques require evaluation methods to ensure their efficiency against adversarial attacks which could manipulate the machine learning models as mentioned in Shaukat et al. [33]. There are a diverse number of metrics and statistical tests to assess the performance of machine learning algorithms as discussed in Shaukat et al. [34]. This section discusses these evaluation aspects and compares the proposed solution against other approaches mentioned in the literature review.

4.1 Performance Metrics

Malware classification techniques are being evaluated by a set of terms indicating the ability of each approach to detect malicious files precisely without providing fake alerts or missing threats. The following metrics in Table 3 elaborate on the role of each metric in evaluating the performance of the malware detection approach.

Table 3: Performance metrics

Name	Description
TP (True positive)	Malware predicted as malware
TN (True negative)	Benign predicted as benign
FP (False positive)	Benign predicted as malware
FN (False negative)	Malware predicted as benign

These terms are used to measure the performance metrics of machine learning models to assess their effectiveness through calculating the below metrics for TPR (True Positive Rate), FPR, ROC (Receiver Operating Characteristics), AUC, precision, accuracy, and F1 Score.

TPR: It is known by sensitivity or recall which measures the performance in a binary classification by counting the rate of positive predicted instances among all malicious executable positive instances.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

FPR: The false predicted malicious executable instances to the total number of benign files inserted in the machine learning model.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \tag{3}$$

Using TPR and FPR, the system accuracy can be measured by ROC which is a graphical representation showing the classification performance by plotting the values of TPR against FPR.

The two-dimensional AUC represents the overall quality of all predicted instances. AUC value summarizes the machine learning performance using a range from 0 to 1 according to performance the value is increasing to show the classification efficiency.

Precision: It shows how the machine learning model has an accurate prediction when it comes to positive instances of malicious executables. Precision value can be calculated by dividing the revised classified malicious executables by the correct and false predictions of malware files.

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4}$$

Accuracy: The general correctness of the machine learning model through combining the correct classified files of both malicious and benign files and then dividing them by the total number of all files.

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{5}$$

F1 Score: Provides comprehensive evaluation for the classification model even if the dataset is imbalanced, while the Accuracy can be inappropriate in case of an imbalanced dataset. The value will increase if precision and recall values increase which means that the best value for an F1 Score is 1.

$$\mathrm{F1\ Score} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{6}$$

4.2 Proposed System Assessment

In Table 4, the performance metrics of each tested machine learning algorithm are included to select the best algorithm providing higher accuracy and F1 Score in the training dataset as in Shaukat et al. [35]. The results showed that DT has the highest accuracy with an accuracy of 99.99% and an F1 Score of 1, while the second was RF with an accuracy of 99.94% and an F1 Score of 0.9995. The mentioned results have been plotted in Fig. 4 to check the ROC curve showing the relation between TPR and FPR.

Table 4: Performance metrics for each algorithm at the training phase

	Metrics			
Model	Accuracy	Precision	Recall	F1 Score
LR	0.87	0.86	0.93	0.9
LDA	0.87	0.86	0.93	0.9
RF	1	1	1	1
DT	1	1	1	1
SVM	0.61	0.61	1	0.76
NB	0.61	0.61	1	0.76
KNN	0.95	0.95	0.97	0.96
XGBoost	0.96	0.95	0.98	0.97
LightGBM	0.97	0.96	0.98	0.97
Voting	0.97	0.96	0.98	0.97

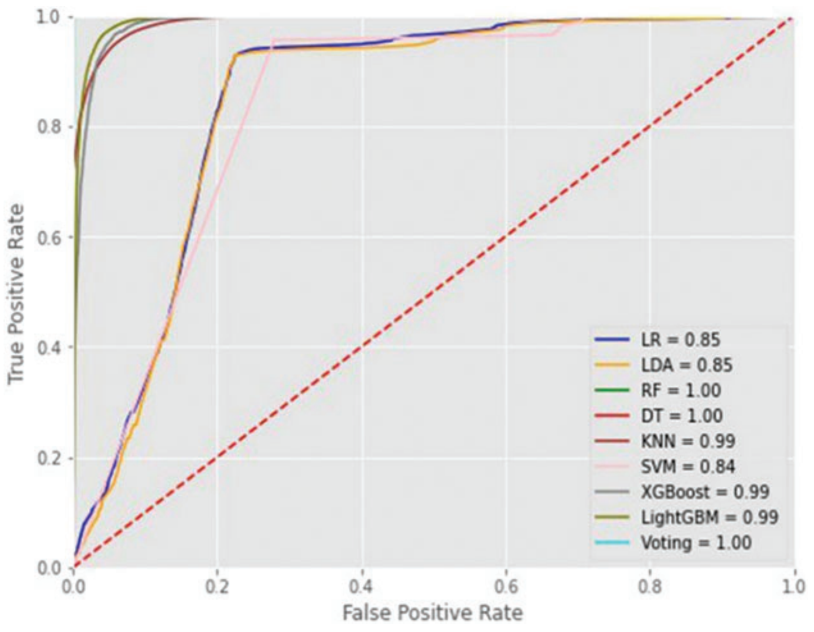

Figure 4: ROC curve at the training phase

For the testing dataset, Table 5 shows the results of each machine learning algorithm used to test the efficiency of executable classification. The test outcome showed that RF has the best accuracy and F1 Score of 98.42% and 0.987, respectively. The second-best algorithm for the testing dataset was DT with an accuracy of 97.49% and an F1 Score of 0.9793. The results were also plotted in Fig. 5 to show the ROC curve and how the classification algorithm can have a great impact on system accuracy and FPR.

Table 5: Performance metrics for each algorithm at the testing phase

	Metrics			
Model	Accuracy	Precision	Recall	F1 Score
LR	0.87	0.86	0.93	0.9
LDA	0.87	0.86	0.93	0.90
RF	0.98	0.98	0.99	0.99
DT	0.97	0.98	0.97	0.98
SVM	0.61	0.61	1	0.76
NB	0.61	0.61	1	0.76
KNN	0.95	0.95	0.96	0.96
XGBoost	0.96	0.95	0.98	0.97
LightGBM	0.96	0.95	0.98	0.97
Voting	0.96	0.96	0.98	0.97

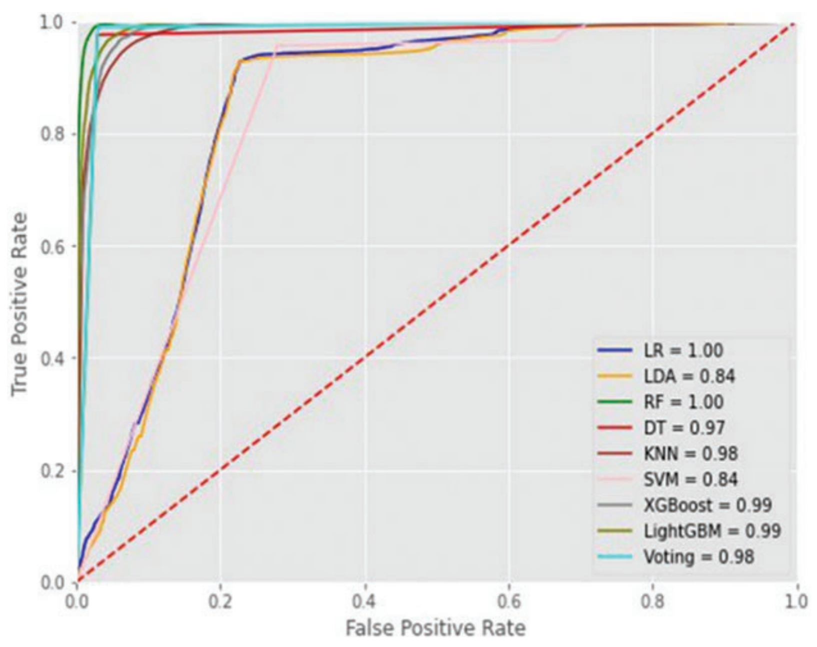

Figure 5: ROC curve at the testing phase

The results showed that the best algorithm was RF which offered the highest accuracy and lowest FPR at the testing phase. These values confirm that the larger dataset plays a vital role in system accuracy in addition to the importance of selecting the most effective features to increase the efficiency of the machine learning algorithm.

After selecting RF, the confusion matrix was plotted, which includes TP, FP, FN, and TN for all instances used in the dataset. Fig. 6 shows the values of the training dataset which is 70% of the total number of datasets, while Fig. 7 clarifies the model's performance using the remaining 30% of the testing dataset. These values clarify the performance of the proposed model and prove the efficiency of detecting malicious executables. For example, the false positives in the training dataset were 96 and became 850 when applied to the testing dataset. The second important indicator is the False Negative which was 19 at the training phase and increased to 544 at the testing phase.

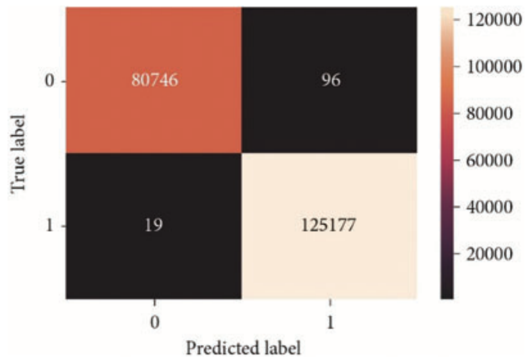

Figure 6: Confusion matrix for training dataset

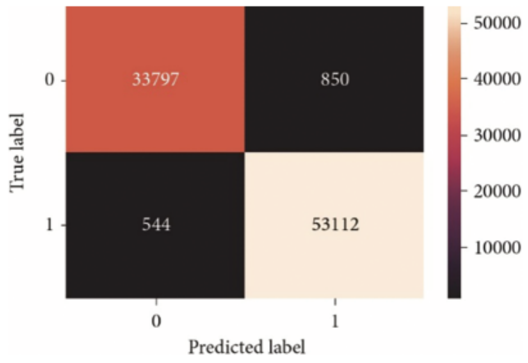

Figure 7: Confusion matrix for testing dataset

The proposed solution using the Random Forest algorithm was also evaluated by statistical tests such as ANOVA, T-Test, and Wilcoxon rank-sum. These tests are done by comparing Random Forest with other algorithms separately and the comparison result is p-value which is the likelihood of finding a mean difference. If the p-value is less than 0.05, so it means that Random Forest has better performance compared to the other algorithm. As shown in Table 6, Random Forest had a significant difference between all algorithms except for NB in the T-Test and Voting CLS in ANOVA and Wilcoxon rank-sum.

Table 6: Random forests statistical tests

	Metrics		
Model	T-Test	ANOVA	Wilcoxon rank-sum
LR	0.005	0.001	0.021
LDA	0.005	0.001	0.021
DT	0.048	0.009	0.021
SVM	0.075	0.039	0.248
NB	0.021	0.004	0.021
KNN	0.005	0.002	0.021
XGBoost	0.038	0.033	0.083
LightGBM	0.019	0.025	0.043
Voting	0.001	0.663	0.386

4.3 Accuracy & Performance Comparison

RMED's proposed system was evaluated by multiple techniques to ensure the improvement of malware detection accuracy. The new dataset was used in the proposed system of Nur Syuhada Selamat et al. [18] with their 28 selected features, then the results were compared with the 15 features selected in this paper. The results showed an enhancement to FPR from 2.26 to 1.58 using the RF algorithm and both systems had the same accuracy. In addition, the CLAMP model which is considered state-of-the-art by Kumar et al. [20] was reproduced using the new dataset generated in this paper. After comparing their 53 features to the 15 features selected, there was a clear improvement in accuracy from 97.37% to 98.42% while the FPR was improved in RMED from 3.98 to 1.58.

In Varma et al. [24], researchers used a deep learning model to detect malicious executables in Windows. The results of the model showed a test accuracy of 97.42% when trained by Anderson et al. [36]. In this study, the same approach was used to compare performances using the gathered dataset, this showed the accuracy on the training dataset to be 97.17%, and the testing dataset decreased to 96.78%. This means that RMED succeeded in improving the accuracy by 2.77% in the training phase and 1.64% in the testing phase. Moreover, there was a prominent difference in the time taken to classify executables with deep learning *vs.* machine learning as well as the impact on Windows performance.

The study has also tested the proposed system in Atluri et al. [37] which used multiple machine learning algorithms to improve the accuracy level. As per their testing, it reached 93.69% system accuracy using the Voting Ensemble Classifier (VEC) which includes five tree-based methods; Bagging Decision Tree Classifier (BDT), Extra Trees Classifier (etc), Random Forest Classifier (RFC), AdaBoost Classifier (ABC), and Gradient Boosting Classifier (GBC).

After building the same machine learning model, the model was tested on the RMED dataset using their 54 features, and it was discovered that the system reached an accuracy equal to 97.22%. This result showed that RMED accuracy was still the best approach with a difference equal to 1.2% when only using 15 features with the Random Forest algorithm.

In terms of system practicality, Windows 10 software has been developed to check the PE files on computers to predict their impact using a stored pickle file including the machine learning model. Then, the software was used with each machine learning model to compare their performance, time, and accuracy on unknown malware samples.

The test was done on 510 Zero-Day Threats which were not included in the dataset to qualify as unknown malware samples. The test results showed that Nur Syuhada Selamat et al. [18] detected 413 samples, Kumar et al. [20] detected 427 samples and Atluri et al. [37] detected 501 samples. Meanwhile, the proposed system RMED succeeded in discovering 509 out of 510 samples.

The test also proved that gathering and analyzing a high number of headers has an impact on time and performance to find an applicable solution for small computing devices as mentioned by NIST standards for lightweight cryptography [38]. When the software collected 15 features only in the case of RMED, it had a faster classification turnover than the other systems collecting 28 or 54 features to predict the impact of each PE file from their headers. The time taken to check executables is a key factor in any cybersecurity solution to prevent the execution of malware in real time. This achievement enabled this study to run the scan continuously on a computer to discover any process hijacking or infected PE files without affecting the performance of Windows devices.

Therefore, it is recommended to reduce the number of features used in the machine learning classifiers and limit the number of algorithms that are highly utilizing the endpoints' resources such as the proposed system in Atluri et al. [37]. This approach can also improve the system's accuracy and avoid overfitting models.

The scan time of each file was also part of the approach evaluation. After testing the model on 469 samples, RMED has shown that it can process each file in 91 milliseconds while Kumar et al. [20] using the CLAMP dataset took 95 milliseconds to classify each sample as benign or malware. The time taken for scanning a file by Manavi et al. [23] was 107 milliseconds. These results prove that RMED can be used to detect Zero-Day threats quickly to prevent any privilege escalation, data exfiltration, or lateral movements.

In addition, it was found that the size of datasets increases the time spent scanning files, therefore, RMED cannot be accurately compared with other approaches using a limited number of datasets such as in Belaoued et al. [22] which reached 77 milliseconds when using 214 benign files and 338 malware samples.

On the other hand, different models were tested on unknown benign samples containing 492 samples to check the false positive rate for each model. RMED has shown a 3% false positive rate, while Nur Syuhada Selamat et al. [18] had 8% and Manavi et al. [23] reached 7% as false detected alerts from all samples.

As shown in Table 7, all the approaches were tested with the same dataset generated in this paper to ensure that all had the same amount of data and types of malware samples. The results showed that RMED had the highest level of accuracy with 98.42% when it was tested by 30% of the unknown malware sample dataset.

Table 7: Performance comparison

Reference number	Learning algorithm	Features selected	Dataset	Accuracy	FPR
[16]	Random forest	28	RMED dataset	0.97	3.65
[18]	Random forest	53	RMED dataset	0.974	2.18
[22]	Deep neural network	8	RMED dataset	0.978	2.57
[34]	Voting ensemble classifier	54	RMED dataset	0.972	2.32
RMED	Random forest	15	RMED dataset	0.984	1.58

The second highest accuracy was found while using the model in Kumar et al. [20] with an accuracy equal to 97.37% while the lowest accuracy detected was for the DNN approach in Varma et al. [24]. It was also clear that the number of used features is not only affecting the scan time, but it is also affecting the system's accuracy. Adding to the accuracy improvement, RMED has succeeded in having the lowest false positive rate with 1.58 compared to 3.65 in Nur Syuhada Selamat et al. [18] and 2.18 in Kumar et al. [20] which both used Random Forest.

5 Discussions

This paper has succeeded in improving the detection accuracy and false positive rate to discover Zero-Day Threats using machine learning techniques. In addition, it has succeeded in generating a novel reliable dataset which is the main challenge for researchers to train and test their machine-learning models. It has also provided a cost-effective solution to be used in a live environment as it selects 15 features only to prevent high computational resources from processing all PE headers.

The proposed solution was compared with 12 other approaches and proved its efficiency and performance. However, the proposed solution was not tested against adversarial attacks which can mislead the machine learning model and increase the false positive rate. The authors also found that the novel dataset still requires improvement by adding new malware samples to empower the model using new behavior of sophisticated attacks.

In this sense, future research studies could discuss the techniques for strengthening Machine Learning models against adversarial attacks. Future work could also merge PE headers and Windows APIs to improve detection accuracy and decrease the false positive rate.

6 Conclusion

Recent cybersecurity measures have proven their inefficiency against advanced cyber-attacks which threaten the continuous development of digital transformation strategies for companies and countries as mentioned in Shaukat et al. [39]. Consequently, there is a great need for innovative techniques to detect Zero-Day Threats using the power of Artificial Intelligence.

This paper has focused on detecting infected PE files which are the highest source of malicious files as ranked by VirusTotal and Statista. The objective of the proposed solution is to detect infected PE files at the host level due to the ability of malware files to be concealed in the network traffic using the wide encryption protocols applied by current websites, emails, and applications worldwide.

This paper succeeded in building a novel dataset from the PE headers to test any machine learning model efficiently and provide a reliable solution to be implemented for a live environment. The dataset size reached 116,031 benign files and 179,071 malware samples collected by the authors from different sources. This dataset was used to train and test multiple machine-learning approaches after the selection of the highly effective 13 PE headers and two entropy values calculated from the selected PE headers.

This novel dataset was tested with 10 different machine learning models and the best results were from Random Forest. After comparing the proposed model RMED with other approaches, this study has proved the accuracy improvement by comparing the results with other papers in addition to performing a practical test with a developed software on Windows 10. The results showed that RMED had the highest accuracy with 98.42% and the lowest false positive rate of 1.58. The test also showed clear progress in scan time which has reached 91 milliseconds to scan each file. In conclusion, this paper has proven the ability to detect Zero-Day Threats in PE files using machine learning algorithms. However, the proposed work can still be improved by collecting Windows APIs as proposed in Kok et al. [40] to have more visibility into the PE files' interactions with the Operating System, In addition, the system can be enhanced using deep learning as in Shaukat et al. [41]. Therefore, this topic is essential to support the global market endeavors which have invested $150 billion in 2021 as per the McKinsey survey published [42] to secure organizations against advanced cyber-attacks.

Acknowledgement: None.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Khaled Soliman; data collection: Khaled Soliman; analysis and interpretation of results: Ayman Bahaa-Eldin, Mohamed Sobh; draft manuscript preparation: Khaled Soliman, Mohamed Sobh, Ayman Bahaa-Eldin. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available within the article or its supplementary materials.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

Supplementary Materials: The supplementary material is available online at https://doi.org/10.32604/cmc.2024.048883.

References

[1] K. Shaukat, T. M. Alam, I. A. Hameed, W. A. Khan, N. Abbas and S. Luo, "A review on security challenges in internet of things (IoT)," in *2021 26th Int. Conf. Automat. Comput. (ICAC)*, Portsmouth, UK, 2021, pp. 1–6. doi: 10.23919/ICAC50006.2021.9594183.

[2] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, "A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review," *Sens.*, vol. 23, no. 8, pp. 4117, Apr. 2023. doi: 10.3390/s23084117.

[3] A. C. Canto, A. Sarker, J. Kaur, M. M. Kermani, and R. Azarderakhsh, "Error detection schemes assessed on FPGA for multipliers in lattice-based key encapsulation mechanisms in post-quantum cryptography," *IEEE Trans. Emerg. Topics in Comput.*, vol. 11, no. 3, pp. 791–797, Jul. 2023. doi: 10.1109/TETC.2022.3217006.

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecur.*, vol. 2, no. 1, pp. 384, Jul. 2019. doi: 10.1186/s42400-019-0038-7.

[5] K. Shaukat *et al.*, "A review of time-series anomaly detection techniques: A step to future perspectives," *Adv. Intell. Syst. Comput.*, vol. 1363, pp. 865–877, 2021. doi: 10.1007/978-3-030-73100-7_60.

[6] "Top submitted file type," VirusTotal. Accessed: Dec. 11, 2022. [Online]. Available: https://www.virustotal.com/gui/stats

[7] "Top malicious file types worldwide 2022," Statista, Accessed: Feb. 06, 2024. [Online]. Available: https://www.statista.com/statistics/1238996/top-malware-by-file-type

[8] Karl-Bridge-Microsoft, "PE Format-Win32 apps," 2017. Accessed: Feb. 06, 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

[9] K. Soliman, M. A. Sobh, and A. M. Bahaa-Eldin, "Survey of machine learning HIDS techniques," in *2021 16th Int. Conf. Comput. Eng. Syst. (ICCES)*, Cairo, Egypt, Dec. 2021, pp. 1–5.

[10] S. Kamran *et al.*, "The impact of artificial intelligence and robotics on the future employment opportunities," *Trends Comput. Sci. Inf. Technol.*, vol. 5, pp. 050–054, Sep. 2020. doi: 10.17352/tcsit.000022.

[11] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: A survey," *J. Cryptogr. Eng.*, vol. 10, no. 2, pp. 135–162, Apr. 2019. doi: 10.1007/s13389-019-00212-8.

[12] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020. doi: 10.1109/ACCESS.2020.3041951.

[13] N. Hubballi and H. Dogra, "Detecting packed executable file: Supervised or anomaly detection method?," in *2016 11th Int. Conf. Availab., Reliab. Secur. (ARES)*, Salzburg, Austria, 2016, pp. 638–643. doi: 10.1109/ARES.2016.18.

[14] A. Nappa, M. Z. Rafique, and J. Caballero, "The MALICIA dataset: Identification and analysis of drive-by download operations," *Int. J. Inf. Secur.*, vol. 14, no. 1, pp. 15–33, Jun. 2014. doi: 10.1007/s10207-014-0248-7.

[15] J. Bai, J. Wang, and G. Zou, "A malware detection scheme based on mining format information," *Sci. World J.*, vol. 2014, no. 1, pp. 1–11, 2014. doi: 10.1155/2014/260905.

[16] X. Ugarte-Pedrero, I. Santos, I. García-Ferreira, S. Huerta, B. Sanz and P. G. Bringas, "On the adoption of anomaly detection for packed executable filtering," *Comput. Secur.*, vol. 43, pp. 126–144, Jun. 2014. doi: 10.1016/j.cose.2014.03.012.

[17] G. Yan, N. Brown, and D. Kong, "Exploring discriminatory features for automated malware classification," *Lect. Notes Comput. Sci.*, vol. 7967, pp. 41–61, Jul. 2013.

[18] N. Syuhada Selamat and F. H. Mohd Ali, "Comparison of malware detection techniques using machine learning algorithm," *Indones. J. Elec. Eng. Comput. Sci.*, vol. 16, no. 1, pp. 435, Oct. 2019. doi: 10.11591/ijeecs.v16.i1.pp435-440.
[19] S. Ranveer and S. Hiray, "Comparative analysis of feature extraction methods of malware detection," *Int. J. Comput. Appl.*, vol. 120, no. 5, pp. 1–7, Jun. 2015. doi: 10.5120/21220-3960.
[20] A. Kumar, K. S. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *J. King Saud Uni.-Comput. Inf. Sci.*, vol. 31, no. 2, pp. 252–265, Apr. 2019. doi: 10.1016/j.jksuci.2017.01.003.
[21] Z. Markel and M. Bilzor, "Building a machine learning classifier for malware detection," in *2014 Sec. Workshop Anti-Malw. Test. Res. (WATeR)*, Canterbury, UK, 2014, pp. 1–4. doi: 10.1109/WATeR.2014.7015757.
[22] M. Belaoued and S. Mazouzi, "A real-time PE-malware detection system based on CHI-square test and PE-file features," *IFIP Adv. Inf. Commun. Tech.*, vol. 456, pp. 416–425, 2015. doi: 10.1007/978-3-319-19578-0.
[23] F. Manavi and A. Hamzeh, "A novel approach for ransomware detection based on PE header using graph embedding," *J. Comput. Virol. Hack. Tech.*, vol. 18, no. 4, pp. 285–296, Jan. 2022. doi: 10.1007/s11416-021-00414-x.
[24] R. K. Varma, P. Raju, S. Raju, and A. Kalidindi, "Feature selection and performance improvement of malware detection system using cuckoo search optimization and rough sets," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 5, pp. 708–714, Jan. 2020. doi: 10.14569/ijacsa.2020.0110587.
[25] S. J. Kattamuri, R. K. V. Penmatsa, S. Chakravarty, and V. S. P. Madabathula, "Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence," *Elec.*, vol. 12, no. 2, pp. 342, Jan. 2023. doi: 10.3390/electronics12020342.
[26] U. Divakarla, K. H. K. Reddy, and K. Chandrasekaran, "A novel approach towards windows malware detection system using deep neural networks," *Procedia Comput. Sci.*, vol. 215, no. 17, pp. 148–157, 2022. doi: 10.1016/j.procs.2022.12.017.
[27] T. Rezaei, F. Manavi, and A. Hamzeh, "A PE header-based method for malware detection using clustering and deep embedding techniques," *J. Inf. Secur. Appl.*, vol. 60, no. 2, pp. 102876, Aug. 2021. doi: 10.1016/j.jisa.2021.102876.
[28] B. David, E. Filiol, and K. Gallienne, "Structural analysis of binary executable headers for malware detection optimization," *J. Comput. Virol. Hack. Tech.*, vol. 13, no. 2, pp. 87–93, Apr. 2016. doi: 10.1007/s11416-016-0274-2.
[29] Erocarrera, "GitHub-erocarrera/pefile: Pefile is a Python module to read and work with PE (Portable Executable) files, GitHub," 2015. Accessed: Feb. 06, 2024. [Online]. Available: https://github.com/erocarrera/pefile
[30] "VirusShare malware collection AAA directory listing," archive.org, 2018. Accessed: Feb. 06, 2024. [Online]. Available: https://archive.org/download/virusshare_malware_collection_aaa
[31] YTISF, "GitHub–ytisf/theZoo: A repository of LIVE malwares, GitHub," 2014. Accessed: Feb. 06, 2024. [Online]. Available: https://github.com/ytisf/theZoo
[32] F. Zatloukal and J. Znoj, "Malware detection based on multiple PE headers identification and optimization for specific types of files," *J. Adv. Eng. Comput.*, vol. 1, no. 2, pp. 153, Nov. 2017. doi: 10.25073/jaec.201712.64.
[33] K. Shaukat, S. Luo, and V. Varadharajan, "A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks," *Eng. Appl. Artif. Intell.*, vol. 116, no. 2, pp. 105461, Nov. 2022. doi: 10.1016/j.engappai.2022.105461.
[34] K. Shaukat, S. Luo, S. Chen, and D. Liu, "Cyber threat detection using machine learning techniques: A performance evaluation perspective," in *2020 Int. Conf. Cyber Warfare Security (ICCWS)*, Islamabad, Pakistan, 2020, pp. 1–6. doi: 10.1109/ICCWS48432.2020.9292388.
[35] K. Shaukat *et al.*, "Performance comparison and current challenges of using machine learning techniques in cybersecurity," *Energies*, vol. 13, no. 10, pp. 2509, May 2020. doi: 10.3390/en13102509.
[36] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," 2018. Accessed: Feb. 06, 2023. [Online]. Available: https://arxiv.org/abs/1804.04637

[37] V. Atluri, "Malware classification of portable executables using tree-based ensemble machine learning," in *2019 SoutheastCon*, Huntsville, AL, USA, 2019, pp. 1–6.

[38] M. S. Turan *et al.*, *Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process*. NIST Technical Series Publications, 2023. Accessed: Feb. 06, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8454.pdf

[39] K. Shaukat, A. Rubab, I. Shehzadi, and R. Iqbal, "A socio-technological analysis of cybercrime and cyber security in Pakistan," *Transylvanian*, pp. 84, Jan. 2017.

[40] S. Kok, A. Abdullah, N. Jhanjhi, and M. Supramaniam, "Prevention of Crypto-ransomware using a pre-encryption detection algorithm," *Comput.*, vol. 8, no. 4, pp. 79, Nov. 2019. doi: 10.3390/computers8040079.

[41] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Eng. Appl. Artif. Intell.*, vol. 122, no. 4, pp. 106030, Jun. 2023. doi: 10.1016/j.engappai.2023.106030.

[42] B. Aiyer, J. Caso, P. Russell, and M. Sorel, "New survey reveals $2 trillion market opportunity for cybersecurity technology and service providers," McKinsey & Company, Oct. 27, 2022. Accessed: Feb. 06, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/new-survey-reveals-2-trillion-dollar-market-opportunity-for-cybersecurity-technology-and-service-providers