Machine Learning Techniques to Detect and Classify Ransomware

Pavithra S[1] Pooja N[2] Shreya S[3] Sharmili S[4]

[1,2,3,4]Department of Information Science and Engineering

[1,2,3,4]East West Institute of Technology, Bengaluru, India

Abstract— The system mainly focuses on the malware where it is a type of malicious software called ransomware designed in such a way to block access to a computer system until a sum of money is paid. The automatic classification system which has high-performance, high accuracy and efficiency based on multi-feature selection fusion of machine learning is proposed in this project. The manual heuristic inspection of malware analysis is not considered to be effective and efficient when it is compared against the high spreading rate of malware which is a serious threat. Hence, using machine learning techniques the automated behavior based malware detection is considered to be a profound solution. The classifiers used in this project are K-Nearest Neighbors (KNN), Random Forests, and Decision Tree. Since, many antiviruses have heavy impact on the user system and sometimes they are not able to detect new emerging harmful threats like ransomware and on the other hand, light weight antiviruses are not so effective in detecting and preventing malwares. As the data security with no heavy impact on the user system is our main concern. Our model can be a cloud based malware classification model. Therefore, without installing any third party application on user system, we can test and predict whether any suspicious file is a malware or not.

Keywords: Accuracy, Automatic Classification, Decision Tree, Efficiency, Feature Extraction, High-Performance, K-Nearest Neighbors, Machine Learning, Malware, Random Forest, Ransomware

I. INTRODUCTION

Machine learning is a branch of artificial intelligence, a science that researches the machines where new knowledge can be acquired along with new skills and to identify existing knowledge.

The precise definition of machine learning is: " It's a computer program learning from experience E with respect to some task T along with some performance measure P, if its performance on T which is measured by P, improves with E."

Machine learning has been widely used in data mining, computer vision, natural language processing, securities market analysis, biometrics, search engines, medical diagnostics, detection of credit card fraud, DNA sequence sequencing, speech and handwriting recognition, strategy games and robotics.

Ransomware is a malware that stealthily gets installed on the device and holds the files or operating system functions for ransom. It restricts the user from using PC or mobile device, by encrypting the files. Paying ransom (through Bitcoins) does not guarantee that files are returned. Prevention is still way better than allowing to be infected and then trying to find a cure. This paper aims at developing a self-learning algorithm to detect Ransomware using Machine Learning Techniques at various stages be it the Initial level, Network level or the Encryption Phase. Our approach will be to have pro-active defense or detect early Intrusion.

II. LITERATURE SURVEY

Several researchers has proposed and implemented detection of ransomware using different approaches of machine learning.

1) A Novel Solutions for Malicious Code Detection and Family Clustering based on Machine Learning by Hangfeng Yang, Shudong Li, Xiaobo Wu, Hui Lu, Weihong Han: Malware has become a major threat to cyberspace security because of the continuously created and produced malicious code. In this paper, Authors propose two novel methods to solve the malware identification problem. To solve malware classification which is different from traditional machine learning, our method introduces the ensemble models where the malware classification problem can be solved. Also to solve malware family clustering which is different from the classic malware family clustering algorithm, our method introduces the t-SNE algorithm to extract the feature data and then determines the number of malware families.
2) A3CM: Automatic Capability Annotation for Android Malware by Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, Yu Wang, Yang Xiang: Android malware poses serious security and privacy threats to the mobile users. Traditional malware detection and family classification technologies are not much effective due to the rapid evolution of malware landscape, with the emerging of so-called zero-day-family malware families. To address this issue, authors presents a novel research problem on identifying automatically the security related capabilities of any detected malware, which authors refer to as Malware Capability Annotation (MCA).
3) Build a Roadmap for Stepping Into the Field of Anti-Malware Research by Weijie Han, Jingfeng Xue, Yong Wang, Shibing Zhu, Zixiao Kong: The principle of secret suffrage be ensured when voters are offered the likelihood to cast their votes using internet voting with the steady introduction of various forms of remote electronic voting since 2000, it’s become apparent that internet voting fails at providing the privacy guarantees offered by traditional paper-based voting systems. Against this assumption, the present proposal suggests reviewing the standard configuration of the principle of vote secrecy. With this in mind, the proposal will assess current accepted standards on voters’ anonymity for traditional and internet-based voting systems evaluate the core elements of lawful relaxations to the principle of secret suffrage, and particularly those traditionally associated to different types of remote voting, and assess whether or not they may be applied to internet voting.

4) A Hierarchical Convolutional Neural Network for Malware Classification by Daniel Gibert, Carles Mateu, Jordi Planes: Malware detection and classification may be a challenging problem and an energetic area of research. Particular challenges include a way to best treat and preprocess malicious executables so as to feed machine learning algorithms. Novel approaches within the literature treat an executable as a sequence of bytes or as a sequence of programing language instructions. However, in those approaches the hierarchical data structure of programs isn't taken into consideration. An executable exhibits different levels of spatial correlation. Adjacent code instructions are correlated spatially but that's not necessarily the case. Function calls along with the jump commands transfer the control of the program to a unique point within the instruction stream. Furthermore, when treating the binary as a sequence of byte values these discontinuities are maintained. Additionally, functions can be arranged randomly if addresses are correctly reorganized.
5) A Cause-Based Classification Approach for Malicious DNS Queries Detected through Blacklists by Akihiro Satoh, Yutaka Nakamura, Yutaka Fukuda, Kazuto Sasai, Gen Kitagata: Some of the foremost serious security threats facing computer networks involve malware. To forestall this threat, administrators must swiftly remove the infected machines from their networks. One common way which is used to detect infected machines in a very network is by monitoring communications supported blacklists. However, detection using this method has the subsequent two problems: no blacklist is totally reliable, and blacklists don't provide sufficient evidence to permit administrators to see the validity and accuracy of the detection results. However, in those approaches the data structure of programs isn't taken into consideration. An executable exhibits different levels of spatial correlation. Adjacent code instructions are correlated spatially but that's not necessarily the case. Function calls along with the jump commands transfer the control of the program to a unique point within the instruction stream.

III. EXISTING SYSTEM

The existing system has the subsequent two problems in malware detection by using machine learning.

With the dynamic behaviors of an outsized number of PC malware viruses, the primary main important problem is a way to extract malware features of malware from documents and train classifier to classify a virus supporting these features. The documents are the xml files outputted by an exe file after the sandbox is run is employed to see whether the file is malware. So as to solve this problem, the key two parts are the feature engineering and also the training classifier, and here authors used the ensemble strategy to construct the classifier with high accuracy.

According to the dynamic behaviors of an oversized number of PC malware viruses, the second important problem is the way to quickly identify and accurately detect the identical family's malware mutation, namely the malware family clustering, where the corresponding clustering algorithm is meant to evaluate the family id of the xml file output by an exe file after passing through the sandbox operation. And therefore the key point for this problem is that the determination of the quantity of cluster centers k.

IV. PROPOSED METHOD

To develop a deeper understanding, it's worth inquiring the final workflow of the machine learning process, which is shown below. The method consists of 5 stages shown in Fig 4.1

A. Data intake:

Initially, the dataset is loaded from the file and is saved in memory.

B. Data transformation:

At this time, the data that was loaded at the previous step is transformed, normalized and cleared to be suitable for the algorithm. Data is converted in order that it lies within the same range, has the identical format, etc. At this time, feature extraction and selection, which are discussed further, are performed. Additionally, the information is separated into sets –'training set' and 'test set'. Data from the training set is employed to create the model.

C. Model Training:

At this stage, a model is constructed using the chosen algorithm.

D. Model Testing:

The model that was built or trained during step 3 is tested using the test data set, and also the produced result's used for building a replacement model, that might consider previous models.

E. Model Deployment:

At this stage, the most effective model is chosen either after the defined number of iteration or as soon as the requireded result is achieved.

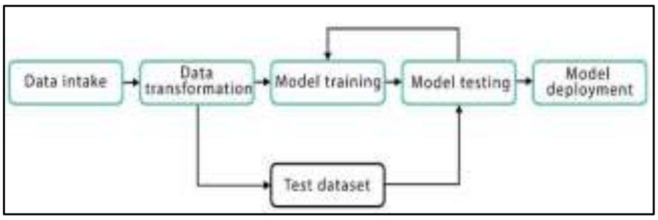


Fig. 4.1: General workflow process

F. Algorithms Used

1) K-NEAREST NEIGHBORS:

The KNN is a non-parametric algorithm, where it does not make any assumptions about the data structure. In real world problems, data rarely obeys the general assumptions which are theoretical, making this algorithms a good solution for such problems. KNN model representation is as simple as the dataset, there is no requirement of learning, and the entire training set is already stored. This KNN algorithm can be used for both classification and regression problems where the prediction is based on the k training instances that are closest to the input instance. In the KNN classification problem, the output would be a class, which includes the input instance, predicted by the majority vote of the k closest neighbors. In the regression problem, the output would be the value of the property, where it is generally a mean value of the k nearest neighbors. The schematic example of the KNN is outlined in Fig 4.2.

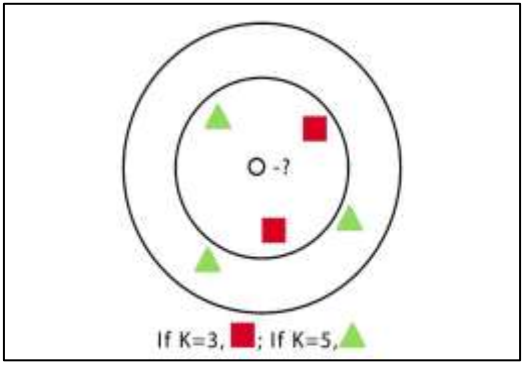


Fig. 4.2: KNN example

Various distance measurement methods are used for locating the nearest neighbors. The popular ones include Hamming Distance, Manhattan Distance, Minkowski distance:

Equations:

$$\textit{Hamming Distance:}\ d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \qquad [1]$$

$$\textit{Manhattan Distance:}\ d_1(p,q) = ||p-q||_1 = \sum_{i=1}^{n} |p_i - q_i| \qquad [2]$$

$$\textit{Minkowski Distance} = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p} \qquad [3]$$

Euclidian distance is good for the problems, where the features are of the same type. For the features of various types, it is advised to use, for example, Manhattan Distance. For the classification problems, the output can also be presented as a set of probabilities of the instance which belongs to the class. For example, for binary problems, the probabilities can be calculated like $P(0) = N0$, where P(0) is $N0+N1$ The probability of the 0 class membership and $N0$, $N1$ are numbers of neighbors that are in the classes 0 and 1 respectively. The value of k plays a important role in the prediction accuracy of the algorithm. However, selecting the k value is a not an easy task. Smaller values of k will result in lower accuracy in the datasets with much noise, since every instance in the training set will be having a higher weight during the decision process. Higher values of k lowers the performance of the algorithm. In addition to that, if the value is too high, the model can over fit, making the class boundaries less distinct and resulting in lower accuracy again. As a general approach, it is advised to select k using the formula below: k [5] For classification problems with an even number of classes, it is advised to select an odd k since it will eliminate the possibility of a tie during the majority vote. The drawback of the algorithm is the low performance on the unevenly distributed datasets. Thus, if one class vastly dominates the other ones, it is more likely to have more neighbors of that class due to the large number and therefore making more incorrect predictions.

2) RANDOM FOREST:

Random Forest is one among the foremost popular machine learning algorithms. It doesn’t requires any data preparation and modeling but usually leads to accurate results. Random Forests are based on the decision trees. To be specific, Random Forests are the combinations of decision trees, producing a better prediction accuracy. That is why it is called a ’forest’ – it is basically a set of decision trees. The essential idea is to increase multiple decision trees based on the independent subsets of the dataset. At each node, selection of n variables out of the feature set is done randomly, and the best split on these variables is found. In simple words, the algorithm can be described as follows:

1) Multiple trees are built roughly on the two third of the training data (62.3%). Data is chosen randomly.
2) Selection of several predictor variables out of all the predictor variables is done randomly. Then, the best split on these selected variables is used to split the node. By default, the amount of the selected variables is given by the square root of the total number of all predictors for classification, and it is constant for all trees.
3) By using remaining data, the misclassification rate is calculated. The overall out-of-bag error rate is the total error rate that is calculated.
4) Each tree which is trained gives its own classification result, giving its own “votes”.

The class that have highest accuracy is chosen as the result. The scheme of the algorithm is seen in Fig 4.3.

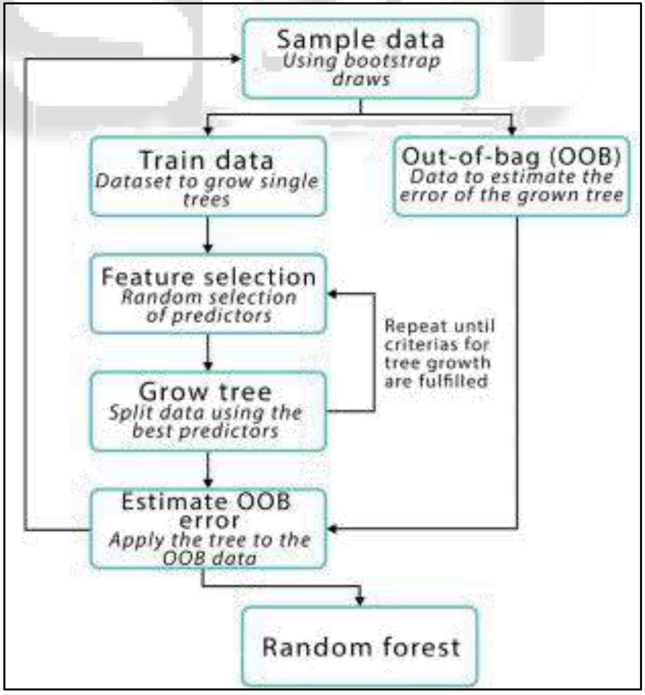
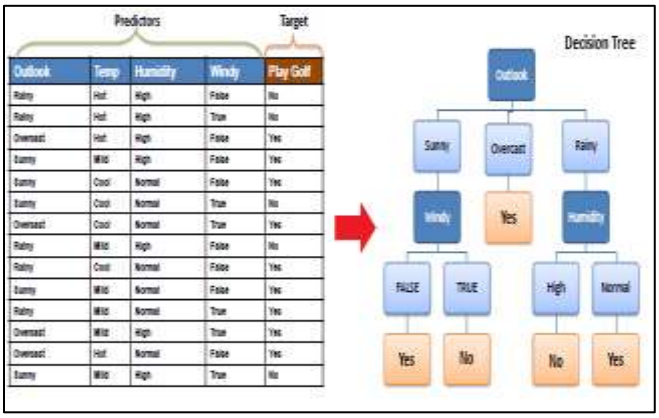

Sample data
Using bootstrap draws
Train data
Dataset to grow single trees
Out-of-bag (OOB)
Data to estimate the error of the grown tree
Feature selection
Random selection of predictors
Repeat until criterias for tree growth are fulfilled
Grow tree
Split data using the best predictors
Estimate OOB error
Apply the tree to the OOB data
Random forest

Fig. 4.3: Random Forest scheme

3) DECISION TREE:

Decision tree creates classification or regression models within the type of a tree structure. It splits a dataset into smaller subsets with increase in exhaustive of tree. The ultimate result’s a tree with decision nodes and leaf nodes. A call node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in tree which corresponds to simplest predictor is named ssroot node.



Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Fig. 4.4: Classification of decision tree

Types of decision trees

- Categorical Variable Decision Tree: Decision Tree which has categorical target variable is called as categorical variable decision tree.
- Continuous Variable Decision Tree: The decision Tree which has continuous target variable then it's called as Continuous Variable Decision Tree.

V. ARCHITECTURE

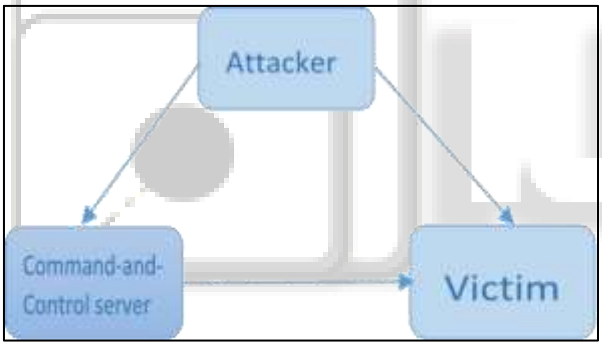


Fig. 5.1: Structure of an attack

The attacker lures the victim using phishing mails, compromised internet sites using exploit kits. Exploit kits may be a collection of exploits which are delivered continuously in an exceedingly pattern to the target machine.

The attack pattern is also generalized in two ways- one where human intervention is critical like attacks based upon social engineering (phishing e- mails) and one where there's no human intervention is required, based upon exploiting vulnerabilities using specific exploit kits.

Once a machine gets infected, it contacts its Command-and-Control(C&C) server for the payload. The payload delivered may carry certain information prefers it may have a code to test for other vulnerable machines, encryption key from the server to encrypt files, information regarding what files are to be encrypted, when the encrypting process should begin etc. curbing the malware at the network level itself. Different families of malware use different techniques to speak with its C&C server. So, if their communication is intercepted by analysing the TCP/UDP packets exchanged, an algorithm are often deployed which could then be used for pro-active defence. This algorithm will analyse patterns of communication with server and predict the following advanced version of malware.

Next the attacker has gained control of the infected machine, has installed and created a backdoor. It's also possible to detect ransomware at this stage and block it at the system level. Patterns will be generated on the premise of how malicious code interacts with the victim machine's kernel (operating system). This can help us form a network cycle which depicts where the attack originated, route taken by the packets, what forms of packets are received and are they encrypted or not, or what port numbers they use thus providing us with network reconnaissance. Our approach is to make the network cycle supported previous ransomwares for pro-active defence.

Once these patterns are analysed using dynamic and intelligent sandboxing (real time execution of ransomware), it can be used to predict future pattern of attacks. These patterns may also be used to kill the supply chain of the malware in pro-active defence.

There is also another high level approach- Cryptology, cracking the encryption. However this approach requires a huge infrastructure and is not covered in this paper.

Thereafter, ensembling techniques can be applied to the patterns collected to generate our vulnerability (Ransomware) detecting Machine learning algorithm.

VI. IMPLEMENTATION

If we see literature of detection malware methods based on machine learning methods different models are successfully used and various machine learning methods are found useful in the liberation of malware detection and classification. Some of these techniques are Support vector machines, ensemble models like random forest, naive Bayes and also we can group together different malware families based on their hidden behavior using the unsupervised techniques for example clustering like K-means, but the main challenge is to decide the number of clusters or groups.

In order to extract information in the format of the object from the portable executable header, a library within Bin-Utile that was lobed. Some of the features we obtained from here are size of the files, also the names of Dynamically linked libraries and Dynamically linked libraries function calls. With the Portable Executable approach, some other features are list of DLLs used by the binary and also the count system calls within each Dynamically linked libraries are extract is used as a feature.

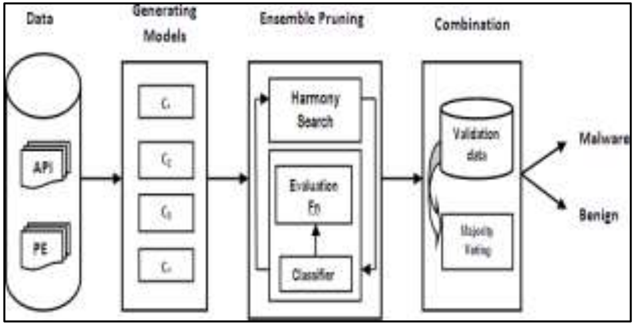


Fig. 6.1: Proposed Model for Ransomware Detection Using Ensemble Pruning

The schematic representation of the proposed model for ransomware detection using ensemble pruning is shown in the above fig 6.1.

1) Data: The discriminating features from the PE files and the API call features from the binary of a program are executed to form a dataset D.(x1,y1) and provides to the n classifier C1,C2----Cn. Here x1 is a vector and y1 is the target variable, which indicates whether the given file is malicious or benign.
2) Generating Models: Typically, an ensemble is constructed in two steps. First, we consider a number of base learners which are trained in parallel or sequentially. The problem of ensemble pruning is to find the best subset of such that the combination of the selected classifiers will have the highest possible degree of accuracy.
3) Spruning the Ensemble: Let be an ensemble of classifiers. Is a classifier that can predict the class of an observation. The problem of ensemble pruning is to find the best subset of such that the combination of the selected classifiers will have the highest possible degree of accuracy.

VII. Results

An experimental evaluation of the proposed ransomware detection system is as shown below. We performed the experiment using algorithms result where it is analyzed accordingly. The algorithm is also applied on malicious data set as a single entity against legitimate dataset. The results shows that the proposed system was able to detect anonymous states of the virtual machine, as well as the presence of both known and unknown ransomware. Comparing our solution with the most relevant ones of the state and also by considering different criteria. The outputs of both methods are sent to the decision and reaction module which will interpret the results and will react in consequence to mitigate the potential ransomware attack

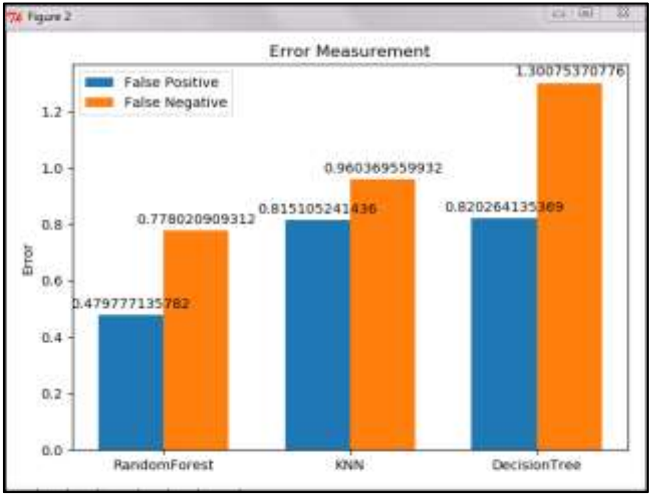
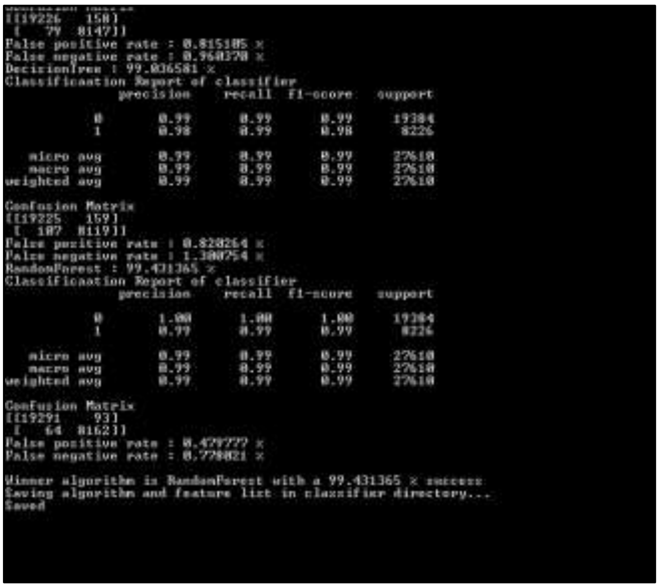


Fig. 7.1: Graph for algorithms obtained by finding accuracy


```
[[19226   158]
 [   79  8147]]
False positive rate : 0.815105 %
False negative rate : 0.960370 %
DecisionTree : 99.036581 %
Classification Report of classifier
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19384
           1       0.98      0.99      0.98      8226

   micro avg       0.99      0.99      0.99     27610
   macro avg       0.99      0.99      0.99     27610
weighted avg       0.99      0.99      0.99     27610

Confusion Matrix
[[19225   159]
 [  107  8119]]
False positive rate : 0.820264 %
False negative rate : 1.300754 %
RandomForest : 99.431365 %
Classification Report of classifier
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19384
           1       0.99      0.99      0.99      8226

   micro avg       0.99      0.99      0.99     27610
   macro avg       0.99      0.99      0.99     27610
weighted avg       0.99      0.99      0.99     27610

Confusion Matrix
[[19291    93]
 [   64  8162]]
False positive rate : 0.479777 %
False negative rate : 0.778021 %

Winner algorithm is RandomForest with a 99.431365 % success
Saving algorithm and feature list in classifier directory...
Saved
```

Fig. 7.2: Testing and Training algorithm and determining the winner algorithm

Fig. 7.3: The Login Page where the registration is done

Fig. 7.4: After Logging In the Page Will Be Displayed To Upload Files

Fig. 7.5: Check Whether the Uploaded File Is Safe Or Malware.

Fig. 7.6: Upload the file which is safe from ransomware

Fig. 7.7: The uploaded file will be displayed in the dashboard section.

VIII. CONCLUSION

Overall, the goals that were defined for this study were finally achieved. The required feature extraction and representation methods were selected and therefore the selected machine learning algorithms were applied and evaluated.

The desired feature representation method was selected to be the combined matrix along with outlining the frequency of successful API calls and failed API calls together with the return codes for them. This was chosen, because it outlines the particular behavior of the file. Unlike other methods, it combines information about different changes in the system, including the changes in the registry, murexes, files, etc. In classification problems, different results were given for different models. The lowest accuracy was achieved by Naive Bayes and k-Nearest-Neighbors. The highest accuracy was achieved with the Decision Tree and Random Forest models, and it was equal to 99.03% and 99.43% for binary classification respectively.

As per the results of our experiment the malicious operations are positively detected in multiple samples under consideration. Analysis of multiple samples has supported our assumption positively. As an extension to this work, we will identify and consolidate the changes occurred to the system after the analysed programs are installed on different testing environments. In our future work, we will emphasize on dynamic analysis because, to discover the hidden behaviour of a packed sample it is essential to execute it. Packing information cannot be overlooked while describing malware or behaviour signature of malware but they should have little significance and should not affect detection in a negative manner. In the next work, we will use different machine learning and other algorithms as an attempt to classify the samples either as malware or benign with more accuracy.

IX. ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project. We would like to take this opportunity to thank them all. First and foremost we would like to thank Mr. Hemanth kumar K, Asst. Professor, EWIT, for his moral support and valuable suggestions and expert advice towards completing our project work and constantly guiding me to organize the project in a systematic manner. We thank our Parents, and all the Faculty members of Department of Information science & Engineering for their constant support and encouragement. Last, but not the least, we would like to thank our peers and friends who provided us with valuable suggestions to improve our project.

REFERENCES

[1] Andreas Moser Christopher Krueger and Engin Kirda. “Exploring Multiple Execution Paths for Malware Analysis”, 2007 IEEE Symposium on security and privacy(SP'07).

[2] Ivan Firdausi Charles Lin Alva Erwin, “Analysis of Machine Learning Techniques used in behavior-based Malware Detection”,2010 IEEE.

[3] Matthew G.Schultz and Eleazar Ekdin, “ Data mining methods for detection of new Malicious Executables”, 2001 IEEE International conference of international engineering.

[4] PHILIP H. SWAIN And HANS HAUSKA, “The Decision Tree classifier: Design and potential”, IEEE Transactions on Geoscience Electronics,Vol.GE-IS,No.3, july 1997.

[5] Wei Wang, Ming Bhi, Xuewen Dengan, Xiaozhou Ye,Yiqiang Sheng, Malware Traffic.

[6] Classification using convolutional neural network for representation Learning, IEEE.

[7] Bander Alsulami, Sports Mancoridis, “Behavioral Malware classification using convolutional recurrent neural networks”,IEEE 2018.

[8] David Morning,Justin Geary,Victor Sending, Sundaratajan Ezekiel,Larry Pearlstein.

[9] Laurent Nikla, “Malware Classification using Deep Convolutional Neural Network “, IEEE 2018