International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249 – 8958 (Online), Volume-9 Issue-4, April 2020

Malware Detection using ANN(Malviz.AI)

Vasu Sethia, Shivam Kataria, Jeyasekar A

Abstract: ***With an increase in the number of internet users, the number of cyber-attacks happening in organizations is increasing day by day. Most of the cyber-attacks involve the use of malicious software known as malware to steal personal information, gain unauthorized access to the computer systems and carry out malicious activities which can cause huge financial losses to the organizations. Viruses, worms, rootkits, adware or anything that performs malicious activities is classified as malware. Detecting malware is a major challenge faced by the anti-malware industry as the signature-based malware detection methods may not provide accurate detection of malware. In this paper, an artificial neural network approach for malware detection is presented to overcome the shortcomings of signature-based malware detection methods. The proposed method can be used as a base model for the malware detection process and can be further developed to enhance the functionality.***

Keywords **: *Malware, Artificial Neural Network, Machine learning, Network security, Malware Detection, Windows PE***

I. INTRODUCTION

In recent years the use of the internet has grown exponentially. Various organizations and people's daily tasks are heavily dependent on the internet. This allows hackers to develop malicious software known as malware which is used to perform malicious activities in organizations and people's systems through the internet. Malware can be defined as anything which performs unauthorized and malicious activities. Malware can be further divided into various subcategories like Viruses, ransomware, worms, rootkits, downloaders, etc. on the basis of their behavior and functionality. According to reports, the global cost of damage caused by malware is expected to reach $6 trillion by the end of 2021 [9]. Traditional malware detection engines use signature-based detection for malware detection. In signature-based detection, the malware engine generates a signature of the malware which is compared to a set of pre-computed signatures [20]. This signature is a hash value of the malware image or unique value computed from the malware image [20]. So, whenever the file enters a network or a computer system, the anti-malware engine generates the unique value of the file entered and compares that unique value against its database. If the value matches any of the entries in the database then the malware detection engine blocks the file and classifies it as malicious. The problem with this approach is that it can be easily bypassed, the change of a single byte in the malicious file or a change in the code of malicious file will change the unique value of the malware making it undetectable. Therefore, from the past few years, the anti-malware industry has started using behavioral analysis and machine learning approaches to detect malware [6]. The behavioral analysis method involves analyzing the system calls made by the malware [8], traffic generated by malware and requires the execution of a given model in a sandboxed circumstance and run-time practices are checked and logged. According to the current research, the machine learning approach requires a system to perform feature extraction and feature reduction. Features of malware can be extracted using two approaches (1) Static Analysis: In this type of technique, malware is analyzed without running the malware [6]. (2) Dynamic Analysis: In this type of technique, malware is executed in a sandbox environment and is analyzed [6]. This gives more information about the malware's behavior during the run time. In [2] the paper a method is proposed to detect malware on the basis of API calls and their arguments. It proposes a multiple-gate CNN for faster malware detection. Once the features are selected, the machine learning approach builds a classification model. All the features extracted during the feature engineering stage are transformed into a feature vector which is given as an input to the machine learning model. Recently, researchers have started to use deep learning approaches also for malware detection [5]. Previous researches use a signature-based mechanism to detect malware which can easily be bypassed by the change of a bit in the code, while the current research in the field of malware detection uses behavior-based analysis and machine learning which lacks a low false positive rate and high accuracy. Many of them apply various other machine learning classifiers on PE header features other than the standard ANN approach. Hence this paper presents an Artificial Neural Network approach for malware detection. The three most popular types of neural networks in deep learning are ANN (Artificial Neural Network), RNN (Recurrent Neural Network) and CNN (Convolution Neural Network)[17]. ANN is known as the Feed-Forward Neural network because the inputs are processed only in the forward direction. ANNs have the capacity to learn weights that map any input to the output which makes it capable of learning any complex input data and finding the relation between the input and the output. RNN's on the other hand have a recurrent connection on the hidden state which turns it from an ANN to a RNN. RNN are good at capturing the sequential information present in the input data. This makes RNN suitable for NLP (Natural Language Processing) tasks where the data is sequential. CNN is mostly used in image and video processing.

Revised Manuscript Received on April 27, 2020.
* Correspondence Author
Vasu Sethia*, B.Tech,SRM Institute of Science and Technology, Chennai, India. Email: vasu.sethia@gmail.com
Shivam Kataria, B.Tech,SRM Institute of Science and Technology, Chennai,India.Email:shivamkataria2000@gmail.com
Jeyasekar A, Associate Professor, CSE,SRM Institute of Science and Technology, Chennai,India.Email:jeyasekar.antony@gmail.com

The main features of CNNs are kernels. Kernels are used to extract the relevant features from the input using the convolution operation. In the case of malware, the data is non-linear, non-sequential. This makes the ANN to be the most suitable approach for the malware detection process.
The rest of the paper is organized as follows. In section II it presents the background work and introduction to ANN (Artificial Neural Network) and section III presents the model and proposed architecture for the malware detection process. In section IV the experimental setup is shown which is used to perform the tests, section V presents the performance and results analysis. Finally, Section VI provides the conclusion to the paper and section VII contains the references used

II. BACKGROUND WORK

The idea of artificial neural networks is derived from biological neural networks. A biological neural network consists of neurons, similarly, an artificial neural network consists of artificial neurons. It is composed of various layers connected to each other and each layer consists of various nodes connected to every other node in the next layer. The first layer is the input layer which takes the input in the form of a vector, followed by the hidden layers and at the end is the output. Each input has a weight and that weighted input is fed into a given node followed by an activation function. The output of the activation is fed as an input to the next layer. The activation function decides whether the neuron in the next layer should be activated or should not be activated on the basis of weighted input and bias. There are various types of activation functions like sigmoid [12],Relu [12],[13] etc .Inthis paper's model, ReLu is used. Its value goes from 0 to infinity and is a nonlinear function.

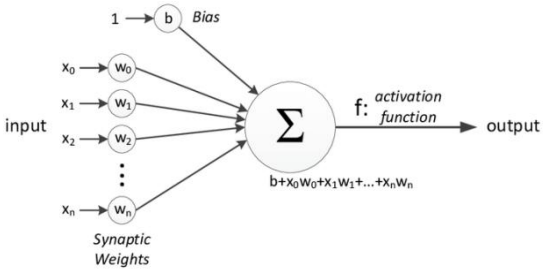

Fig 1. Single-layer Neural Network (perceptron)

The diagram (Fig 1) shows a single layer perceptron. The input values are denoted by x_0, x_1 … x_n while the weights are represented by w_1, w_2 ... w_n. These weights tell about the strength of the node and these weights are multiplied with the input values. The product of various input values and weights are then added and fed into the activation function to get the output value. This output value is then fed to the node in the next layer. The sum can be represented mathematically as,

$$\sum_0^n \quad x.w = x_0w_0+x_1w_1\ldots\ldots\ldots\ldots\ldots+x_nw_n \,(1)$$

Once the output value is produced then the difference between the actual value and the predicted value is calculated which is also known as the cost function. This cost function is then analyzed, based on this cost function weights and threshold are adjusted [18] which is sent back to the entire neural network again. This process keeps on happening until the minimum error cost function is reached.
The main justification for using this ANN model is that it can learn and demonstrate non-direct and complex connections, which is extremely significant on the grounds of establishing a relationship between input and output which are non-linear and complex. Also, it is very responsive to noise and it is easy to maintain. In the process of malware detection, input variables are the features of malware [6] while the output variable tells whether the file is malicious or not. Unlike other machine learning techniques, the ANN doesn't put any restriction on the input variables which allows the use of the maximum number of variables which aids in the malware detection process. After learning, ANN can infer the relationship on the unseen data and predict the output. In [3] the authors present a framework for malware detection which aims to get a low false-positive rate. It uses a simple multi-stage combination (cascade) of different versions of the perceptron algorithm and achieves the highest accuracy of 96.25% on 5-fold cross-validation. It achieves a low false-positive rate but does not implement various other machine learning classifiers like SVM which may give better accuracy. In [15] the author uses naive Bayes, J48, k-nearest neighbor, multi-level perceptron, decision tree, and support vector machine for malware detection. The best performance is achieved by a J48 decision tree with a recall of 95.9%, a false positive-rate of 2.4%, a precision of 97.3%, and an accuracy of 96.8%. They use a behavior-based dataset which involves dynamic analysis of the malware. The main disadvantage is that it only contains 220 malware samples related to the Indonesian family. In [16] the author uses Chi-square, Information gain], Gain ratio, T-test and fisher score to select and rank important features of the malware and shows that the Fisher score can be used for feature selection. The benefit of this approach is that it helps in selecting the important features which can be used in malware detection but the author doesn't consider other statistical tests like F-score to rank the features. In [14] the author uses Windows API calls as its features with stacked autoencoders for malware detection and achieves an accuracy of 96.85%. This paper includes the dynamic property of the malware such as API calls which tells a lot about the malware's behavior, although various other dynamic properties are not considered during the training. The model is tested against 2/3/4/5 hidden layers and obtains the best accuracy with 3 hidden layers and 100 neurons at each hidden layer. In [17] the author uses the Convolution Neural Network approach to classify the malware into different families and achieves an accuracy of 98%. In this approach, the malware binary was converted into a grayscale image. Malware belonging to the same malware families will have the same visual representation. Therefore, an image classification approach is proposed. The [14] approach classifies only 25 families while there are various other types of files that are not considered; hence this approach isn't sufficient to detect the malware.

III. PROPOSED WORK

The overview of the malware detection process using ANN is shown in Fig 2. The file to be detected is given into the feature extraction and encoding process where the features are extracted and encoded between the value of 0 to 1. Then the feature vector of the file to be detected is generated from the features extracted and fed into the trained ANN model to make the prediction.

Retrieval Number: D8025049420/2020©BEIESP
DOI: 10.35940/ijeat.D8025.049420
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Then the prediction made by the ANN model tells us whether the file is malicious or not based on the training of the ANN model.

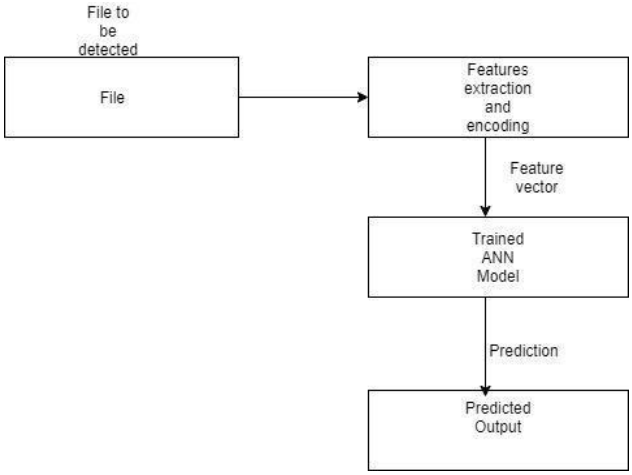

Fig 2. Malware detection process

The artificial neural network approach with ReLu as the activation function [7] is used and the model is trained with different amounts of hidden layers containing either 70 or 80 neurons in each layer with Adam optimization[19] technique. Adam optimization helps the model in optimizing the learning rate as the input of the model is sparse in some cases. This optimizer is easy to implement, is efficient, has low memory requirements, is invariant to the diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters, to detect the malware with

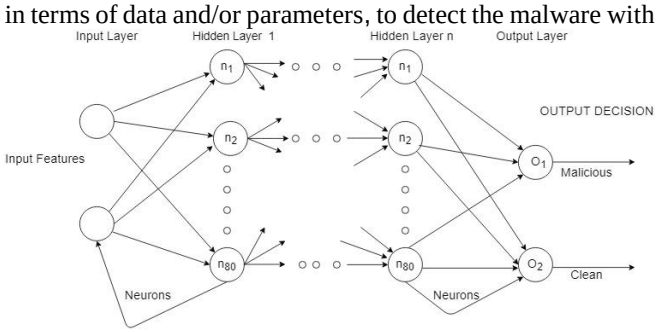

Fig 3. Proposed Model

higher accuracy. Fig 3 shows the proposed model varying from 2,3,7 hidden layers. The input features are extracted from the portable executable (PE) file. Portable Executable [4] is the file format for the Windows operating system for executables, .NET Object, COM files, etc. It is a data structure that is used by the Windows loader to load the PE into the memory and this data structure defines various properties of the file.

A PE file has different headers, each header has various properties like base address, address of the entry point, the size on disk, size in memory. Each of these properties has a certain value which is used in feature engineering. PE also has different sections [4] like code, data, etc. The code section contains the code of the PE while the data section consists of feature engineering. Each PE file has an IAT (Import Address Table) [10][11] which is in the. idata section which tells about all the APIs, system DLL it is importing. First, in the model 11 different properties are selected based on literature surveys. These properties are extracted from PE and each property is represented as 1 it is true and as 0 if it is false. Following properties are extracted from PE

- Debug section
- Export Symbol Table
- Resource section
- TLS [4]
- NX bit
- Import Symbol Table
- Relocations Entries
- Rich header
- Load Configuration
- Digital Signatures

PE is parsed and if any of the above properties is found to be in the PE then it is represented as 1 else it is represented as 0. Based on the literature survey various DLLs, API used by the malware are selected and for each API being used by the PE one column of the relative library incremented, then normalized by the total amount of API being imported. Next, the Shanon entropy of the various sections present in the PE file is included. Finally, the entry point of the PE is included in the feature set. After taking the input features from the input vector the ANN feeds the input to the input layer where the activation function is used to transform the sum of the weighted input in the node into the activation of the node. There are many activation functions that can be used in a neural network like tanh, sigmoid, ReLU, etc. The tanh activation function returns a value between -1 to 1 (i.e. -1 < output < 1) , the sigmoid activation function returns a value between 0 to 1 (i.e. 0 < output < 1) , and the ReLU activation function returns a value between 0 to infinity (i.e. 0 < output < +∞) . This paper proposes to use ReLU (Rectified Linear Unit) in the model as it is computationally cheaper compared to sigmoid and tanh. It requires a simple max function which is trivial when compared to tanh and sigmoid which require the use of exponential calculation. ReLU is a nonlinear function but behaves like a linear function if the input is more than zero and returns a zero output if the input is less than or equal to zero which makes the ReLU easier to optimize. Therefore, in this paper, ReLu is used and the output of it is fed to the next layer in the neural network. The last layer (output layer) of the ANN then finally gives the output as 0 (clean) or 1 (malicious) based on the input it receives from the previous layer.

IV. EXPERIMENTAL SETUP

The proposed malware detection system is developed using python, and the ANN model is implemented using Keras library. The data set used for training and testing are collected from various websites like Virusshare, GitHub repository, honeypot [6] Malware.lu, etc. on the cloud. This dataset consists of 200,000 samples which are in the form of Windows PE[4] file format. These samples are divided into two parts: one contains 100,000 malicious samples and the other contains 100,000 clean samples. 70% of the dataset is used for training, 15% samples for validation and the rest of the 15% of the samples are used for testing the accuracy of the model. Using these samples, the proposed model is trained and tested. The proposed ANN model is tested by varying the number of layers and numbers neurons in a layer. 100 neurons in a layer are used [14] for malware detection but in this paper, 70 and 80 neurons are used in a hidden layer. Initially, the number of hidden layers is kept at 2 and is gradually increased to 3 hidden layers and 7 hidden layers. The dropout value is set as 30%.

V. PERFORMANCE ANALYSIS

The proposed model is trained and tested by varying the parameters such as the number of neurons and the number of hidden layers. The variant models are referred to as given below

- ANN-80-2H: Artificial Neural Network with 80 neurons and 2 hidden layers
- ANN-80-3H: Artificial Neural Network with 80 neurons and 3 hidden layers
- ANN-80-7H: Artificial Neural Network with 80 neurons and 7 hidden layers
- ANN-70-2H: Artificial Neural Network with 70 neurons and 2 hidden layers
- ANN-70-3H: Artificial Neural Network with 70 neurons and 3 hidden layers
- ANN-70-7H: Artificial Neural Network with 70 neurons and 7 hidden layers

These variant models are evaluated by measuring the accuracy of prediction. The accuracy of prediction is measured for training, validation and testing data set. The accuracy of prediction is calculated as follows

Accuracy = Correct predictions / Total predictions

$$= (TP + TN) / (TP + TN + FP + FN) \quad (2)$$

where TP denotes True Positive, which means that observation is positive and it is predicted to be positive. FN denotes False Negative which means that observation is positive, but it is predicted negative. TN denotes True Negative which means observation is negative, and it is predicted to be negative. FP denotes False Positive which means that observation is negative, but it is predicted positive. Table 1 shows the accuracy of prediction for variant models using the training dataset, validation dataset, and testing dataset. It is observed from the result shown in Table 1 that there is a positive correlation between the number of hidden layers, neurons in the model with the accuracy of the model. Although all the variants of the ANN model had similar accuracy ANN-80-7N performed best in the tests with the test accuracy of 98.2%.

TABLE 1: Accuracy of prediction for variant models using training, validation and testing samples.

Model	TR-ACC	VAL-ACC	TE-ACC
ANN-70-2H	96.57%	96.35%	96.1%
ANN-70-3H	97.6%	96.4%	96.24%
ANN-70-7H	96.34%	96.21%	95.7%
ANN-80-2H	97.1%	96.5%	96%
ANN-80-3H	97.45%	97.3%	97.32%
ANN-80-7H	98.3%	98.04%	98.2%

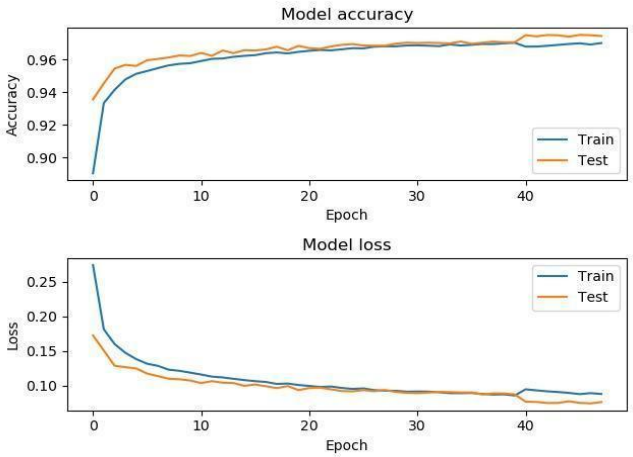

Fig 3. Model Accuracy vs Epoch

Fig 3 represents the accuracy rate over the test and train dataset graphically. It shows the training history of ANN-80-7N and it is evident that the model accuracy improved over time while decreasing the loss over epochs. From Fig 3, the loss value over the training data is 0.0820792 while the loss value over the test data is 0.0685853. Loss is the total sum of errors made for every input of training or validation sets. It is used to determine how well the model is doing for the test or validation set. In the proposed ANN model, SoftMax [21] is used as the activation function for the output layer, to determine the loss of the model.

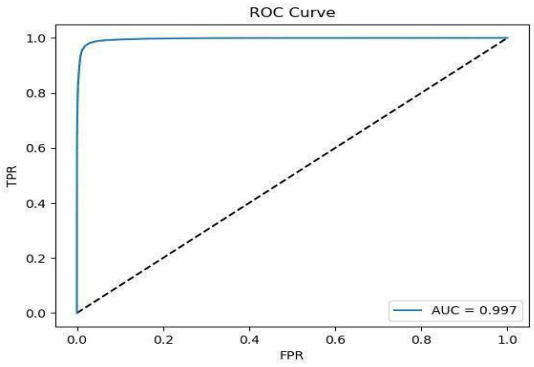

Fig 4. ROC Curve

Fig 4 shows the ROC (Receiver Output Characteristics) which determines with what accuracy the ANN-80N-7H model is able to distinguish between the malicious and clean sample. Sensitivity or True Positive Rate (TPR) is calculated as the number of correct positive predictions (TP) divided by the total number of positives (P). Sensitivity's value varies between 0 and 1, 1 being the best possible value for the model.

TPR (True positive rate or sensitivity)

$$= (TP) \div (TP + FP) \quad (3)$$

where TP represents true positive, FN represents False Negative. The false-positive rate (FPR) is calculated as the number of incorrect positive predictions divided by the total number of negatives. The value of the false positive rate ranges between 0 and 1, where 0 is considered the ideal outcome. It is calculated as.

FPR (False Positive Rate)

$$= (FN) \div (TN + FP) \quad (4)$$

where FN denotes False Negative, TN denotes True Negative and FP denotes False Positive.

Calculations of the TPR and FPR values in Fig 4 are:

$$\text{TPR} = (14700) \div (14700 + 164) = 0.98$$

$$\text{FPR} = (541) \div (541 + 14590) = 0.035$$

AUC (Area under the curve)-ROC curve is a performance measurement that tells how effective the model is. ROC is a probability curve and AUC tell about the degree or measure of separability. It tells how effective the model is in distinguishing between the clean and malicious. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the higher the AUC, the better the model is at distinguishing between malware and clean samples. AUC near to 1 is characteristic of a good model and Fig 4 shows that AUC (Area under the curve) is 0.997 which tells how effective the model is.

The given below shows the confusion matrix of test, training and validation sets.

Table 2: Test confusion matrix

n=29995	**Predicted: No**	**Predicted: Yes**
Actual: No	14700 (98.9%)	164 (1.1%)
Actual: Yes	541 (3.6%)	14590 (96.4%)

Table 3: Training confusion matrix

n=139979	**Predicted: No**	**Predicted: Yes**
Actual: No	69346 (99%)	674 (1%)
Actual: Yes	2271 (3.2%)	67688 (96.8%)

Table 4: Validation confusion matrix

n=29995	**Predicted: No**	**Predicted: Yes**
Actual: No	14879 (98.6%)	213 (1.4%)
Actual: Yes	554 (3.7%)	14349 (96.3%)

Initially, the number of hidden layers is kept at 2 and is gradually increased to 3 hidden layers and 7 hidden layers for each type i.e. 70 neuron variant and 80 neuron variants. **After increasing the hidden layers in the model, it is found that accuracy increases until the 7th layer after that the value of the accuracy starts dropping. Based on the tests and literature survey [1][14] it can be concluded that as the number of hidden layers is increased, the accuracy of the model increases until a certain number of hidden layers, after that it starts decreasing.**

VI. CONCLUSION

This paper explained the traditional malware detection techniques like signature based and behavior-based detection and their shortcomings. The paper also highlighted the conventional machine learning based detection methods. This paper presents and compares different variants of ANN (Artificial neural network) approach for the malware [6] detection as the traditional detection techniques used by the various anti-virus tend to fail. The Steps needed for implementing the ANN model was explained. From the performance analysis it was found that the ANN-80-7H variant be formed the best with an accuracy of 98.2% and AUC (Area under the curve) of 0.997 which is better when compared to the model proposed by Babak Bashari Rad et al.[22] in which the accuracy is 97.8% and a precision 97.6%. Furthermore, the model can be improved by including more features of the PE files in the dataset, dynamic features of the PE files. The framework is not yet fully capable of replacing the commercial antivirus but provides an ANN approach to malware detection research. New samples can be added to the dataset to improve the model for the detection of newer malware.

REFERENCES

1. Rathore, Hemant et al. “Malware Detection Using Machine Learning and Deep Learning.” Lecture Notes in Computer Science (2018): 402–411. Crossref. Web.
2. Y. Ye, D. Wang, T. Li, and D. Ye, “Imds: intelligent malware detection system,” in KDD, P. Berkhin, R. Caruana, and X. Wu, Eds. ACM, 2007, pp. 1043–1047.
3. Gavriluţ, Dragoş & Cimpoesu, Mihai & Anton, D. & Ciortuz, Liviu. (2009). Malware detection using machine learning. 4. 735 - 741. 10.1109/IMCSIT.2009.5352759
4. Wang, Tzu-Yen & Wu, Chin-Hsiung & Hsieh, Chu-Cheng. (2009). Detecting Unknown Malicious Executables Using Portable Executable Headers. NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC. 278-284. 10.1109/NCM.2009.385.
5. Guo, Wei & Wang, Tenghai & Wei, Jizeng. (2018). Malware Detection with Convolutional Neural Network Using Hardware Events. 10.1007/978-981-10-7844-6_11.
6. V. Sethia and A. Jeyasekar, "Malware Capturing and Analysis using Dionaea Honeypot," *2019 International Carnahan Conference on Security Technology (ICCST)*, CHENNAI, India, 2019, pp. 1-4
7. B. Ding, H. Qian and J. Zhou, "Activation functions and their characteristics in deep neural networks," *2018 Chinese Control And Decision Conference (CCDC)*, Shenyang, 2018, pp. 1836-1841.doi: 10.1109/CCDC.2018.8407425
8. Galal, Hisham. (2015). Behavior-based features model for malware detection. Journal of Computer Virology and Hacking Techniques. 10.1007/s11416-015-0244-0.
9. Steve Morgan” Cyber crime damages by $6 trillion by 2021”, October 16, 2017 [Online] Available: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/
10. Microsoft, Microsoft Portable Executable and Common Object File Format Specification, 2008
11. M. Pietrek, “Peering Inside the PE: A Tour of the Win32 Portable Executable File Format,” Microsoft Systems Journal, vol. 9, no. 3, 1994, pp. 15-34.
12. Ittiyavirah, Sibi & Jones, S. & Siddarth, P.. (2013). Analysis of different activation functions using Backpropagation Neural Networks. Journal of Theoretical and Applied Information Technology. 47. 1344-1348.
13. Hardy, William, Lingwei Chen, Shifu Hou, Yanfang Ye and Xin Li. “DL 4 MD : A Deep Learning Framework for Intelligent Malware Detection.” (2016).
14. Firdausi, Ivan & Lim, Charles & Erwin, Alva & Nugroho, Anto. (2010). Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection. Advances in Computing, Control, and Telecommunication Technologies, International Conference on. 201-203. 10.1109/ACT.2010.33
15. S. Shah, H. Jani, S. Shetty, and K. Bhowmick. Virus Detection using Artificial Neural Networks. In International Journal of Computer Applications, vol. 84(5), 2013
16. Nataraj, Lakshmanan & Karthikeyan, Shanmugavadivel & Jacob, Grégoire & Manjunath, B.. (2011). Malware Images: Visualization and Automatic Classification. 10.1145/2016904.2016908.

17. Köker, Raşit & Sari, Yavuz. (2003). Neural Network Based Automatic Threshold Selection for an Industrial Vision System.
18. Kingma, Diederik P. and Jimmy Ba. "Adam: A Method for Stochastic Optimization." CoRR abs/1412.6980 (2014): n. pag.
19. G. Hu and D. Venugopal, "A Malware Signature Extraction and Detection Method Applied to Mobile Networks," *2007 IEEE International Performance, Computing, and Communications Conference*, New Orleans, LA, 2007, pp. 19-26.
20. Nwankpa, Chigozie & Ijomah, Winifred & Gachagan, Anthony & Marshall, Stephen. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning.
21. Bashari Rad, Babak & Shahpasand, Maryam & Nejad, Mohammad. (2018). Malware classification and detection using artificial neural network. Journal of Engineering Science and Technology. 13. 14-23
22. Sherstinsky, Alex. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM
23. vol. 2, Aug. 1987, pp. 740–741 [Dig. 9th Annu. Conf. Magnetics Japan, 1982, p. 301].
24. M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.
25. (Basic Book/Monograph Online Sources) J. K. Author. (year, month, day). Title (edition) [Type of medium]. Volume(issue). Available: http://www.(URL)
26. J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com
27. (Journal Online Sources style) K. Author. (year, month). Title. Journal [Type of medium]. Volume(issue), paging if given. Available: http://www.(URL)

AUTHORS PROFILE

Vasu Sethia is currently pursuing B. Tech in computer science from the SRM Institute of Science and Technology Chennai, India. He is in fourth year and his area of research related to malware analysis and reverse engineering. He was the lead of sector433 labs in SRM Institute of Science and Technology Chennai, India and led a team of 5 members where the area of research was related to cybersecurity and its application. He has been selected thrice for Blackhat Arsenal to present his project. He is an active member of NULL Chennai and he loves to play CTF online.

Shivam Kataria is a competitive coder pursuing a B.Tech degree in computer science engineering from the SRM Institute of Science and Technology Chennai, India. He was born on 17th Jan 1998 in Delhi. His area of interests are software development, problem solving, competitive programming, web development and malware analysis. His project Malviz: - Malware Visualization on graph network was selected for BlackHat Asia arsenal 2020. He regularly participates in coding competitions on hackerearth and codechef. He has secured 27th rank April Circuits' 19 on HackerEarth Long programming Contest. He has also participate in many hackathons and secured 2nd position in IET Hackathon'19 ,SRM chapter.

Mr. A. Jeyasekar was born at Vickramasingapuram, Tirunelveli, Tamilnadu, India on June 11, 1969. He received his B.E. (Electronics and Communication Engineering) from The Indian Engineering College affiliated to Madurai Kamaraj University, Tamilnadu, India. He received his M.E. (Computer Science and Engineering) from Karunya Institute of Technology affiliated to Anna University, Tamilnadu, India in the year 2004. He got his Ph.D. in the area of congestion avoidance algorithm in heterogeneous wired-wireless network. He has filed a patent on preventing mechanisms for stegosploit attack and published/presented 36 papers in international/national journals and conferences. He was awarded as "Man Engineer" for scientific research contribution for Year 2015 by IET. His area of interest includes Networking, Network Security, Software Quality Management. He is presently working as Associate Professor in the department of Computer Science and Engineering, SRM Institute of Science and Technology (formerly called as SRM University), Tamilnadu, India