A NOVEL MALWARE DETECTION APPROACH USING PERFORMANCE IMPORTANCE WEIGHTED RANDOM FOREST (PERI-WRF) LEARNING MODEL

Amit Parmar

Ph.D. Research Scholar, Gujarat Technological University, Ahmedabad, India
er.amitparmar@gmail.com

Dr Keyur Brahmbhatt

Associate Professor, Information Technology Department,
Birla Vishwakarma Mahavidyalaya, Vallabh Vidyanagar, India
keyur.brahmbhatt@bvmengineering.ac.in

Abstract

Malware detection has gained huge attention in recent times. This is mainly because of the increase in new malware variants which pose a significant threat to information security. The conventional malware detection systems are not capable of detecting new generation malwares due to the constant changes in the network behavior. An efficient malware detection approach must be able to handle the dynamic changes in the malware behavior with a very minimum processing time to identify malicious attacks at the initial stage. This paper presents a novel Performance Importance Weighted Random Forest (PERI-WRF) for detecting different types of malwares in network systems. The proposed PERI-WRF incorporates a novel data reduction technique which is capable of reducing the size of training data to maximize the classification accuracy. A clustering algorithm consisting of GWO and K-means++ algorithm is implemented to group the malicious data samples collected from input. To validate the effectiveness of the detection framework, the system was tested using various evaluation metrics. Results show that the proposed malware detection model with novel data reduction techniques achieves superior classification accuracy, and the proposed approach is appropriate for detecting real-time malwares with superior accuracy and low MAE score.

***Keywords*: Malware Detection, Performance Importance Weighted Random Forest (PERI-WRF), Principal Component Analysis (PCA), K-means++ clustering, Data Reduction, Grey Wolf Optimization (GWO).**

1. Introduction

Malware is intrusive software which can cause significant harm to the computer systems (Vinod et al., 2009) [1]. It is one of the biggest security threats with new variants appearing on a daily basis. Malwares is designed to exploit or harm programmable devices such as computer networks, smart phones etc. To identify the possible malwares in a system, the device files are scanned thoroughly. This process also involves the analysis of the origin, functionality, and types of malware. Malware detection has become one of the significant tasks in software industry and has attracted various researchers in recent decades (Aafer et al., 2013) [2] (Das et al., 2015) [3] (Ozsoy et al., 2015) [4] (Suciu et al., 2019) [5]. For detecting any malicious or suspicious activity in the system, it is essential to have a robust scanning/detection model. These scanning techniques depend on various conventional methods such as static and dynamic analysis, static analysis, statistical and content analysis. However, these techniques fail to protect the system against the attackers since it is easy for the attackers to recreate different variants of the malware which can easily escape from these detection techniques (Komatwar&Kokare, 2021) [6]. Traditional signature-based techniques are not quite effective in detecting malwares since it autonomously creates obfuscation with completely differently appearing signatures that makes textual and binary data difficult to understand. As an alternative, behavior-based techniques have gained significant attention in recent times (Fukushima et al., 2010) [7] (Chang et al., 2016) [8] (Xiao et al., 2019) [9]. These techniques use learning approaches for detecting malicious behavior of the system. Behavior-based approach analyses the behavior of the system using monitoring tools such as feature extraction, data mining or graph mining tools to classify the files as malware or benign (Ming et al., 2017) [10]. Even if the codes of the program are changed, the behavior will remain constant and hence new variants of malware can be easily detected

using this technique. However, some of the behavior-based techniques do not work effectively with new variants of malware because of its complex features. New generation malwares can introduce potential harmful attacks such as targeted and persistent attacks which keep changing, and also different types of malwares are used while launching the attacks (Aslan & Samet, 2020) [11]. It requires advanced and sophisticated techniques to detect new generation malwares. Automated techniques such as machine learning (ML) and deep learning (DL) techniques are considered as potential tools for malware detection because of their superior generalization capacity (El Merabet&Hajraoui, 2019) [12]. Different ML algorithms such as support vector machine (SVM) (Rashidi et al., 2017) [13], Decision Trees (DT) (Zulkifli et al., 2018) [14], Logistic Regression (LR) (Kumar et al., 2017) [15], are used widely in different malware detection approaches. However, conventional ML algorithms depend mainly on feature extraction and feature learning mechanisms which require expert domain knowledge (Raff et al., 2018) [16] (Rhode et al., 2018) [17]. Besides, once the hacker is able to access and understand the features, it becomes easy for the attacker to conceal their identity and deceive the detection models(Anderson et al., 2017) [18]. In this context, this paper identifies Random Forest (RF) algorithm as one of the potential algorithms for malware detection because of its classification and prediction capability. They have shown superior detection ability in various literary works (Alam&Vuong, 2013) [19] (Mills et al., 2019) [20] (Roseline et al., 2020) [21].

Though RF algorithms have shown outstanding performance in malware detections, its performance can be improved by incorporating base models to enhance the prediction accuracy. However, considering equal weights i.e., a simple average for all base models is not appropriate since it affects the final predictions. This is mainly because of the random selection of input features and random sampling. Randomization does not ensure that all the constructed trees in a random forest have the same decision-making capabilities (Li et al., 2010) [22]. Hence, it is essential to have an appropriate weighting procedure to add weight to the trees based on their learning ability. In this context, research proposes a Performance Importance Weighted Random Forest (PERI-WRF) based learning model for malware detection. A novel data reduction technique is implemented to maximize the classification accuracy of the proposed model.

The contributions of this paper are summarized as follows:

- This paper presents a novel PERI-WRF learning model for detecting malware from the given input dataset.
- A novel Grey Wolf Optimization (GWO) with K-means++ clustering algorithm is implemented to cluster and label the data samples into malicious files.
- This work employs a t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm as a data reduction process to reduce the data dimensionality and number of features for classification.
- This research aims to show that reduction in the number of features will have a significant impact on the accuracy.
- A comparative analysis is presented which compares the time complexity for different training and testing data split ratios.

The rest of the paper is organized as follows: Section 2 reviews some of the related works while section 3 discusses the construction of RF classifiers. Section 4 discusses the proposed research methodology for malware detection which includes dataset details, PERI-WRF mod
el and data reduction mechanism. Section 5 discusses the experimental results and performance evaluation and the conclusion is outlined in section 6.

2. Literature Review

In the past decades, several works have discussed the implementation of ML and DL algorithms for developing efficient malware detection systems (Liu et al., 2017) [23] (Rathoreet al., 2018) [24] (Li et al., 2018) [25] (Pan et al., 2020) [26]. This section presents the analysis of various existing works. The survey focuses mainly on two categories namely feature extraction and reduction, and classification. As discussed previously, in static analysis, the features are generated without running the code and in dynamic analysis, features are accessed while executing the code. (Schmidt et al., 2009) [27] performed static analysis on the data samples for extracting the function calls in android smartphones. (Dhaya&Poongodi, 2015) [28] detects the vulnerabilities in the software system using static analysis. The focus was given to the analysis of the static code using the ML algorithm known as N-gram analysis. This model detected the novel malicious behavior present in android smart phones with good accuracy. (Shijo& Salim, 2015) [29] integrated both static and dynamic analysis for classifying novel executable files. This work employed ML algorithms where both benign and malware data samples are used as training data. The ML algorithm analyses the binary code and the dynamic behavior of the system and based on that the feature vectors are selected. Results show that the integrated approach improved the classification accuracy. While the static and dynamic model achieved an accuracy of 97.1% when applied as a single model, the proposed integrated approach achieved 98.7% accuracy. However, the static and dynamic analysis is outperformed by the deep learning based learning models. One such approach is presented in (Choi et al., 2017) [30], where a novel method of detecting the malwares using DL algorithm is presented. The images are generated from both benign and malware files. As

a next step, a trained model is used for identifying malware from the system. This approach eliminates the need of both static and dynamic analysis. An effective ML-based classifier was proposed in (Narudin et al., 2016) [31] evaluated the implementation of ML algorithms for classifying in mobile applications. The proposed approach combined anomaly detection with ML algorithms to improve the detection accuracy. Experimental evaluation showed that both RF and Bayes algorithm achieved an highest accuracy of 99.97% compared to multi-layer perceptron which achieved an accuracy of 93.03%. However, it was inferred that the k-nearest neighbor classifier detected the new generation malware in android applications with highest true-positive rate compared to other techniques. (Rana et al., 2018) [32] evaluated different tree-based ML classifiers for malware detection using an effective feature selection approach. Experimental evaluation was conducted on 11,120 applications from the DREBIN dataset wherein 5560 data samples contained malwares and remaining samples were benign. As observed from the results, RF shows significantly higher performance in terms of accuracy of 97.24% compared to SVM which achieved an accuracy of 94% and hence provides a potential base for developing potential tools for detecting potential threats and attacks. The efficacy of ML algorithms, specifically random forest, is discussed in (Agrawal & Trivedi, 2021) [33]. The study analyzed different ML classifiers for malware detection in android applications. The study discussed the strengths and limitations of the classifiers. The paper concludes that the RF algorithm is one of the efficient algorithms and achieves higher accuracy in comparison with SVM and Naive Bayes algorithms.

3. Random Forest (RF) Classifier

RF is a supervised ML algorithm which performs classification and regression simultaneously. The RF algorithm was initially developed by Leo Breiman (Ellis et al., 2014) [34] which create the forest with a certain number of trees. More the number of trees in the algorithm, more robust is the potential of the algorithm i.e., higher the number of trees in the algorithm leads to higher classification and prediction accuracy. In RF, the trees are is constructed using a bootstrap sample of instance and the feature set of the input data in each tree is considered as a random subset of the global features. Correspondingly, RF algorithms incorporate the advantages of two ML techniques namely bagging and random feature selection for decision making. Each decision tree in a random forest has a low bias since it is unpruned and the correlation between each tree is very low due to bagging and random feature selection. Hence, RF provides an ensemble with low bias and variance.

The classification and prediction is obtained from a set of results through various evaluation strategies such as averaging (simple and weighted average) and voting. When compared to single decision tree classifiers, RF shows a higher performance (Li et al., 2015) [35] (Ham et al., 2013) [36] and has following advantages:

- It is suitable for applications which have more features than actual observations.
- It achieves desired performance despite complexities such as increase in the number of predictor variables and noise.
- It overcomes the problem of overfitting.
- No continuous fine-tuning of parameters is required to achieve better performance.

In this work, the RF algorithm initially considers 'L' bootstrap samples from the dataset and for each sample. A classification tree is constructed and the features are selected randomly. The classifiers in the training model and the output of the classifier are denoted as $\{c_1, c_2, \ldots, c_T\}$ and $c_i(x)$ respectively. During classification, each classifier will predict an output from the sample labeled dataset $\{o_1, o_2, \ldots, o_N\}$. In this research, the labeled dataset is malware and benign. Further the outputs of the individual classifier are combined to make a final prediction which is based on majority voting. The predicted outputs are represented as an N-dimensional vector to make it suitable for defining the prediction constraint as shown in equation 1. (Zhu et al., 2018) [37].

$$C(x) = f(x) = \begin{cases} o_k & if \ \sum_{i=1}^{T} c_i^k(x) > 0.5 \sum_{l=1}^{N} \sum_{i=1}^{T} c_i^l(x) \\ reject, & Otherwise \end{cases} \quad (1)$$

Where C(x) is the final prediction classification result, c_i^k (x) is the N-dimensional vector of individual classifiers, ok is the class label.

However, the classification accuracy of the RF classifiers reduces with the increase in the number of training data. A novel data reduction technique is proposed in this research which reduces the size of training data and extracts only relevant and small subset of features from the dataset to reduce the classification error. In addition, the conventional RF classifier assumes equal weight for all the base models which is not reasonable. This paper proposes a performance importance weighted RF algorithm.

4. Proposed research methodology

In ML- based malware detection frameworks, the input data is represented using a larger number of features and hence the number of training samples required to train the RF model also increases. However, all the features are

not relevant for prediction and the presence of irrelevant features increases the computational complexity of the ML algorithms and has a negative impact on the classification and prediction accuracy. Conversely, extraction of relevant and small subset of features can improve the performance of these algorithms (Moonsamy et al., 2011) [38]. Based on this concept, this paper employs a data reduction technique and a performance weighted RF algorithm. The preliminary aim of this research is to show that reduction in the number of features and the size of training data is possible without compromising the accuracy. The steps involved in the implementation of the proposed approach are discussed in below subsections;

4.1. *Dataset Details*

For achieving desired performance, it is essential to select a representative dataset to train the ML algorithm. The dataset should accurately be adaptive to the real-time applications. In this work, the data for experimental analysis was obtained from the Classification of Malware with PE headers (ClaMP) dataset which contains both malware and benign data. The data set comprises 54 features extracted from binary executable using PEFILE features. The dataset contains 5210 rows and 70 columns and the feature set is identified from 96724 benign files and 41323 malicious files. For each of the entries pertaining to a malicious file, an associated attribute specifying the risk level is recorded.

During data preprocessing, the redundant data is filtered out from the data frame to make it appropriate for classification. Next, the dataset is split into label 0 and label 1, where label 0 contains the benign files and label 1 contains all malicious files. These labels are used to construct the proposed PERI-WRF classification model. Further, the malicious files are converted into three-level malicious files (malicious 0, malicious 1, and malicious 2) to split the malicious samples from the dataset. This is done using a clustering algorithm.

4.2. *Clustering Technique*

This work proposes a novel K-means++ cluster with Grey Wolf Optimization (GWO) based clustering algorithm to cluster the malicious dataset from the dataset samples. In general, clustering techniques work based on centroid and GWO optimization is used to find the best centroid value. Based on the obtained centroid, K-means++ clustering algorithm will group/cluster the data. GWO is a meta-heuristic optimization algorithm which is inspired by gray wolves. The GWO algorithm follows the hunting process and leadership hierarchy of the gray wolves. It employs four types of wolves' behaviors such as hunting, searching for prey, encircling prey, and attacking prey for finding the optimized value. Along with this, four types of gray wolves namely alpha, beta, delta, and omega are used for identifying the leadership hierarchy (Mirjalili et al., 2014) [39]. The alpha, beta, delta, and omega estimates the location of the prey and other wolves will update their locations randomly around the prey. Whereas, K-means++ algorithm initializes the centroid obtained from the GWO and clusters all the data points.

In this work, it randomly selects the first centroid from the available data points and for each data point the distance from the nearest centroid (or previously chosen centroid) is calculated using the K-means++ algorithm. The next centroid is selected from the data points whose distance is maximum from the nearest centroid. The process is repeated until the 'k' number of centroids is sampled. Here, the number of clusters is fixed as 3 and 10 numbers of wolves are considered for the clustering process. An overall 1000 iterations was performed for grey wolves and the centroid values are updated after every iteration. For each iteration, 3 best centroid values are obtained and these values are used by the K-means++ algorithm to group the malicious data from the dataset.

The pseudo code for the GWO-K-means++ algorithm is given below:

Step 1: Initialization
Initialize the number of wolves, clusters, iteration:
Initialize the number_wolfs=10 ,number_cluster=3 ,n_iteration =1000
Wolfs =[] //empty wolf list

Step 2: Finding the wolfs

```
Class GWOCluster:
        wolf = None
        def  _init_wolfs()
        for  (i in range(number_wolfs)):
                Wolf(); // called wolf function, here calculating the
                best_score, gbest_centroids, gbest_score  //Defined the wolfs based clustering
                if (wolf.best_score<gbest_score):
                        wolfs.append(wolf)
                end if
        end for
```

```
        End function
```

Step 3:(Finding the gbest_centroids,gbest_score)

```
Class GWOCluster:
        def run():
history = []
                for(i in range(n_iteration)):
                for(wolf in range(number_wolfs)):
                        wolf.update(gbest_centroids,data) // wolf will updated based on centroids
                        If(wolf.best_score<gbest_score)
                                gbest_centroids=wolf.centroids
                                gbest_score=wolf.best_score
                        End if
                        history.append(gbest_score)
                End for
                End for
         Return history
```

Step :4 (Finding Clustering based on gbest_centroids)

```
Class Kmeans ():
        defint():
                n_cluster =3
                Centroid =gbest_centroids // from GWO class
        End function

        def fit(data): // passing the data
                distance = self._calc_distance(data) // calling the calc_distance
                        function for calculating the distance
                cluster = self._assign_cluster(distance)
                new_centroid = self._update_centroid(data, cluster) //finding the new
                                                                    //centroids based on data ,cluster
                diff = numpy.abs(self.centroid - new_centroid).mean() //find the mean
between gbestcentroid ,newcentroid.
                self.centroid = new_centroid
        End function

                def predict(data):
                distance = self._calc_distance(data)
        cluster = self._assign_cluster(distance)
                        return cluster // final cluster data will return here .
        End function
```

4.3. *Data Reduction Technique*

Once the data is clustered into malicious data, the data is further reduced using a t-SNE data reduction technique. As inferred from existing works, processing a large-scale training dataset increases the training time. One of the best solutions to address this issue is to minimize the dimension of the training set while preserving the characteristics of the data. It can be achieved by selecting a representative subset of the actual massive data set. This approach is referred to as the Data Reduction Technique (DRT). The proposed t-SNE will reduce the issue of data dimensionality in order to explore high-dimensional (HD) data.

The proposed t-SNE is an unsupervised and nonlinear algorithm which is employed to investigate and visualize HD data. In other words, t-SNE algorithm will provide a clear vision about the arrangement of data in a HDspace. It overcomes the limitations of other dimensionality reduction techniques such as Principal Component Analysis (PCA). PCA operates using a linear process and cannot effectively compute complicated polynomial relationships between features. Also, PCA places all dissimilar data samples in a low-dimensional (LD) space. But, to represent the HD data in LD space, it is essential to represent all the similar data points placed closely, which is not possible in linear data reduction techniques such as PCA. The proposed t-SNE algorithm finds the data patterns by determining their resemblance with the features. The resemblance is determined using a conditional probability wherein point xi will choose point xjas its neighbor as shown in the equation 2. (Pouyet et al., 2018) [40].

$$p_{ij} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_j||^2 / 2\sigma_i^2)} \quad (2)$$

Where P_{ij} is the conditional probability, σ_i is the variance of the Gaussian which is centered on data point x_i. For representing the LD data points represented as y_i and y_j from the HD x_i and x_j, the same condition (shown in equation 2) can be used which is denoted by q_{ij}. The probabilities of LD data points are calculated as shown in equation 3.

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_k - y_i||^2)^{-1}} \quad (3)$$

Further, the difference between the conditional probabilities or similarities of the two data points is reduced in both high-dimensional and low-dimensional space for representing the data points in lower-dimensional space.

The process flow of the proposed data reduction technique using t-SNE is shown in figure 4.1

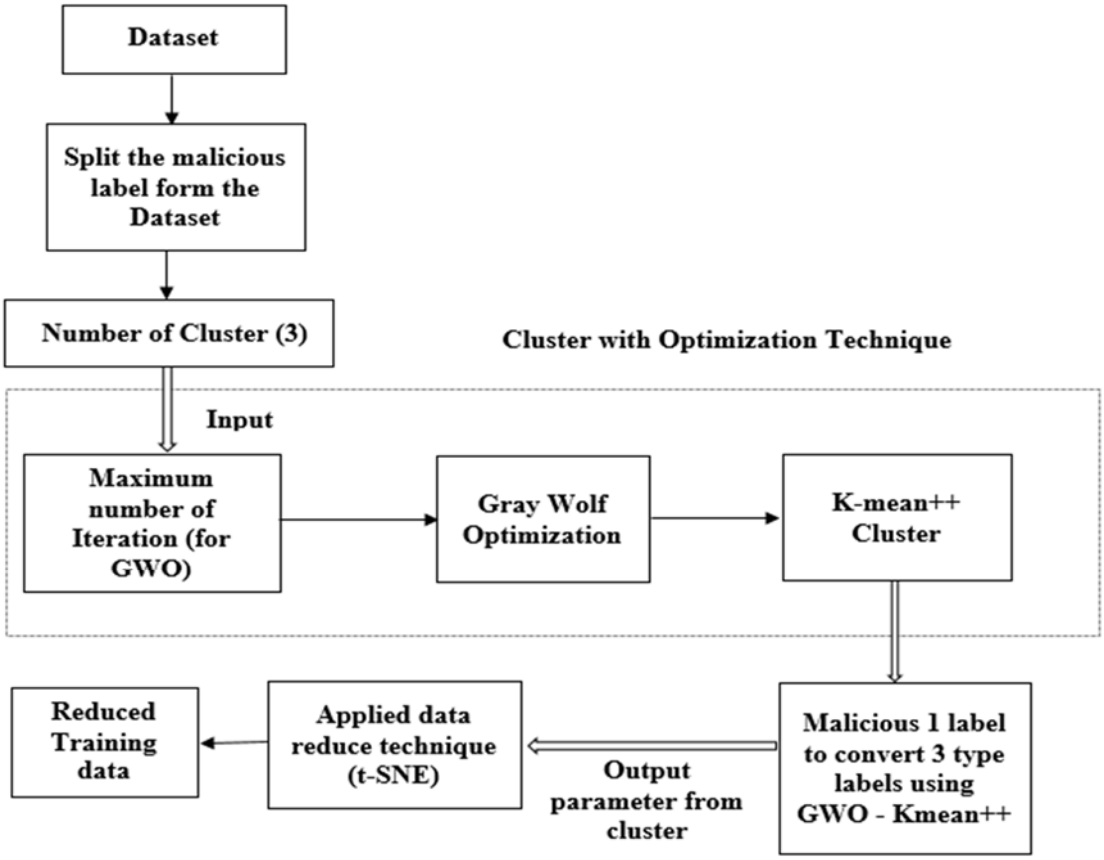

Fig. 4.1 Data reduction technique using t-SNE algorithm

4.4. *Construction of PERI-WRF model*

In an ensemble model, many base learners are combined together for producing better results. Whenever the predictive learners are aggregated together, weighting each model and tuning the parameters are very critical. Generally, Random Forest treats tree level results equally. It can be improvised by weighting trees in such a way that better performing trees are weighted more. In this research, the PERI-WRF models are designed using out-of-bag (OOB) accuracy. Here, the out-of-bag performance of the classifier is determined in terms of accuracy and the normalized accuracy of each classifier is used as its weight to form an ensemble of classifiers. The process of the proposed PERI-WRF model is illustrated in figure 4.2.

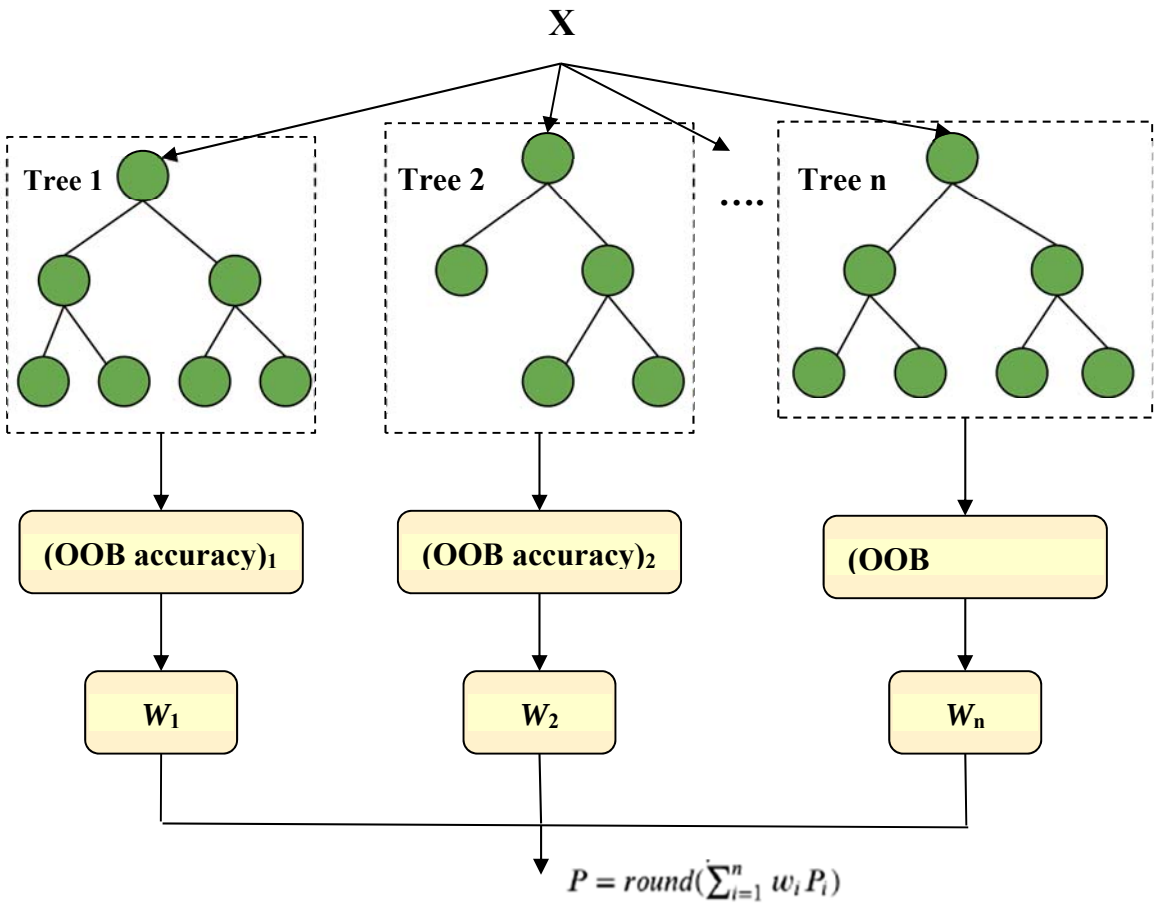

Fig. 4.2 Illustration of the proposed PERI-WRF model

Considering training data T for training the model and the dataset has M number of features are available. The X_i is defined as a feature set and y_i is defined as labels. Now, the training data T is expressed as

$$T = \{(X_1,y_1), (X_2,y_2), \ldots (X_n, y_n)\} \quad (4)$$

The size of the training dataset is assumed to be the same as the original dataset. The random re-sampling of the training data T results in the formation of {T1, T2, ….,Ts} datasets. This dataset is called the bootstrap dataset. Due to "with-replacement", every dataset Ti can have duplicate data records. As discussed previously, the performance of the PERI-WRF model is analysed using OOB accuracy which uses a bagging approach. Bagging is the process of taking bootstraps & aggregating the models learned on each bootstrap. In this process, each decision tree (base models) generates Ntree (j □ {1, ….,Ntree) and are sampled randomly. The newly generated training data is called an in-bag sample. Each in-bag sample is assumed to have an approximate value of 63.2% unique observations and the remaining samples are considered as out-of-bag samples (Gajowniczek et al., 2020) [41].

The actual implementation of the RF model combines the outputs of each individual classifier and obtains a final probability score Yi^{RF} which is given in equation 5.

$$Y_i^{RF} = \sum_{j=1}^{Ntree} I[Y_{ij} > 0.5] \quad (5)$$

Where I represent the indicator function, You is the probability for ith observation for jth tree. Equation 5 employs a majority voting mechanism with a specific threshold limit of 0.5. In this work, the trees in the random forest for classification are built using a general RF algorithm. However, performance-based weights are used for aggregating the outputs of the trees. Specifically, weighting probability is considered for each tree in the forest in such a way that trees which perform better are weighted more heavily than others. The weighting probability is defined as;

$$Y_i^{RF} = \frac{1}{Ntree} \sum_{j=1}^{Ntree} Y_{ij} * w_j \quad (6)$$

Where, w_j represents the weight of the j^{th} tree.

The accuracy of the weighted RF model is calculated using OOB and bootstrap based on the majority vote. Further, the performance is measured based on the Area Under the Curve (AUC) score. Then the weight on each tree will be calculated and the output of the trees with higher weights is used for classifying the malicious files based on the OOB observations.

5. Results and Discussion

The data collected from the (ClaMP) dataset which contains both malware and benign data. The data is split into training data (70%) and testing data (30%). Different performance metrics are used to determine the performance. The classification ability of the proposed PERI-WRF model is compared with traditional RF algorithms. Furthermore, the proposed data reduction technique is compared with PCA to validate the performance. The results of the experimental analysis are discussed in below sections. This section should come before the References. Funding information may also be included here.

5.1. *Performance Evaluation of PERI-WRF model*

The performance of the PERI-WRF model is analyzed using the Evaluation metrics as discussed in below equations. The metrics are defined using the elements of the confusion matrix namely: True positives (TP), True negatives (TN), False positives (FP), False negatives (FN). The expressions for evaluating the performance metrics are discussed as follows:

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN) \quad (7)$$

Recall is calculated as the ratio of the malwares detected and which are accurately classified and is given as:

$$\text{Recall} = TP/(TP + FN) \quad (8)$$

Precision defines the accuracy of positive predictions

$$\text{Precision} = TP/(TP + FP) \quad (9)$$

Correspondingly, F1 score is defined as:

$$\text{F1 score} = (2 * \text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall}) \quad (10)$$

In general, the value of F1 score lies between 0 and 1 which defines worst and best values respectively. F1 score can also provide an estimated measurement of accuracy.

The confusion matrix of the PERI-WRF model and the corresponding results are given in figure 5.1 and table 1 respectively.

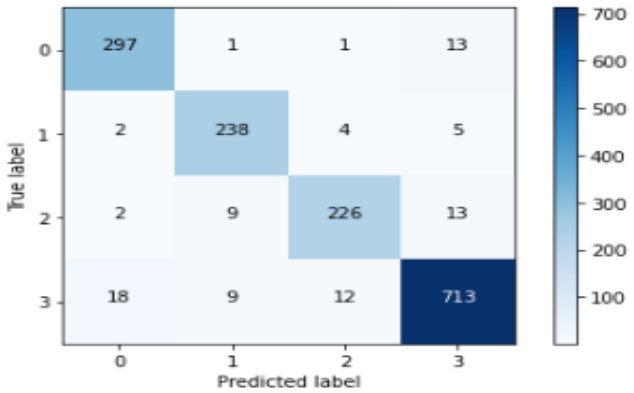

Fig. 5.1 Confusion matrix of the proposed PERI-WRF model

Performance Metrics	Kmeans++	PCA	Random Forest	PERI-WRF
Accuracy	0.89	0.82	0.88	0.95
Recall (Sensitivity)	0.85	0.78	0.80	0.93
Precision (Specificity)	0.88	0.78	0.80	0.94
F1-score	0.87	0.78	0.80	0.93
Mean Absolute Error (MAE)	0.15	0.25	0.22	0.08
AUC	0.97	0.90	0.79	0.99

Table 1. Evaluation of PERI-WRF model

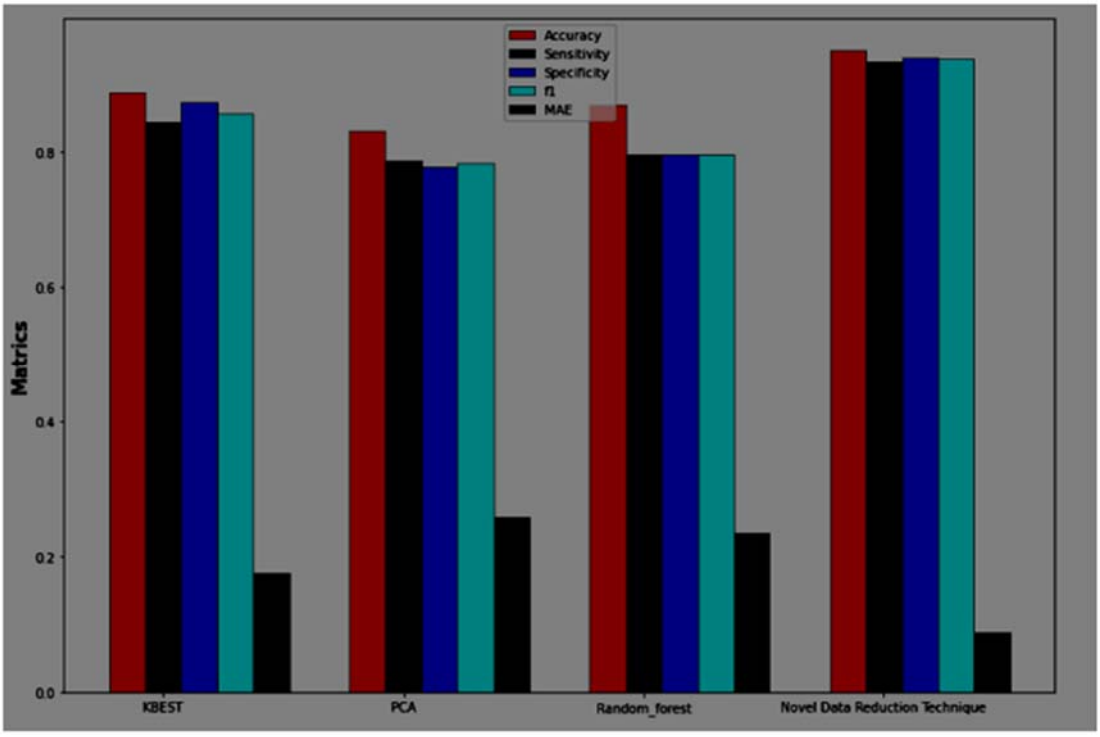

Fig. 5.2 Comparative analysis of different data reduction techniques

Figure 5.2 describes the performance measurement of different data reduction techniques in terms of accuracy, sensitivity, specificity, MAE, and F1 score. It can be inferred that the proposed PERI-WRF model outperforms all the other models by achieving an outstanding accuracy of 95% and with a lower MAE score of 0.08. Correspondingly, the ROC of the PERI-WRF model and other models is shown in figure 5.3.

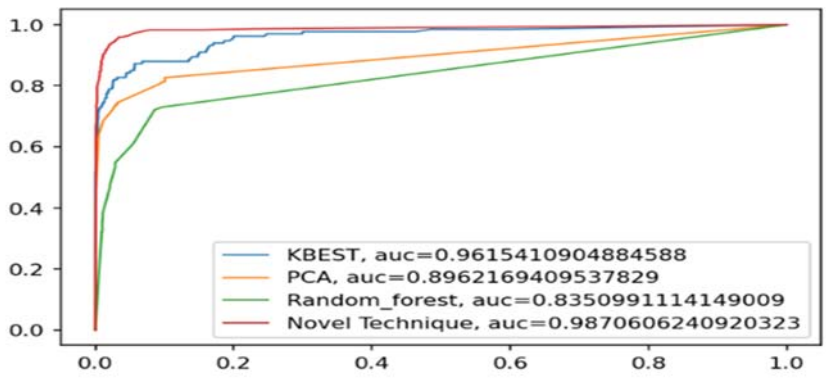

Fig. 5.3 ROC of the PERI-WRF and other models

The ROC curve is used to determine the classification process of the proposed model. AUC provides an overall measure of performance at different classification levels. Higher the AUC rate, better is the performance of the model in distinguishing between TP and FP rates. The proposed approach achieves an AUC of 97.70% which is significantly higher compared to the AUC values of other models such as K-means, PCA and conventional random forest.

5.2. *Comparative analysis*

To validate the performance of the proposed approach, the results of the PERI-WRF model were compared with conventional random forest algorithms using the same dataset without data reduction technique.

The confusion matrix of the RF model and results are given in figure 5.4 and table 2 respectively.

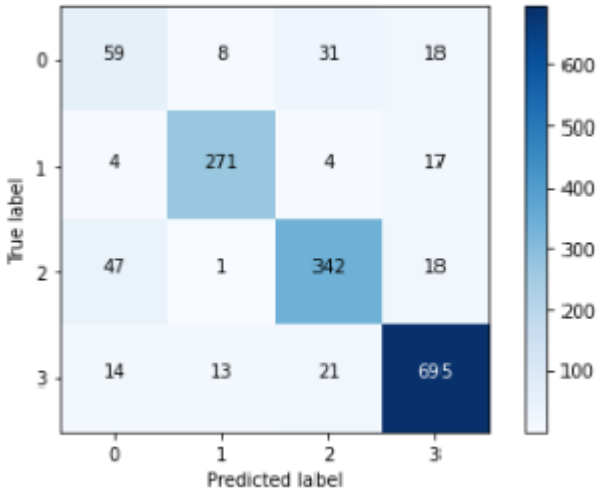

Fig. 5.4 Confusion matrix of the proposed RF model

Performance Metrics	Kmeans++	PCA	Random Forest
Accuracy	0.87	0.82	0.88
Recall (Sensitivity)	0.83	0.77	0.81
Precision (Specificity)	0.84	0.77	0.80
F1-score	0.84	0.77	0.80
Mean Absolute Error (MAE)	0.20	0.28	0.21
AUC	0.91	0.90	0.85

Table 2. Evaluation of RF model with K-means and PCA

Compared to the results of the PERI-WRF model with data reduction technique (as shown in table 2), the performance of random forest algorithm without data reduction is quite moderate. The RF algorithm achieves an accuracy of 88% with a MAE of 0.21. Similar to the proposed approach, the performance of the RF classifier was also determined by plotting an ROC curve as shown in figure 5.6.

The ROC curve was plotted for RF classifiers and other models to determine the performance of these techniques without data reduction. As inferred from figures 5.3 and 5.6, the data reduction technique has a significant effect on the performance on the AUC. Here, the RF algorithm with normal feature selection mechanism shows deteriorated performance with an overall AUC of 83.94% in comparison to the proposed PERI-WRF model with improved accuracy. While K-means++ achieves second best performance with an AUC of 90.45%, PCA achieves lower performance with an AUC of 88.64%.

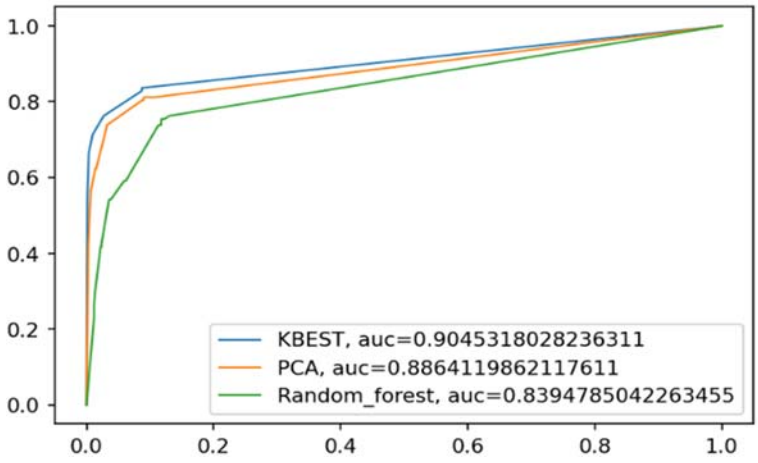

Fig. 5.6 ROC of the RF classifier and other models

In addition to the previously discussed performance, the time complexity of different approaches along with the proposed approach. The time complexities are analyzed with respect to different training and testing dataset ratios such as 60:40, 70:30, and 80:20. The results of the simulation analysis are tabulated in below given figures.

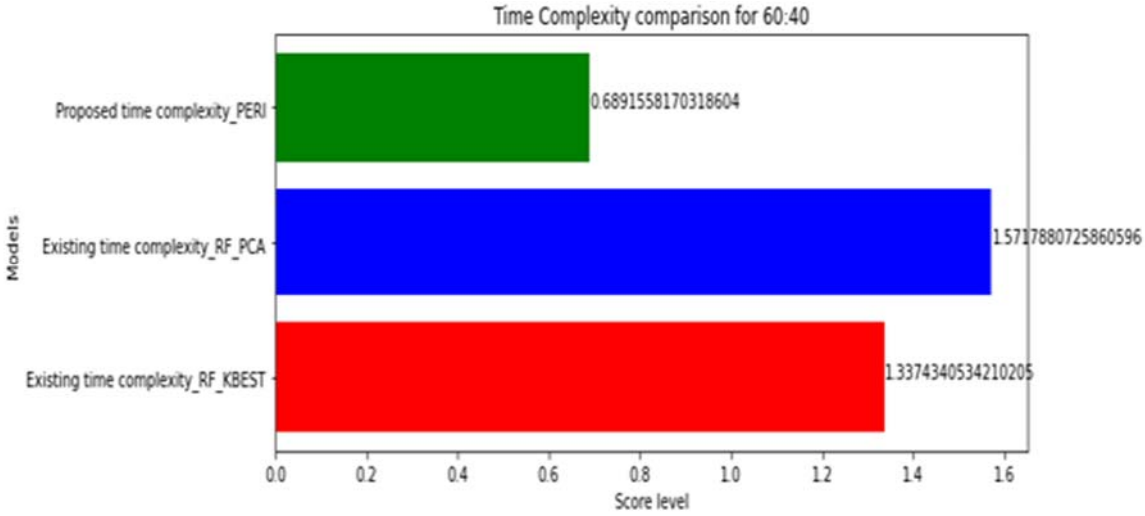

Fig. 5.7 Time complexity for a data split ratio of 60:40

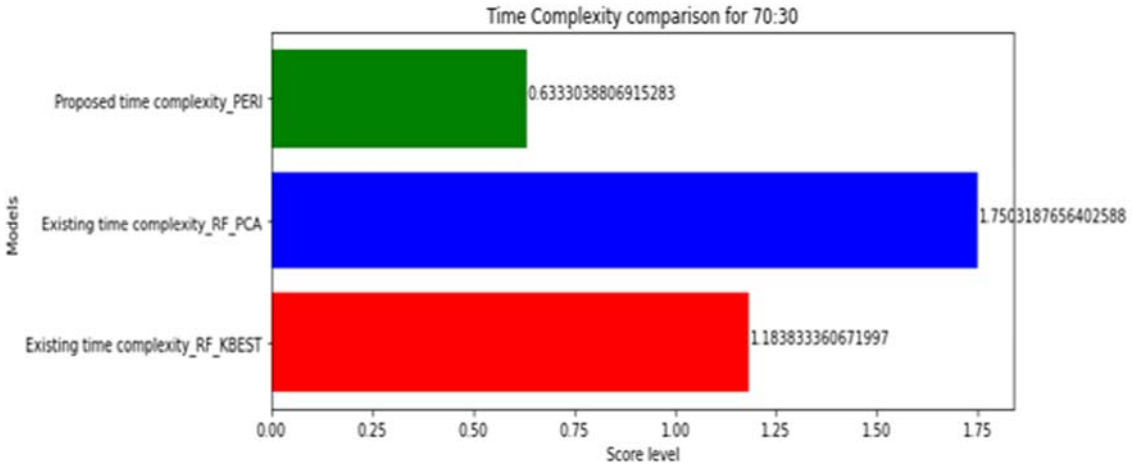

Fig. 5.8 Time complexity for a data split ratio of 70:30

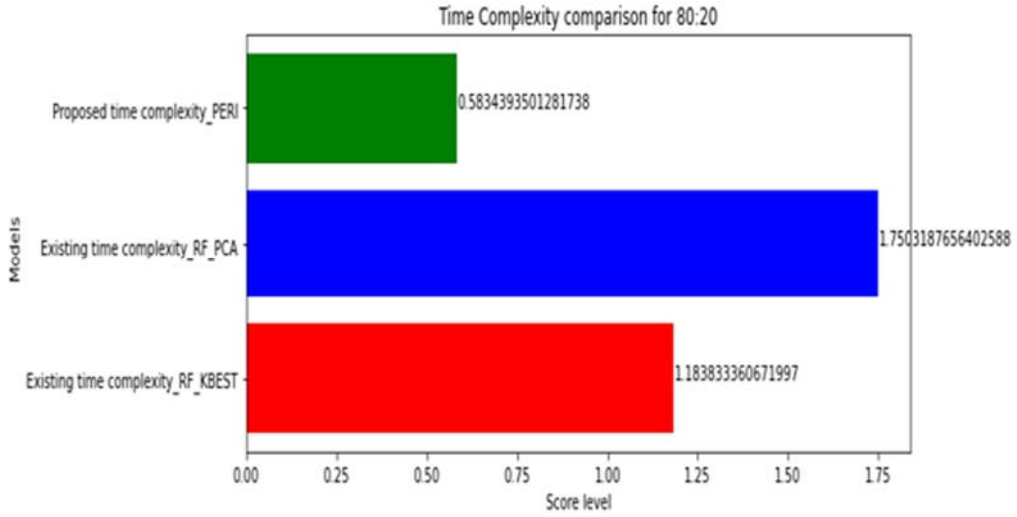

Fig. 5.9 Time complexity for a data split ratio of 80:20

The figures show the time complexity which determines the time taken for execution. The time required by the RF model to execute a particular task is defined as the time complexity. The superiority of the model is determined by its time complexity wherein the execution speed of the model increases with the reduction in the time complexity. As inferred from figures 5.7, 5.8, and 5.9, the time complexity for the proposed model is very less for a data split ratio of 80:20 compared to other approaches for different data splits.

6. Conclusion

This research discusses the implementation of a novel PERI-WRF framework with novel data reduction techniques for malware detection. The proposed research consists of a clustering technique to label and cluster malicious data from the benign data. As inferred from existing studies, the increase in the number of features and size of training data affects the classification accuracy. To overcome this problem, this research implemented t-SNE as a data reduction technique. The t-SNE algorithm reduced the size of the training data to enhance the accuracy of the PERI-WRF classifier. Experimental analysis was mainly conducted to analyze the influence of data reduction techniques on the classification accuracy. In addition, the same dataset was used to test the performance of a conventional random forest classifier without data reduction technique. The primary objective of this research was to show that the data reduction technique and selection of relevant features will have a positive impact on the accuracy and performance of the classifiers. It can be inferred from experimental results that the proposed approach achieved a superior accuracy, a high recall and a high precision score. The maximum accuracy achieved by the proposed PERI-WRF model was 95% with the mean score being 0.08. In future work, the study intends to deploy the proposed PERI-WRF model in a real environment and validate the performance of the proposed model by launching different variants of malwares.

References

[1] Vinod, P., Jaipur, R., Laxmi, V., & Gaur, M. (2009, March). Survey on malware detection methods. In Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09) (pp. 74-79).
[2] Aafer, Y., Du, W., & Yin, H. (2013, September). Droidapiminer: Mining api-level features for robust malware detection in android. In International conference on security and privacy in communication systems (pp. 86-103). Springer, Cham.
[3] Das, S., Liu, Y., Zhang, W., &Chandramohan, M. (2015). Semantics-based online malware detection: Towards efficient real-time protection against malware. IEEE transactions on information forensics and security, 11(2), 289-302.
[4] Ozsoy, M., Donovick, C., Gorelik, I., Abu-Ghazaleh, N., &Ponomarev, D. (2015, February). Malware-aware processors: A framework for efficient online malware detection. In 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA) (pp. 651-661). IEEE.
[5] Suciu, O., Coull, S. E., & Johns, J. (2019, May). Exploring adversarial examples in malware detection. In 2019 IEEE Security and Privacy Workshops (SPW) (pp. 8-14). IEEE.
[6] Komatwar, R., &Kokare, M. (2021). A survey on malware detection and classification. Journal of Applied Security Research, 16(3), 390-420.
[7] Fukushima, Y., Sakai, A., Hori, Y., & Sakurai, K. (2010, October). A behavior based malware detection scheme for avoiding false positive. In 2010 6th IEEE workshop on secure network protocols (pp. 79-84). IEEE.
[8] Chang, W. L., Sun, H. M., & Wu, W. (2016, August). An android behavior-based malware detection method using machine learning. In 2016 IEEE International conference on signal processing, communications and computing (ICSPCC) (pp. 1-4). IEEE.
[9] Xiao, F., Lin, Z., Sun, Y., & Ma, Y. (2019). Malware detection based on deep learning of behavior graphs. Mathematical Problems in Engineering, 2019.
[10] Ming, J., Xin, Z., Lan, P., Wu, D., Liu, P., & Mao, B. (2017). Impeding behavior-based malware analysis via replacement attacks to malware specifications. Journal of Computer Virology and Hacking Techniques,13(3), 193-207.
[11] Aslan, Ö. A., &Samet, R. (2020). A comprehensive review on malware detection approaches. IEEE Access, 8, 6249-6271.
[12] El Merabet, H., &Hajraoui, A. (2019). A survey of malware detection techniques based on machine learning. Int. J. Adv. Comput. Sci. Appl, 10(1), 366-373.

[13] Rashidi, B., Fung, C., &Bertino, E. (2017, November). Android malicious application detection using support vector machine and active learning. In 2017 13th International Conference on Network and Service Management (CNSM) (pp. 1-9). IEEE.
[14] Zulkifli, A., Hamid, I. R. A., Shah, W. M., & Abdullah, Z. (2018, February). Android malware detection based on network traffic using decision tree algorithm. In International Conference on Soft Computing and Data Mining (pp. 485-494). Springer, Cham.
[15] Kumar, B. J., Naveen, H., Kumar, B. P., Sharma, S. S., & Villegas, J. (2017, March). Logistic regression for polymorphic malware detection using ANOVA F-test. In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) (pp. 1-5). IEEE.
[16] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C. K. (2018, June). Malware detection by eating a whole exe. In Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence.
[17] Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. computers& security, 77, 578-594.
[18] Anderson, H. S., Kharkar, A., Filar, B., & Roth, P. (2017). Evading machine learning malware detection. Black Hat, 1-6.
[19] Alam, M. S., &Vuong, S. T. (2013, August). Random forest classification for detecting android malware. In 2013 IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing (pp. 663-669). IEEE.
[20] Mills, A., Spyridopoulos, T., & Legg, P. (2019, June). Efficient and interpretable real-time malware detection using random-forest. In 2019 International conference on cyber situational awareness, data analytics and assessment (Cyber SA) (pp. 1-8). IEEE.
[21] Roseline, S. A., Geetha, S., Kadry, S., & Nam, Y. (2020). Intelligent vision-based malware detection and classification using deep random forest paradigm. IEEE Access, 8, 206303-206324.
[22] Li, H. B., Wang, W., Ding, H. W., & Dong, J. (2010, November). Trees weighting random forest method for classifying high-dimensional noisy data. In 2010 IEEE 7th international conference on e-business engineering (pp. 160-163). IEEE.
[23] Liu, L., Wang, B. S., Yu, B., &Zhong, Q. X. (2017). Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, 18(9), 1336-1347.
[24] Rathore, H., Agarwal, S., Sahay, S. K., &Sewak, M. (2018, December). Malware detection using machine learning and deep learning. In International Conference on Big Data Analytics (pp. 402-411). Springer, Cham.
[25] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics, 14(7), 3216-3225.
[26] Pan, Z., Sheldon, J., & Mishra, P. (2020, October). Hardware-assisted malware detection using explainable machine learning. In 2020 IEEE 38th International Conference on Computer Design (ICCD) (pp. 663-666). IEEE.
[27] Schmidt, A. D., Bye, R., Schmidt, H. G., Clausen, J., Kiraz, O., Yuksel, K. A., ... &Albayrak, S. (2009, June). Static analysis of executables for collaborative malware detection on android. In 2009 IEEE International Conference on Communications (pp. 1-5). IEEE.
[28] Dhaya, R., &Poongodi, M. (2014, May). Detecting software vulnerabilities in android using static analysis. In 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies (pp. 915-918). IEEE.
[29] Shijo, P. V., & Salim, A. J. P. C. S. (2015). Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46, 804-811.
[30] Choi, S., Jang, S., Kim, Y., & Kim, J. (2017, October). Malware detection using malware image and deep learning. In 2017 International Conference on Information and Communication Technology Convergence (ICTC) (pp. 1193-1195). IEEE.
[31] Narudin, F. A., Feizollah, A., Anuar, N. B., &Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. Soft Computing, 20(1), 343-357.
[32] Rana, M. S., Rahman, S. S. M. M., & Sung, A. H. (2018, September). Evaluation of tree based machine learning classifiers for android malware detection. In International Conference on Computational Collective Intelligence (pp. 377-385). Springer, Cham.
[33] Agrawal, P., & Trivedi, B. (2021). Machine learning classifiers for Android malware detection. In Data Management, Analytics and Innovation (pp. 311-322). Springer, Singapore.
[34] Ellis, K., Kerr, J., Godbole, S., Lanckriet, G., Wing, D., & Marshall, S. (2014). A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers. Physiological measurement, 35(11), 2191.
[35] Li, W., Ge, J., & Dai, G. (2015, November). Detecting malware for android platform: Ansvm-based approach. In 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (pp. 464-469). IEEE.
[36] Ham, H. S., & Choi, M. J. (2013, October). Analysis of android malware detection performance using machine learning classifiers. In 2013 international conference on ICT Convergence (ICTC) (pp. 490-495). IEEE.
[37] Zhu, H. J., Jiang, T. H., Ma, B., You, Z. H., Shi, W. L., & Cheng, L. (2018). HEMD: a highly efficient random forest-based malware detection framework for Android. Neural Computing and Applications, 30(11), 3353-3361.
[38] Moonsamy, V., Tian, R., & Batten, L. (2011, October). Feature reduction to speed up malware classification. In Nordic Conference on Secure IT Systems (pp. 176-188). Springer, Berlin, Heidelberg.
[39] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.
[40] Pouyet, E., Rohani, N., Katsaggelos, A. K., Cossairt, O., & Walton, M. (2018). Innovative data reduction and visualization strategy for hyperspectral imaging datasets using t-SNE approach. Pure and Applied Chemistry, 90(3), 493-506.
[41] Gajowniczek, K., Grzegorczyk, I., Ząbkowski, T., & Bajaj, C. (2020). Weighted random forests to improve arrhythmia classification. Electronics, 9(1), 99.

Authors Profile

Amit B Parmar is currently working as Assistant Professor, Department of Computer Engineering, Government Engineering college, Modasa. He is having more than 7 years of teaching experience. He received his B.E. degree and M.E. degree in Computer Engineering from Gujarat Technological University, Gujarat. He is currently a Research Scholar from Gujarat Technological University. His area of interest is Machine Learning and Computer Security. He has guided projects for B.E. and M.E. students in domain of Machine Learning, Deep Learning and Cyber Security. He is a member of ISTE society.

Dr. Keyur N. Brahmbhatt is working as an Associate Professor and head of Information Technology Department at BVM Engineering College, V.V.Nagar. he is B.E in Information Technology , M.E in Computer Engineering and Ph.D in the area of Image Processing. He has published more than 30 research papers in the area of image processing, image fusion and data mining. He has guided PG students. He is a member of IETE, India.