Malware Analysis Through Random Forest Approach

Ajay Kumar[1,*], Kumar Abhishek[1], Shishir Kumar Shandilya[2] and Muhammad Rukunuddin Ghalib[3]

[1]*Department of Computer Science & Engineering, NIT Patna, Bihar, India*
[2]*Division Head, Cyber Security and Digital Forensics, Vellore Institute of Technology, VIT Bhopal University, India*
[3]*School of Computer Science and Engineering, Vellore Institute of Technology (VIT), Vellore, India*
E-mail: ajayk.phd18.cs@nitp.ac.in; kumar.abhishek@nitp.ac.in; shishir.sam@gmail.com; ruk.ghalib@gmail.com
[*]*Corresponding Author*

Received 30 April 2020; Accepted 11 August 2020;
Publication 06 December 2020

Abstract

This paper gives precise and comprehensive detail along with a proposed system for malware detection using ML and Deep Learning techniques by integrating both behavior-based detection methods and signature-based methods. The primary purpose of this paper is (A) Outline difficulty identified with malware detection. (B) Represent detail and categorized ML technique for malware detection. (C) Investigating the structure of basic strategies in malware discovery. (D) Inspecting the essential deep learning approach for malware detection using a grouping of malware inside the data mining. The point of interest and downside of various malware detection approaches were analyzed based on evaluation strategy and their capability. The proposed model uses random forest for making an end-to-end pipeline for malware detection. During comparative study with five other state of the art models, the proposed model obtained accuracy of 99.7% on the dataset. The experimental results show the proposed model outperformed other five state of the art techniques. This research paper encourages the researcher to think about the best approach for malware detection.

Keywords: Machine intelligence, deep learning, signature-centric discovery, behavioral-based detection.

1 Introduction

The term malware is a program which developed in order to introduce malicious harm in the system, Network, or information for personal beneath [1, 2]. Malware is divided into subgroups depending on their behavior, which include Worms, Spyware, Botnet, Trojans, Rootkits, Backdoor, and Virus. Terminology malware detection indicates the method for determining whether the current program supposed P is malicious or not based on prior knowledge called Knowledge Store-something that previously known. Malware examination is arranged into two sub-classes, to be specific Static analysis(Testing sample is analyzed without its execution concerning with signature of a program) and Dynamic analysis(Test sample is analyzed by its execution concerning with behavior of program). Currently, a Machine learning-based classification approach, used for identifying malicious programs from being one [3, 4]. There several ML algorithms that perform best over others based on no of future included and type of data. Malware can be accessible for rent in the market and are used by cybercriminals to begin massive attacks like a phishing attack, click fraud, and DoS. Contrasting and other existing malware, Polymorphic or changeable malware are more hazardous because, more often than not, it found undetected [5, 6]. Numerous Organization including McAfee, Kaspersky Lab, Microsoft, DELL, and Social site portal, in particular, Facebook, Twitter, Wikileaks, and numerous other government companies and organizations of created nations to be specific FBI, Europol and the UK's National Wrongdoing Office (NCA) are as yet proceeding with their fight against the malware worldwide using controls measure, laws, and other numerous measures.

1.1 Problem Statement

No perfect anti-malware framework is available in the market for advanced malware discovery. Using ML and Deep Learning classification algorithm on the malware dataset, we will successfully detect the malware attack. For best classification accuracy with the help of Windows PE format, we are going

Table 1 Advantage and Disadvantage of Behavior-based malware identification

Advantage	Disadvantage
Zero-Day attack detected	Complex Execution
Polymorphic attack undetected	Execution time rises
Metamorphic attack detected	Storage complexity
Detecting new kind of attacks	–
Easily detect data-flow dependency	–

to choose the most representative subset of the features toward the accurate classification of the malware Indirectly, supervising users about potential Malware attack.

1.2 Detection Approaches

Malware identification proof methodologies are isolated into two key classes that incorporate behavior-based and signature-based procedures.

1.2.1 Behavior-based detection

The primary benefit provides a better understanding of wherewith malware is made and executed. Malware sample is executed in a sandbox environment called Dynamic analysis, where both imitating conditions and virtualization needed, and runtime activity is monitored and recorded.

1.2.2 Signature-centric discovery

One primary benefit comes from their attention to detail since they go behind all possible execution behavior of a specific archive by checking the sign. The anti-malware supplier used the evolutionary algorithm surveillance to search malicious things using the signature. This Malicious thing included in the existing database (Which contains a vast number of numerous signature)once they identified by the anti-malware software.

2 Related Works

In the given section, the existing Malware detection is analyzed by utilizing different assessment factors such as the Classification technique used, Accuracy, False positive(FP), Total Dataset used, Data analysis method preferred, Dataset sources, Site [7].

Table 2 Advantage and Disadvantage of signature-based malware identification

Advantage	Disadvantage
Simple Execution	Metamorphic attack Undetected
Less Complexity	Polymorphic attack undetected
Less Execution Time	Zero-day attack undetected
Fast Detection	Repeated Huge info in dataset
Detail malware info Collection	–

2.1 Analysis of Behavior-based detection

Table 1 shows essential view for each logical research in behavior-based methodologies. The essential benefit is distinguishing all of the suspicious activity according to the Programming interface(API calls) considers that raises the precision of malware detection. The main drawback is its run-time operating cost. Target Environment includes embedded frameworks, Windows-based, then smartphones. Majority investigation studies have utilized the Smartphone environment for representing novel malware detection approach using behavior methodologies. A logical comparison includes classification or clustering method, data analysis technique, in addition to the kind of data set with features used and its accuracy.

2.2 Analysis of Signature-based Detection

Table 2 shows an essential view for each logical research in signature-based methodologies. The essential benefit is it utilizing signature discovery that declines the system operating cost and execution time for malware detection. Target Environment includes embedded frameworks, Windows-dependent then smartphones. Majority investigation studies have utilized the Windows-based environment for speaking to novel Malware discovery strategy using signature methodologies. A logical comparison includes classification or clustering method, data analysis technique, in addition to the kind of data set with features used and its accuracy.

2.3 Discussion

Behavior-based detection on Window Platform: Table 3 Presents factual diagram for behavior-based malware detection strategies. The analysis suggested a different detection approach using ML, such as RandomForest(RF),

Table 3 Malware detection algorithm comparison – for window system

Paper	Algorithm	Top Accuracy with Model	Features	Analysis Method
NB [12]	RIPPER,Naive-Bayes and Multi Naive-Bayes classifiers	0.9969	4	Static
LCB [1]	Leveraging Compression-based Graph Mining	0.993	2	Dynamic
NBJ48 [13]	Naive Bayses, BaysNet, IB1,J48,and regression algorithms for classification	0.993	2	Dynamic
DPI [14]	Deep Packet Inspection (OPI) and IPpacket headers classification, NLP, Signalpipeline, J48, Naivebase	0.98	11	Dynamic
SS [15]	Sub-SCDG	0.973	3	Dynamic
DTRF [**?**]	Decision tree,Random Forests,and Support Vector Machine	0.9719	4	Dynamic
SIA [16]	Security importance assessment for system objects using “No read down” and “nowrite”	0.9392	3	Dynamic
OOA [17]	OOA+CBAA	0.913	455	Hybrid
OC-SVM [18]	OC-SVM,Naive Bayses, Logistic regression, SMO,SVM, Decision tree,Voted perceptrons	0.835	6	Dynamic

ANN, SVM, J48 decision tree, Naive Bayes(NB) [3, 4, 7, 8]. The top 3 model includes NaiveBayes(NB), Leveraging Compression-based Graph Mining(LCB) and J48 has the best accuracy 0.9969, 0.993 and 0.983 for corresponding features 4, 12 and 22 with false-positive rates 0.074, 0.005 etc. The current investigation shows that the highest number of the dataset used 15000, 7507 with malicious sample 15000(M), 6994(M), and 7158(M) and clean sample 251(C), 513(C) [9]. The majority detection strategies have utilized dynamical analysis with 0.778 percent, the hybrid analysis utilized with 77.8 percent, and the remaining static analysis utilized with 11.11 percent usage [10, 11]. The graph represents the comparison between the existing model.

Table 4 Malware detection algorithm comparison – for window system

Algorithm	Top Accuracy with Model	Features	Analysis Method
Leveraging Compression-based Graph Mining [1]	0.993	12	Dynamic
J48(Decision trees),Naive Bayes,SVM,IB, Collaborative,MCDF [7]	0.989	20	
BAYESNET,NAIVE BAYES,SMO,J48,RANDOM TREE [30]	0.978	6	Static
KNN [31]	0.9766	300	Static
Deep-learning-based Android malware detection engine [15]	0.9676	192	Hybrid
Naive Bayes,BaysNet, IB1, J48, SVM, Regression [32]	0.960715	11	Dynamic
SVM Poly Kernel, SVM linear Kernel,k-nearest neighbour [12]	0.956	65	Dynamic
DBN Deep Learning Model [33]	0.9505	–	Hybrid
SVM,Decision Tree [34]	0.85	73	Hybrid
SafeDriod v2.0 [35,36]	0.7713	–	Dynamic

Table 5 Malware detection algorithm comparison – for embedded system

Algorithm	Top Accuracy with Model	Features	Analysis Method
DeepAm learning [37,38]	0.993	12	Dynamic

2.3.1 Behavior-based detection on Android Platform

Table 4 Presents a factual diagram for all signature and behavior-based malware detection strategies. The analysis suggested a different detection approach using ML, such as RandomForest(RF), ANN, SVM, J48 decision tree, Naive Bayes(NB) [3,7,19,20]. The top 3 model includes Leveraging Compression-based Graph Mining, SVM, and RANDOM TREE has the best accuracy 0.993, 0.989, and 0.978 for corresponding features 12, 20, and 6 with false-positive rates 0.005, 0.988 and 39 [16,21,22]. The current investigation shows that that highest number of the dataset used 7507, A summation of 135 permits and 210 API requests and 734 with malicious sample 6994(M),1073 and 231(M) and clean sample 513(C), 904 AND 504(C) [17,23,24]. The majority detection strategies have utilized dynamical analysis with 40 percent, the hybrid analysis utilized with 30 percent and the remaining static analysis utilized with 30 percent usage [25–29].

2.3.2 Behavior-based detection on Embedded Platform

The analysis suggested a different detection approach using ML, such as RandomForest(RF), ANN, SVM, J48 decision tree, Naive Bayes(NB) [39]. The top model includes 'DeepAm learning, and AutoEncoder stacked up amidst multilayer restricted Boltzmann' has the best accuracy 0.978 for corresponding features 5 with false-positive rates 0.05 [40–42]. The current investigation shows that the highest number of the dataset used 20,000 with malicious sample 4500(M) and clean sample 4500(C).

2.3.3 Signature-based detection on window Platform

Table 5 Presents factual diagram for behavior-based malware detection strategies. The analysis suggested a different detection approach using ML, such as RandomForest(RF), ANN, SVM, J48 decision tree, Naive Bayes(NB) [22, 43, 44]. The top 3 model includes Topological feature extraction-Belief Propagation (BP), MLP, and Density-based K-means has the best accuracy 0.9999, 0.986 and 0.9836 for corresponding features 20, 40 with false-positive rates 0.0001, 0.02. The current investigation shows that the highest number of the dataset used 28760,52,185 and with malicious sample 10760(M), 41265(M), and clean sample 16,800 (C), 10920(unknown sample). The majority detection strategies have utilized dynamical analysis with 57.14 percent, the hybrid analysis utilized with 42.86 percent.

2.3.4 Signature-based detection on Android Platform

Table 6 Presents factual diagram for all signature-based malware detection strategies. The analysis suggested a different detection approach using

Table 6 Malware detection algorithm comparison – for windows system

Algorithm	Top Accuracy with Model	Features	Analysis Method
Topological feature extraction-Belief Propagation (BP) [14, 24]	0.9999	20	Dynamic
MLP, SVM, NaiVe [38]	0.986	–	Hybrid
ANN, Density based K-means [36]	0.9836	40	Hybrid
Naive Bayes and support vector machines [27]	0.968	–	Dynamic
ANN, KNN, NaiveBayes, SVM, J-48 [29]	0.9525	4	Hybrid
Bayesian, J-48 [39]	0.95	3	Dynamic
N-grams and improved version of SVM [23]	0.9300	8	Dynamic

Table 7 Malware detection algorithm comparison – for android system

Algorithm	Top Accuracy with Model	Features	Analysis Method
SVM, Decision Tree [34]	0.85	73	Hybrid
SVM [33]	0.987	19	Hybrid
MKLDroid []	0.98	5000	Static
NaiveBayes, Decision Trees [28]	0.973	8	Hybrid
MocDroid, Classifier generic Algorithm [31]	0.9515	140	Static
Droid classifier using SVM [40]	0.9433	–	Dynamic
DroidNative, CF GO-IL [37]	0.94	194	Dynamic
NaiveBayes, Decision Tree [28]	0.9357	8	Hybrid
KNN [32]	0.9590+-0.015	1000	Hybrid

Table 8 Malware detection algorithm comparison – for embedded system

Algorithm	Top Accuracy with Model	Features	Analysis Method
KNN, SVM, and Adaboost [41]	0.8470	12	Dynamic

ML, such as RandomForest(RF), ANN, SVM, J48 decision tree, Naive Bayes(NB). The top 3 model includes Decision Tree, SVM, and MKL-Droid has the best accuracy for corresponding features 73, 19, and 5000 with false-positive rates 0.0645, 0.17. The current investigation shows that the highest number of the dataset used 401, 62561 with malicious sample 197(M), 29877+17684(M), and clean sample 204(C),15000(C). The majority detection strategies have utilized dynamical analysis with 11.11 percent, the hybrid analysis utilized with 33.33 percent, and the remaining static analysis utilized with 55.56 percent usage.

2.3.5 Signature-based detection on Embedded Platfrom

The analysis suggested a different detection approach using ML, such as RandomForest(RF), ANN, SVM, J48 decision tree, Naive Bayes(NB). The top model includes AdaBoost has the best accuracy 0.8470 for corresponding features 12 (see Table 7).

3 Proposed System

Table 8 Represents for Malware Analysis using Behavior Approach.

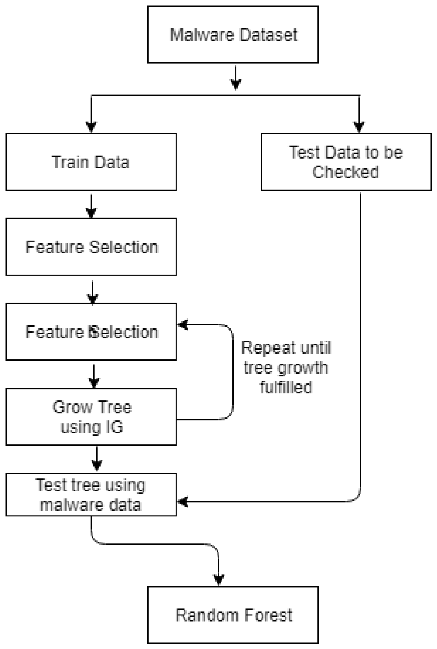


Figure 1 Flow diagram.

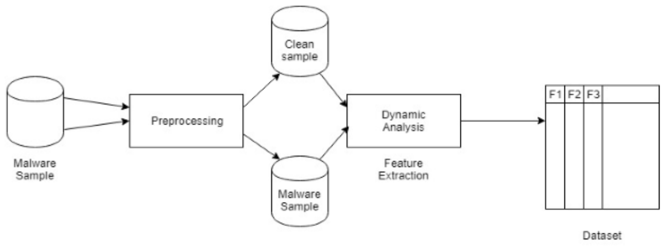


Figure 2 Architecture diagram.

3.1 Architecture

The data generation is all about objects or things that deliver the information assessed by the detection framework. E.g., there is a system in a network that generates traffic, which either produces both clean and malicious data [45–47]. This data generation performs through Cuckoo Sandbox or Honey-pot, which executing malware in a controlled manner to extricate its behavior pattern. In a considerable network, there could be tens or many thousands of information generators.

3.1.1 Data Collection

Collect and combine data into a single stream from various individual information generators. The collector could be a system screen on the edge of a vast system or the ISP. Contingent upon the extent of the system screen, the data collection might be various leveled and be comprised of various levels of the collector at the various edge point. Data collection acts as the first level of filtering information that screens around Cuckoo Sandbox or Honeypot for e.g., DNS.

3.1.2 Preprocessing

The undesirable information will be anything not required for identification and will commonly packet payload is removed before going to feature extraction level [8].

3.1.3 Features Selection

Remove features from data and process it as per ML Algorithm need. Features selected carefully so that to improve the efficiency and performance of the model.

3.1.4 Eliminate unwanted data

Lessen the size of data to a certain extent as well as expel conceivable false positives (FPs) [48]. Most Widely recognized tools such as Whitelists for evacuating known generous data focus and blacklist for evacuating known malicious focuses or signature-based channel to expel known and simple to identify malware.

3.1.5 Detection module

It can act as an anomaly detector that can detect suspicious behavior. In border case, it will be the ML algorithm mainly supervised or unsupervised algorithm. In a supervised ML algorithm used basically for labeling sample or assign it to a specific category basically either clean or malicious. In an unsupervised ML algorithm, clustering utilized to segregate data. To achieve higher accuracy and performance, multiple ML modules can be used in combination. For the classifier, the model should be trained using both clean and malicious samples. For the clustering algorithm, a new point or data is incorporated effectively in existing clustered information.

3.1.6 Knowledge Store

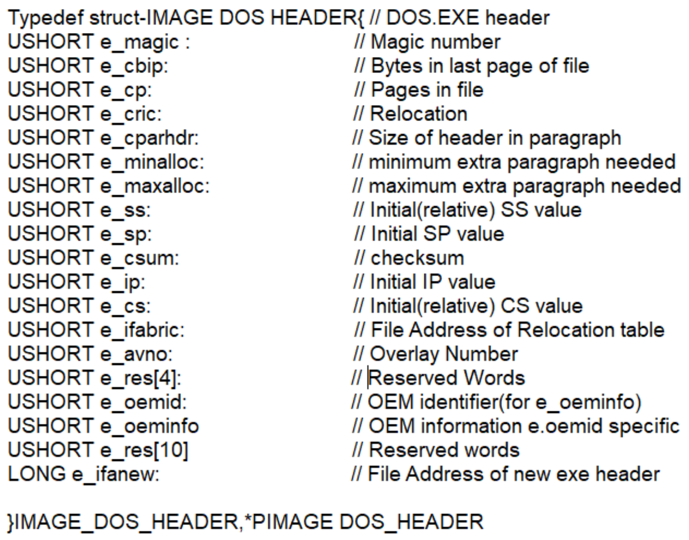
They were generally used by a supervised ML algorithm to classify new data based on existing information. In the anomaly ML identification algorithm, Past behavior would be a knowledge store that utilized for new data classification purposes. In the clustering ML identification algorithm, existing knowledge used for classifying the clean and malicious sample.

3.1.7 Result

Output includes a comparison between various ML models using accuracy, FP, TP, and performance factor. Top accuracy considers with features for the corresponding ML algorithm.

4 PE File Structure

PE File format design for executables, Obj.code, DLLs, FON Font records utilized in 32/64-bit Windows OS (see Figure 3). PE record describes how the loader outlines the information in memory when preparing the start load. The data structure of Windows includes DDL deferences, API import and export table, TLS data, resources dealing data, executables, and powerfully connected libraries, which is important for window OS loader to deal with



```
Typedef struct-IMAGE DOS HEADER{ // DOS.EXE header
USHORT e_magic :                  // Magic number
USHORT e_cbip:                    // Bytes in last page of file
USHORT e_cp:                      // Pages in file
USHORT e_cric:                    // Relocation
USHORT e_cparhdr:                 // Size of header in paragraph
USHORT e_minalloc:                // minimum extra paragraph needed
USHORT e_maxalloc:                // maximum extra paragraph needed
USHORT e_ss:                      // Initial(relative) SS value
USHORT e_sp:                      // Initial SP value
USHORT e_csum:                    // checksum
USHORT e_ip:                      // Initial IP value
USHORT e_cs:                      // Initial(relative) CS value
USHORT e_ifabric:                 // File Address of Relocation table
USHORT e_avno:                    // Overlay Number
USHORT e_res[4]:                  // Reserved Words
USHORT e_oemid:                   // OEM identifier(for e_oeminfo)
USHORT e_oeminfo                  // OEM information e.oemid specific
USHORT e_res[10]                  // Reserved words
LONG e_ifanew:                    // File Address of new exe header

}IMAGE_DOS_HEADER,*PIMAGE DOS_HEADER
```

Figure 3 PEFile structure.

executable code. Executable record characterizes the structure of Windows executables and powerfully connected libraries (DLLs). It characterizes how the loader should map the information in memory when a handle is being stacked. The PE record arrange composed of a definite stream of information. It starts first from a MS-DOS-header, a genuine mode program-stub, and a PE-sign. PE file structure continues with a file and an optional header. After that section header continued with section body. Finishing off the PE structure with are a couple of different areas of random data, including migration data, image table data, line number data, and string table information.

5 Experimental Setup

In this area, we portray the dataset utilized during investigations, features selection, and extraction. At that point, we present an assessment of the proposed model, which gives bits of knowledge picked up from the analyses. At last, we contrast our work, and an ongoing malware detection methodology talked about in related work. Implementation is available at.[1]

5.1 PC and Software Requirement

To validate the proposed idea, an experimental environment is expected with Ubuntu 16.04.5 LTS os running an Intel(R) Core i5 @ 3.2 GigaHertz processor having four GigaBytes of primary memory, 500 GB of secondary memory with window professional, Adobe Reader PDF version 11.1.0 for running Cuckoo Sandbox and python. As dynamic analysis performed on Cuckoo Sandbox using an Ubuntu system with Windows OS as the base system. For sandbox need Intel(R) Core i5 @ 3.2 GHz processor with R for selection of feature and python 2.7 for feature extraction.

5.2 Dataset Information

We gathered 2683 malware sample that covers all the standard malware classifications and 2501 benign programs for experiments. The malicious sample collected from VXHeaven, whereas the benign sample gathered from multiple origins, that includes a clean Windows OS, download.com3, and onlinedown.net.4. Table Shows dataset used in the Proposed System.

[1]https://github.com/ajaykumar121182/malware

5.3 Selected PE File Features

Since feature set is too large and some bigger features challenging to operate on an algorithm [49], feature selection aims to eliminate non-essential features from features set without affecting the accuracy rate. We extracted 65 features and presented them in a combined matrix. We understood that the size of the dataset is immense and take around two and a half hour for the load it. Therefore it is essential to remove non-essential features. The following are feature selection methodology. Wrapping methodology:- Wrapping methodology chooses different combinations of features and check again model. The combination which gives a higher accuracy rate is kept in a dataset. Filtering methodology:- Filter methodology runs the features utilizing statistics. Feature with the lower run is eliminated, While features with high run kept in a dataset. Embedded methodology:- Embedded methodology asses the top feature used when the model is applied. After running the algorithm, we selected Top 20 features are given in Table 9:

Algorithm 1 Feature Selection algorithm for proposed system

```
1:  procedure START(F,Tf)     ▷ The F is set of random group of Features and tf is set of top
    selected features
2:  Tf= $\psi$ and F$\neq$ $\psi$
3:  For each $fi$ in $F$ and $tfi$ in $Tf$
4:  Repeat
5:  $Train - RandomForestfi$
6:  $Weight \leftarrow MDA(fi)$     ▷ Apply feature important measure using
    $MDA - meandecreasesaccuracy$
7:  if($InitialWeight > Weight$) then
8:  Remove($fi$)
9:  Else
10: $tfi \leftarrow fi$
11: End if
12: Until all features $Fi$ in $F$ either selected or removed after particular iteration of Random
    Forest
13: End For
14: return $tf$
15: EndProcedure
```

5.4 Performance Criteria

The following are the essential measurement for malware detection.

- TP:- The TP rate estimates the level of malware tests that are marked accurately as malware.

Table 9 Data set structure

PEfile Structure	Random Forest	Gradient Boosting
IMAGE_DOS_HEADER	IMAGE_DOS_HEADER e-csum, e_data, e_oemid E_ifanew,e_ip	E_ifanew,e_ip
FILE_HEADER	Filesize,fileinfo, fk_char12	Filesize,fileinfo,fH_char2
OPTIONAL_HEADER	MajorlinkerVersion, MinoprlinkerVersion, AddressofEntryPoint	AddressOfEntryPoint, sussections, packer_typeNETExecuta

- FP:- The FP rate measures the number of authenticating sample that is erroneously marked as malware.
- FN:- The FN rate measures the number of malware tests erroneously marked as clean
- TN:- The TN rate measures the number of clean samples that are accurately marked as clean.
- Sensitivity:- The Sensitivity estimates fraction of authenticating malware sample that is marked accurately as malware.

$$Sensitivity = TP/(TP + FN) \tag{1}$$

- Specificity:- The Specificity estimates the fraction of authenticating clean sample that is marked accurately as clean.

$$Specificity = TN/(TN + FP) \tag{2}$$

- Accuracy-: The final accuracy to detect malware and clean sample accurately

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{3}$$

5.5 Algorithm

- Probability of targeted malware calculated using following equation

$$P(M, m) = \sum_{n=i}^{M} P(i).E(i) \tag{4}$$

$$E(M) = \sum_{n=i}^{c} -P(i)log_2 pi \tag{5}$$

Algorithm 2 Training algorithm

```
1: procedure START(S,R)    ▷ The S is set of malware and clean sample and R is set of decision tree
2: D= $\psi$ and S$\neq$ $\psi$
3: For each $Di$ in $D$ and $Si$ in $S$
4: Repeat
5: Features-Value $> 0$
6: Loading dataset into tree
7: Repeat-1
8: $MaxInformationGain \leftarrow 0$
9: $Split \leftarrow \psi$
10: $p \leftarrow Probability(Features - values)$    ▷ Probability refers to the level of uncertainty in the data content
11: For all Features-values $f$ in $Si$ do
12: $IGi \leftarrow InformationGain(f, p)$
13: if($IGi > MaxInformationGain$) then
14: $MaxInformationGain \leftarrow IGi$
15: $Split \leftarrow f$
16: End if
17: End For
18: Partition(Si,Ri,Split)
19: Until-1 all partition completed
20: Until complete dataset get partitioned into random forest
21: End For
22: return $R$
23: EndProcedure
```

- The information gain is difference between initial probability and the proportional sum of the probability of branches InformationGain(M,m) = E(M)-P(M, m)

5.6 Model Evaluation

Firstly we extracted feature through dynamic analysis and prepared Malware dataset accordingly ML and Deep learning Model needs. We set binary value 1 for the malicious sample and 0 for the clean sample. This Binary vector will be acting as input to varied ML classifying algorithm (see Figure 1).

We utilized the python anaconda navigator platform for training and testing samples on ML and deep learning algorithms. This platform provided various interesting tools such as pre-processing, clustering, regression, classification, visualization, and association rules (see Figure 2). We fetched binary vector of all data into a table labeled as a sample dataset. We optimized

Table 10 Comparison table for proposed system over existing system for windows platform

Algorithm	Top Accuracy	Feature	FP
Random Forest	0.997	20	0.07
Gradient Boosting	0.994	20	0.02
DeepLearning Convolution	0.90215605751	65	–
Naive-Bayes	0.9969	4	0.074
Leveraging Compression	0.993	12	0.005
NBJ48	0.983	22	–

features by using information gain with a specific threshold. To validate the accuracy, we used a cross-validation technique. We compare and presented the result for corresponding ML and deep learning algorithms using FP, TN, TP, F, Sensitivity, and Predictive. We tested 5184 malware samples with 65 and 20 features. We found RF model show highest accuracy for 20 features with TP = 528, FP = 7, TN = 500, FN, Sensitivity = 0.9862 and Predictability = 0.9862, accuracy = 0.9866 Followed by model Gradient Boosting with accuracy TP = 533, TN = 506, FP = 2 and FN = 1, Sensitivity = 0.9980, Specificity = 0.9963, Precision = 0.9961 for 20 features and precision model with accuracy 0.9971. The graph represents the comparison of the Recommended Model with the current system model. We found some malware samples unable to identify which indicate some drawbacks of Dynamic analysis. Also, this malware sample hides their behavior even after executing in the virtual environment through a code obfuscation technique using polymorphism and metamorphism. We also concluded that prediction accuracy for malware detection is different for different ML and Deep learning algorithms. We understood that quantity of features selection affects the precision of the detection system, which is valuable insight logic for the researcher. We improved the performance of the ML model and algorithm by considering the top 20 features (see Table 10). This discovery helps the researcher to think deeply about malware analysis.

6 Concluding Remarks and Subsequent Works

The research, as mentioned earlier, has highlighted in comprehensive detail along with gaps which highlighted various past implemented malware detection model for the respective platform such as window, Android, and embedded platform. This survey detail compares various Ml And deep learning techniques that help malware researchers to think over gaps trough meaningful insight. The proposed behavior structure represents the way malware detection commonly carry out. Based on research investigation, we found that the implemented proposed model, along with the Feature Optimization technique, improves the classification accuracy rate of malware detection over existing models. Top accuracy for window-based platform includes 0.997 with 20 features for the Random Forest model. The features selection technique required malware research effort, not an utterly automated process. The dynamic analysis not covering all malware. Some malware seems to easily escape as a clean sample in the virtual environment by executing them-self passively. Feature optimization results in partial perceive malware behavior. So some false malware count as a clean sample with an increasing malicious rate, it an enormous task to maintain malicious sample at voluminous level. To investigate this ML model to optimized feature selection without being lost any crucial information in order to improve performance and accuracy. To process multiple malware model in Parallel manner in order to perform detection at a larger-scale level.

6.1 Dataset

Dataset and scripts for reproducing the results were uploaded at Github repository – https://github.com/ajaykumar121182/malware.

References

[1] Dali Zhu, Hao Jin, Ying Yang, Di Wu, and Weiyi Chen. Deepflow: Deep learning-based malware detection by mining android application for abnormal usage of sensitive data. In *2017 IEEE symposium on computers and communications (ISCC)*, pages 438–443. IEEE, 2017.

[2] Abhijeet Thakare, Euijong Lee, Ajay Kumar, Valmik B Nikam, and Young-Gab Kim. Parbac: Priority-attribute-based rbac model for azure iot cloud. *IEEE Internet of Things Journal*, 7(4):2890–2900, 2020.

[3] Mayur Rahul, Narendra Kohli, Rashi Agarwal, and Sanju Mishra. Facial expression recognition using geometric features and modified hidden markov model. *International Journal of Grid and Utility Computing*, 10(5):488–496, 2019.

[4] Devottam Gaurav, Sanju Mishra Tiwari, Ayush Goyal, Niketa Gandhi, and Ajith Abraham. Machine intelligence-based algorithms for spam filtering on document labeling. *Soft Computing*, pages 1–14, 2019.

[5] Zahoor-Ur Rehman, Sidra Nasim Khan, Khan Muhammad, Jong Weon Lee, Zhihan Lv, Sung Wook Baik, Peer Azmat Shah, Khalid Awan, and Irfan Mehmood. Machine learning-assisted signature and heuristic-based detection of malwares in android devices. *Computers & Electrical Engineering*, 69:828–841, 2018.

[6] Ajay Kumar, Kumar Abhishek, Amit Kumar Singh, Pranav Nerurkar, Madhav Chandane, Sunil Bhirud, Dhiren Patel, and Yann Busnel. Multilabel classification of remote sensed satellite imagery. *Transactions on Emerging Telecommunications Technologies*, page e3988, 2020.

[7] Sanju Mishra, Rafid Sagban, Ali Yakoob, and Niketa Gandhi. Swarm intelligence in anomaly detection systems: an overview. *International Journal of Computers and Applications*, pages 1–10, 2018.

[8] Saiteja Prasad Chatrati, Gahangir Hossain, Ayush Goyal, Anupama Bhan, Sayantan Bhattacharya, Devottam Gaurav, and Sanju Mishra Tiwari. Smart home health monitoring system for predicting type 2 diabetes and hypertension. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[9] Pranav Nerurkar, Madhav Chandane, and Sunil Bhirud. Survey of network embedding techniques for social networks. *Turkish Journal of Electrical Engineering & Computer Sciences*, 27(6):4768–4782, 2019.

[10] Pranav Nerurkar, Aruna Pavate, Mansi Shah, and Samuel Jacob. Performance of internal cluster validations measures for evolutionary clustering. In *Computing, Communication and Signal Processing*, pages 305–312. Springer, 2019.

[11] Shahid Alam, Zhengyang Qu, Ryan Riley, Yan Chen, and Vaibhav Rastogi. Droidnative: Automating and optimizing detection of android native code malware variants. *computers & security*, 65:230–246, 2017.

[12] Hisham Shehata Galal, Yousef Bassyouni Mahdy, and Mohammed Ali Atiea. Behavior-based features model for malware detection. *Journal of Computer Virology and Hacking Techniques*, 12(2):59–67, 2016.

[13] Monire Norouzi, Alireza Souri, and Majid Samad Zamini. A data mining classification approach for behavioral malware detection. *Journal of Computer Networks and Communications*, 2016, 2016.

[14] Aashima Malhotra and Karan Bajaj. A hybrid pattern based text mining approach for malware detection using dbscan. *CSI transactions on ICT*, 4(2-4):141–149, 2016.

[15] Zhiqiang Li, Lichao Sun, Qiben Yan, Witawas Srisa-an, and Zhenxiang Chen. Droidclassifier: Efficient adaptive mining of application-layer header for classifying android malware. In *International Conference*

on Security and Privacy in Communication Systems, pages 597–616. Springer, 2016.

[16] Muazzam Siddiqui, Morgan C Wang, and Joohan Lee. A survey of data mining techniques for malware detection using file features. In *Proceedings of the 46th annual southeast regional conference on xx*, pages 509–510, 2008.

[17] Yuxin Ding, Xuebing Yuan, Ke Tang, Xiao Xiao, and Yibin Zhang. A fast malware detection algorithm based on objective-oriented association mining. *Computers & security*, 39:315–324, 2013.

[18] Chun-I Fan, Han-Wei Hsiao, Chun-Han Chou, and Yi-Fan Tseng. Malware detection systems based on api log data mining. In *2015 IEEE 39th annual computer software and applications conference*, volume 3, pages 255–260. IEEE, 2015.

[19] Pranav Nerurkar, Madhav Chandane, and Sunil Bhirud. A comparative analysis of community detection algorithms on social networks. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*, pages 287–298. Springer, 2019.

[20] Pranav Nerurkar, Madhav Chandane, and Sunil Bhirud. Community detection using node attributes: A non-negative matrix factorization approach. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*, pages 275–285. Springer, 2019.

[21] Munkhbayar Bat-Erdene, Hyundo Park, Hongzhe Li, Heejo Lee, and Mahn-Soo Choi. Entropy analysis to classify unknown packing algorithms for malware detection. *International Journal of Information Security*, 16(3):227–248, 2017.

[22] Tobias Wüchner, Aleksander Cisłak, Martin Ochoa, and Alexander Pretschner. Leveraging compression-based graph mining for behavior-based malware detection. *IEEE Transactions on Dependable and Secure Computing*, 16(1):99–112, 2017.

[23] Qiguang Miao, Jiachen Liu, Ying Cao, and Jianfeng Song. Malware detection using bilayer behavior abstraction and improved one-class support vector machines. *International Journal of Information Security*, 15(4):361–379, 2016.

[24] Mojtaba Eskandari, Zeinab Khorshidpour, and Sattar Hashemi. Hdm-analyser: a hybrid analysis approach based on data mining techniques for malware detection. *Journal of Computer Virology and Hacking Techniques*, 9(2):77–93, 2013.

[25] Stavros D Nikolopoulos and Iosif Polenakis. A graph-based model for malware detection and classification using system-call groups. *Journal of Computer Virology and Hacking Techniques*, 13(1):29–46, 2017.

[26] Pranav Nerurkar, Archana Shirke, Madhav Chandane, and Sunil Bhirud. A novel heuristic for evolutionary clustering. *Procedia Computer Science*, 125:780–789, 2018.

[27] Jiang Ming, Zhi Xin, Pengwei Lan, Dinghao Wu, Peng Liu, and Bing Mao. Impeding behavior-based malware analysis via replacement attacks to malware specifications. *Journal of Computer Virology and Hacking Techniques*, 13(3):193–207, 2017.

[28] Pranav Nerurkar, Archana Shirke, Madhav Chandane, and Sunil Bhirud. Empirical analysis of data clustering algorithms. *Procedia Computer Science*, 125:770–779, 2018.

[29] Shina Sheen, R Anitha, and V Natarajan. Android based malware detection using a multifeature collaborative decision fusion approach. *Neurocomputing*, 151:905–912, 2015.

[30] Altyeb Altaher. An improved android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (ehnfc) and permission-based features. *Neural Computing and Applications*, 28(12):4147–4157, 2017.

[31] Weixuan Mao, Zhongmin Cai, Don Towsley, Qian Feng, and Xiaohong Guan. Security importance assessment for system objects and malware detection. *Computers & Security*, 68:47–68, 2017.

[32] Hashem Hashemi, Amin Azmoodeh, Ali Hamzeh, and Sattar Hashemi. Graph embedding as a new approach for unknown malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(3):153–166, 2017.

[33] Songyang Wu, Pan Wang, Xun Li, and Yong Zhang. Effective detection of android malware based on the usage of data flow apis and machine learning. *Information and software technology*, 75:17–25, 2016.

[34] Igor Santos, Felix Brezo, Xabier Ugarte-Pedrero, and Pablo G Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, 231:64–82, 2013.

[35] Abhishek Bhattacharya and Radha Tamal Goswami. Dmdam: data mining based detection of android malware. In *Proceedings of the first international conference on intelligent computing and communication*, pages 187–194. Springer, 2017.

[36] Abhishek Bhattacharya and Radha Tamal Goswami. Comparative analysis of different feature ranking techniques in data mining-based android malware detection. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*, pages 39–49. Springer, 2017.

[37] Alejandro Martín, Héctor D Menéndez, and David Camacho. Mocdroid: multi-objective evolutionary classifier for android malware detection. *Soft Computing*, 21(24):7405–7415, 2017.

[38] Aya Hellal and Lotfi Ben Romdhane. Minimal contrast frequent pattern mining for malware detection. *Computers & Security*, 62:19–32, 2016.

[39] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, and Yang Liu. A multi-view context-aware approach to android malware detection and malicious code localization. *Empirical Software Engineering*, 23(3):1222–1274, 2018.

[40] Baojiang Cui, Haifeng Jin, Giuliana Carullo, and Zheli Liu. Service-oriented mobile malware detection system based on mining strategies. *Pervasive and Mobile Computing*, 24:101–116, 2015.

[41] Yanfang Ye, Lingwei Chen, Shifu Hou, William Hardy, and Xin Li. Deepam: a heterogeneous deep learning framework for intelligent malware detection. *Knowledge and Information Systems*, 54(2):265–285, 2018.

[42] Bin Wu, Tianliang Lu, Kangfeng Zheng, Dongmei Zhang, and Xing Lin. Smartphone malware detection model based on artificial immune system. *China Communications*, 11(13):86–92, 2014.

[43] James B Fraley and Marco Figueroa. Polymorphic malware detection using topological feature extraction with data mining. In *SoutheastCon 2016*, pages 1–7. IEEE, 2016.

[44] Mohamed El Boujnouni, Mohamed Jedra, and Noureddine Zahid. New malware detection framework based on n-grams and support vector domain description. In *2015 11th international conference on information assurance and security (IAS)*, pages 123–128. IEEE, 2015.

[45] Amine Boukhtouta, Serguei A Mokhov, Nour-Eddine Lakhdari, Mourad Debbabi, and Joey Paquet. Network malware classification comparison using dpi and flow packet headers. *Journal of Computer Virology and Hacking Techniques*, 12(2):69–100, 2016.

[46] Zhenlong Yuan, Yongqiang Lu, and Yibo Xue. Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, 21(1):114–123, 2016.

[47] Aziz Mohaisen, Omar Alrawi, and Manar Mohaisen. Amal: High-fidelity, behavior-based automated malware analysis and classification. *computers & security*, 52:251–266, 2015.

[48] Ping Wang and Yu-Shih Wang. Malware behavioural detection and vaccine development by using a support vector model classifier. *Journal of Computer and System Sciences*, 81(6):1012–1026, 2015.

[49] Mozammel Chowdhury, Azizur Rahman, and Rafiqul Islam. Malware analysis and detection using data mining and machine learning classification. In *International Conference on Applications and Techniques in Cyber Security and Intelligence*, pages 266–274. Springer, 2017.

Biographies

Ajay Kumar is a senior Network/IT Analyst, working with Government of India (Ministry of Defense). He has B-Tech from Central University Delhi with distinction and M Tech from VJTI, Mumbai with distinction. He is a Ph.D. scholar of Dept. of Computer Science & Engineering, NIT Patna. His area of research is network security, Authentication, IoT and Machine Learning. He has published more than 20 research papers in various renowned International conferences and SCI indexed journals.

Kumar Abhishek is working as an Assistant Professor, Department of Computer Science and Engineering, National Institute of Technology Patna, India. His area of interest lies in RDF, Semantic Web, Ontology, Semantic Sensor Web, Ontology mapping and Approximation. He has published more than 100 research papers in various renowned International conferences and SCI indexed journals.

Shishir Kumar Shandilya is the Division Head of Cyber Security and Digital Forensics at Vellore Institute of Technology, VIT Bhopal University, India. He is also a Visiting Research Fellow at Liverpool Hope University-United Kingdom, a Cambridge University Certified Professional Teacher Trainer, ACM Distinguished Speaker and a Senior Member of IEEE. He is an Academic Advisor to National Cyber Safety Security Standards, New Delhi. He has received IDA Teaching Excellence Award for distinctive use of technology in Teaching by Indian Didactics Association, Bangalore (2016) and Young Scientist Award for two consecutive years, 2005 and 2006, by Indian Science Congress MP Council of Science Technology. He has seven books published by Springer Nature-Singapore, IGI-USA, River-Denmark and Prentice Hall of India. His recently published book is on Advances in Cyber Security Analytics and Decision Systems by Springer.

Muhammad Rukunuddin Ghalib currently works at the Division of Analytics, VIT University. Dr. Muhammad does research in Artificial Neural Network, Data Mining and Computing in Mathematics, Natural Science, Engineering and Medicine. Currently working on IOT based artificial rain creation.