Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

DEEP LEARNING LSTM MODEL DYNAMIC ANALYSIS ON PE FILES FOR MALICIOUS CODE DETECTION

Mallishetty. Praveen Kuamr, Dept of Computer Science and Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.
P. Nagendra Babu, Dept of Computer Science and Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.
K.Venkata Nagendra, Dept of Computer Science and Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.
N.Kesav Kumar, Dept of Electronics and Communication Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.

ABSTRACT
Malicious software threats and their detection are becoming an important part of information security as Information and Communication Technology (ICT) applications are widely used in our daily lives. Identifying malware is one of the most difficult problems in the design and development of antimalware systems. It is important to develop dynamic analysis algorithms that enable rapid detection of polymorphic and metamorphic malware. Demonstrates techniques for analyzing trace data and detecting malicious code using Long Short-Term Memory (LSTM) (LSTM). Models were created for both malicious and good Portable Executable (PE) files' execution traces. Starting with the execution trace output gathered through dynamic analysis of the PE file, we built our first dataset. The suggested solution is more than 98% accurate, according to extensive testing with a data set that includes both benign and malicious programmers.
Keywords: LSTM, malware, deep learning, classification

1. INTRODUCTION
Malware is software created to perform harmful actions, including stealing confidential information, gaining root access, and incapacitating the target computer. In the meantime, a wide range of malware has appeared as a result of the Internet's and the software industry's fast expansion. Over the last three quarters, there have been more than 774 million malware samples, an increase of almost 34%. Malware (also known as malware) increases over time. Therefore, malware detection is an important and fascinating topic. How to recognize malware has been studied extensively. Because they have a limited ability to detect new threats, signature-based static anti-virus software is often used to identify malware. Many malware can easily avoid detection from signature-based security measures if it has been encrypted, obfuscated, or packaged to avoid detection. This detection method can be circumvented by zero-day malware. Real-time system scanning Unlike cloaking strategies, lurenjie17@mails.ucas.ac.c is a more effective malware detection tool. For dynamic behavior-based malware detection approaches, a protected and controlled environment, such as virtual machines, emulators, sandboxes, etc., is frequently needed [4] [5]. The following stage involves doing behavioural analysis utilising data acquired from interactions with the environment, like API calls and DLL calls. These techniques have been extensively studied, however they are ineffective when used to huge data sets [6]. To keep dynamic behavior-based malware detection systems from contaminating the operating environment, it requires a lot of time and work. Over the past few years, machine learning-based malware detection techniques have been developed. The first publication of the data mining-based malware detection technique was in ref [7]. It employs three different kinds of static features:
A text string, a byte sequence, and a PE header are used to spot malware. Kolter and Maloof [8] evaluated the effectiveness of naive Bayes, decision trees, and support vector machines for virus identification using n-grams as opposed to byte sequences. In recent years [9,10], malware has also been detected using artificial neural networks [9]. There are also new methods for detecting malware.

Malware can be detected by image processing in [11] and [12]. However, the previous attempt was successful enough in terms of malware detection. The machine learning classifier is trained by manually analyzing malicious code and comparing it with features extracted from the code itself. An innovative and efficient approach to determine if a Windows executable is malware has been proposed in this study to reduce engineering costs for artificial features. The assembly format files of the executables need to be recovered by splitting them first with IDA Pro. We need a method to take the opcode string out of each file in assembly format. Then, word embedding techniques [13] and long-term memory (LSTM) [14] are used to comprehend the feature vector representation of the opcode and automatically learn the opcode sequence patterns of the malware. After the second LSTM layer, we add an average aggregation layer to improve the immutability of the local feature representation. We ran a series of tests on a dataset consisting of 969 malicious files and 123 benign files to see if our strategy worked. MalwareXiv:
1906.04593v1 [cs.CR] 10 June 2019 Detection performance was evaluated during the experimental period and comprehensive performance comparison with other similar studies was conducted. The assessment results demonstrate that our suggested method can identify malware with an average AUC of 0.99 and classify malware with an average AUC of 0.987.

2. LITERATURE SURVEY

Unlike static analysis, dynamic analysis-based malware analysis techniques are more resistant to obfuscation. Dynamic analysis was used to classify API requests that took less than five minutes in [1]. The AUC, a measure of quality, was calculated using 170 samples and yielded a score of 0.96. Separately collected samples of benign and malicious software were used to build a response network using the API call feature set. It performs well compared to previous methods, but lacks research on execution speed, which is essential for real-time implementations. ESN and RNN tests were performed in [3] to learn the language of the malware. ESN performed better than RNN in the majority of studies. Tests were performed in [4] to determine when to stop running viruses on network traffic, as shown in [5]. Conventional procedures require 67% more time than this method. With long-chain API calls as a feature, the RNN and its long-term short-term memory (LSTM) and CNN versions were used for malware classification in [6]. The main problem with current methods is that they take a long time to test the behavior of the system during operation. It has been used in [7] to classify malware using a system call sequence in the form of a hybrid CNN and RNN. SVM and Hidden Markov Model were previously used to obtain these system calls, but dynamic analysis was used to obtain them and it was found to be more efficient (HMM). The biggest problem, however, is the lack of discussion about the relevance of runtime in real-time virus detection. Using RNN and two datasets, [13] proposes a technique. In addition, they tested the performance of several well-known classical machine learning classifiers. With a run time of 5 seconds, they claimed an accuracy of 94%. Analysis-based static, dynamic, and mixed malware detection methods have been the subject of several investigations. HMM has been used for both static and dynamic feature set analysis and to compare detection rates for a large number of malware types in [8]. Overall, they found that dynamic analysis had the best detection rate WindowsDynamic-Brain-Droid (WDBD) was the model we developed to compare and contrast several traditional machine learning algorithms (MLA) and deep learning architecture to determine which technique is best for Windows Malware Classification. The number of malware and benign patterns in our dataset varies with runtime, which is why we used two separate datasets.

3. PROPOSED WORK

The main goal of Long-Term Memory (LSTM), a specialised RNN architecture, is to resolve the leak slope problem or at the very least lessen the impact of the gradability problem. performance of computers is a leak slope. Nodes in the LSTM neural network have concealed state from the preceding phase, similar to RNN. The node, a typical LSTM unit, has a better structure than an RNN,

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

which is crucial for supplying long-term memory by lessening the impact of the leak gradient.
A standard LSTM unit generates an output value from an input value. The produced output value of the current cell and the prior cell's cell state value—which will be detailed in more detail in the coming paragraphs—are both used during this process. The following three functions can be carried out by an LSTM unit.
Erasing undesired information from the tile's present state by using the Forgotten Gate
Through the front door, add new details to the current state of the cell.
Output the condition of the current cell through the output port.

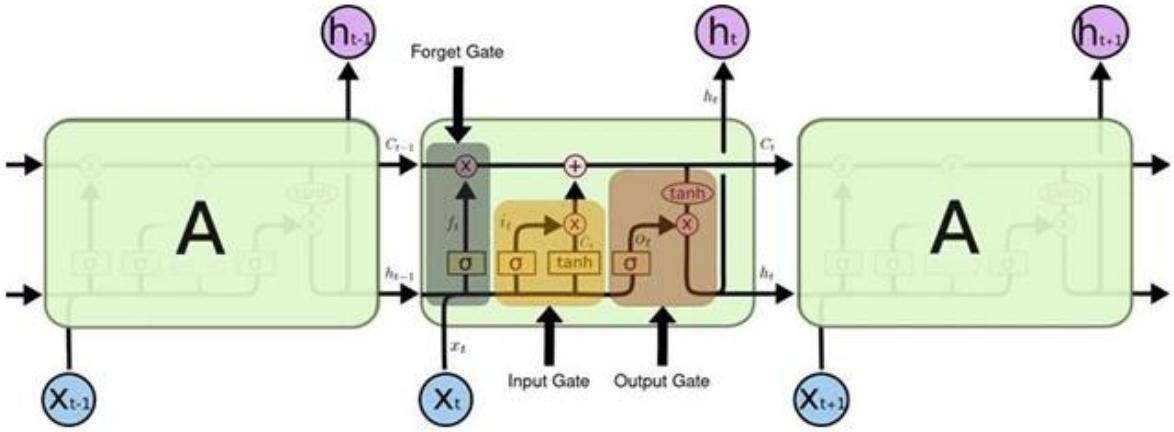

Fig-1:TheinteriordesignofacommonLSTMcell[24]

An typical LSTM unit's interior is straightforward and practical, as seen in Figure 1. By disregarding the input of this current cell (Xt) and its output (ht1) of the preceding cell (ht1), the sigmoid function may be utilised to create an output between 0 and 1 on the left side of a cell. The current cell state is updated and created by multiplying this value, ft, by the cell state that was previously displayed, Ct1. A value that travels across cells to transfer information between them is essentially what a cell state is. The forgetting gate is a component of the unit that uses a multiplier operation to determine which information will be forgotten and how much will be remembered in succeeding cells [26].

In the centre of the cell, there are two sigmoid functions, a tan h function, and their output is multiplied together. The output of the previous cell, ht1, as well as the input from the current cell, Xt, are both used as inputs for the sigmoid function in this. The output value of this sigmoid, in contrast to the sigmoid function used in the monitoring process, will be used to signify what new value should be added to the existing state of the cell.

An array of possible values is created by the tan h function, which may or may not be added to the cell's present state in the future. By dividing the output of the cell's sigmoid by the output of tan h C t, one may determine the values that should be added to the cell's present state. The output of the sigmoid it is multiplied by the output of the tan h C t to achieve this. The final current cell state is produced by updating the prior cell state Ct1, which was altered by the forget gate, with fresh data from the input via an add operation. The front gate is the name given to this portion of the LSTM unit as a result [26].

Algorithm 1: Algorithm for Mawere analysis
input : run Trace Pool = {f1, f2, ..., fN } where N = 290
output: accuracy Rate, loss
for f ∈ run Trace Pool do lines ← f .read Lines (); if f is malware then

An array of possible values is created by the tan h function, which may or may not be added to the cell's present state in the future. By dividing the output of the cell's sigmoid by the output of tan h C t, one may determine the values that should be added to the cell's present state. The output of the

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

sigmoid it is multiplied by the output of the tan h C t to achieve this. The final current cell state is produced by updating the prior cell state Ct1, which was altered by the forget gate, with fresh data from the input via an add operation. The front gate is the name given to this portion of the LSTM unit as a result [26].

In general, the current LSTM cell creates the output by updating the cell state of the previous cell while also receiving the output and cell state of the previous cell in addition to the current cell's input. The sequential internal nature of the LSTM architecture gives higher efficiency when handling data that comprises prolonged sequences of events than the basic RNN architecture.

4. RESULTS & DISCUSSION

The dataset (Drebin-215) also contains 215 functionalities from 15,036 app samples, of which 9476 were malware samples from the Drebin project and the remaining 5560 were safe samples [4]. The dataset (Drebin-215) also contains 215 data points from 15,036 app specimens, of which 9476 were deemed benign and the remaining 5560 were deemed malicious. The public can access the Drebin samples for free and they are frequently used by scientists. Two datasets, Drebin-215 and Malgenome-215, are available for download in the supplementary information.

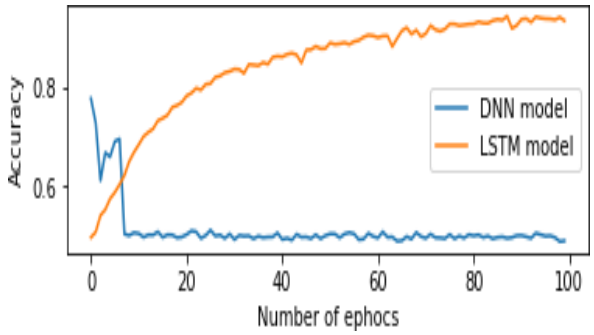

Fig-2:Accuracy

Classification Whenever we say "accuracy," we typically refer to the degree to which something isaccurate. The ratio of correct predictions to the total number of input samples is avaluable indicatorofaccuracyin aforecastingmodel.

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

Herefig-2representsthe precision of the distinction between benign and malicious samples. Additionally, the graph contrasts the proposed LSTM model with the existing ANN model. The ANN model is unable to accurately differentiate between dangerous and benign components. Because malicious code consists of a series of operations, ANN is unable to remember code sequences. However, because the LSTM model has a memory unit, it can provide improved accuracy as the number of epochs increases. While the number of ages is increased, ANN fails to offer improved accuracy at the same time.

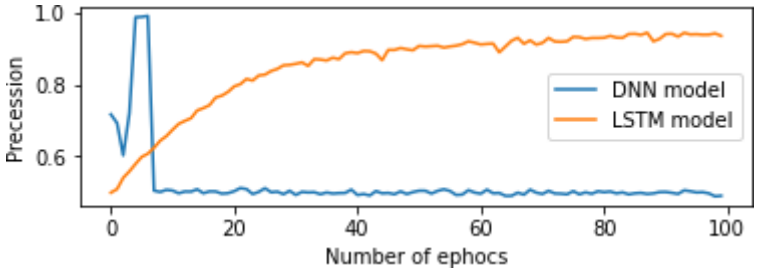

Fig-3:Precession

The ratio of true positives to all other true positives and false positives can be used to determine precision. Precision looks analyses the information to see how many false positives were mixed in.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

The model's accuracy is 100% if there are no false positives (FPs). The more FPs added to the mix, the more unsightly that precision will appear.

Precision is defined as TP/(TP+FP). Figure 3 depicts the classification of harmful samples and good samples. The graph also compares the proposed LSTM model to the current ANN model. The ANN method falls short of offering a more precise classification of damaging and beneficial cases. ANN is unable to recall harmful code sequences since it comprises of a number of operations. However, because the LSTM model has a memory component, it can provide better precession as the number of epochs rises. At the same time, as the number of epochs rises, ANN fails to provide.

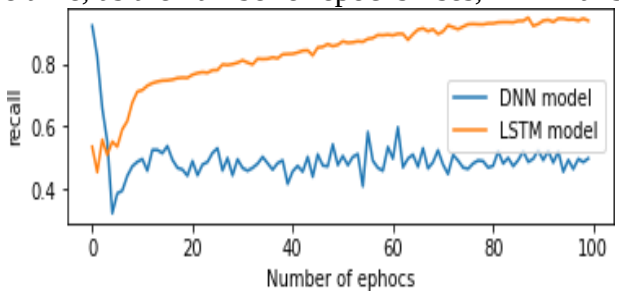

Fig-4:Recall

Itiscalculatedas the quantity of accurate optimistic findingsseparated by the total amountofappropriatesamples.

Precision=TP/(TP+ FN)

Figure 4 shows recall for the distinction between harmful and benign samples. Additionally, the graph contrasts the proposed LSTM model with the existing ANN model. The ANN model does not provide superior

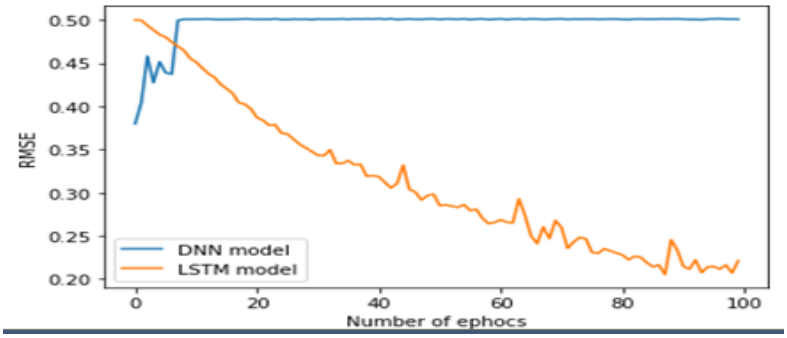

Fig-5: RMSE

Keeping in mind both bad and good instances. ANN is unable to recall malicious code sequences since it comprises of a number of operations. The LSTM model, however, incorporates a memory element that enhances recall as the number of epochs increases. ANN also fails to provide as the number of epochs increases.

The RMSE is the statistical term for the standard deviation of errors that occur when a forecast is made based on a dataset. The only difference between this and MSE is that when assessing the model's correctness, the foundation of the number is taken into account.

Here fig-5 represents the precession of harmful samples and benign samples according to the RMSE categorization. The suggested LSTM model and the current ANN model are contrasted in the graph. While the LSTM generates a low RMSE value, the ANN model produces an error value that ranges from 0.4 to 0.6. Because malicious code consists of a series of operations, ANN is unable to remember code sequences. However, the LSTM model has a memory unit that can perform better as the number of epochs rises. At the same time, as the number of epochs rises, ANN fails to provide.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

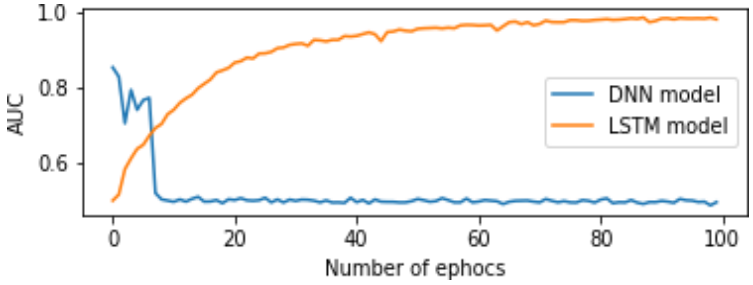

Fig-6:AUC

The Receiver Operating Characteristics (ROC) curve is summarised using the Area Under the Curve (AUC), which evaluates a classifier's aptitude for differentiating across classes. The better the user perceives the model's capacity to distinguish between positive and negative categories, the higher the AUC.

FIGURE 6: AUC for identifying malicious and benign samples is shown here. Furthermore, the proposed LSTM model and the current ANN model are contrasted in the graph. The ANN model falls short in providing improved AUC for malicious and benign cases. ANN is unable to remember code sequences because harmful code consists of a succession of activities. AUC can improve as the number of epochs rises.

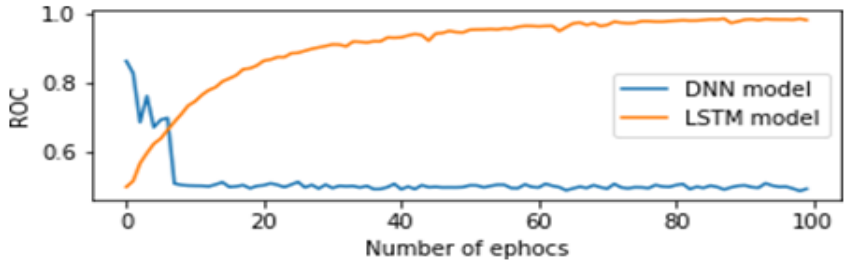

Fig-7:ROC

More items are categorised as positive when the classification threshold is lowered, which raises the proportion of True Positives and False Positives in the database.

Fig. 7 depicts the classification of dangerous and benign samples. The graph also compares the proposed LSTM model to the current ANN model. The ANN model is unable to offer a superior ROC of dangerous and benign situations. ANN is unable to recall harmful code sequences since it comprises of a number of operations. The memory component of the LSTM model, however, enhances ROC as the number of epochs increases. As the number of epochs rises, ANN also becomes ineffective at providing.

CONCLUSION

Since the introduction of information systems that had a significant impact on people's lives, malware detection techniques have been developing. The exponential growth of the information technology sector necessitates the development of malware detection techniques that are speedier and more precise. Additionally, sophisticated and fully automated malware detection systems are required due to malware writers' anti-detection strategies like obfuscation techniques. Given these specifications, artificial intelligence (AI)-based technologies make the best candidates for creating more dependable malware detection techniques. Machine learning (ML) classification techniques were frequently utilised in early AI-based research to distinguish between data produced by harmful and helpful software. But because feature extraction and selection require time and effort, ML classification algorithms do not offer fully automated methods.

REFERENCES

1. Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., & Yagi, T. (2016, June). Malware detection with adeepneuralnetworkusingprocessbehavior.InComputerSoftwareandApplicationsConference(COMPSAC),2016IEEE40thAnnual (Vol.2, pp.577-582). IEEE.
2. Huang, W., & Stokes, J. W. (2016, July). MtNet: a multi-task neural network for dynamic malwareclassification.InInternationalConferenceonDetectionofIntrusionsandMalware,andVulnerabilityAssessment(pp. 399-418).Springer, Cham.
3. Pascanu,R.,Stokes,J.W.,Sanossian,H.,Marinescu,M.,&Thomas,A.(2015,April).Malwareclassification with recurrent networks. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEEInternationalConference on(pp. 1916-1920).IEEE.
4. Shibahara, T., Yagi, T., Akiyama, M., Chiba, D., & Yada, T. (2016, December). Efficient dynamicmalware analysis based on network behavior using deep learning. In Global Communications Conference(GLOBECOM),2016IEEE(pp. 1-7). IEEE.
5. Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016, December). Deep learning for classificationof malware system call sequences. In Australasian Joint Conference on Artificial Intelligence (pp. 137-149).Springer,Cham.
6. Raff,E.,Zak,R.,Cox,R.,Sylvester,J.,Yacci,P.,Ward,R.,...&Nicholas,C.(2018).Aninvestigationof byte n-gram features for malware classification. Journal of Computer Virology and Hacking Techniques,14(1), 1-20.
7. Anderson, H. S., & Roth, P. (2018). EMBER: An Open Dataset for Training Static PE Malware MachineLearningModels. arXivpreprintarXiv:1804.04637.
8. Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T. H., & Stamp, M. (2017). A comparison of static,dynamic, and hybrid analysis for malware detection. Journal of Computer Virology and Hacking Techniques,13(1), 1-12.
9. Nataraj, L. (2015). A signal processing approach to malware analysis. University of California, SantaBarbara.
10. Nataraj, L., Kirat, D., Manjunath, B. S., & Vigna, G. (2013, December). Sarvam: Search and retrieval ofmalware.InProceedingsoftheAnnualComputerSecurityConference(ACSAC)WorkshoponNextGenerationMalwareAttacksandDefense (NGMAD).
11. Nataraj, L., Yegneswaran, V., Porras, P., & Zhang, J. (2011, October). A comparative assessment ofmalware classification using binary texture analysis and dynamic analysis. In Proceedings of the 4th ACMWorkshoponSecurityandArtificialIntelligence(pp.21-30).ACM.[12]Nataraj,L.,Jacob,G.,&Manjunath,B.S.(2010). Detectingpacked executablesbased onrawbinarydata. Technical report.
12. Farrokhmanesh, M., & Hamzeh, A. (2016, April). A novel method for malware detection using audiosignalprocessingtechniques.InArtificialIntelligenceandRobotics(IRANOPEN),2016(pp.85-91).IEEE.
13. Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011, July). Malware images: visualizationand automatic classification. In Proceedings of the 8th international symposium on visualization for cybersecurity(p. 4). ACM.
14. Nataraj, L., & Manjunath, B. S. (2016). SPAM: signal processing to analyze malware. arXiv preprintarXiv:1605.05280.
15. Kirat,D.,Nataraj,L., Vigna,G.,&Manjunath,B.S.(2013, December).Signal:A staticsignalprocessing-basedmalwaretriage.InProceedingsofthe29thAnnualComputerSecurityApplicationsConference(pp. 89-98). ACM.