Deep Learning Driven Cross platform Web Security Solutions

Chalasani Akhil Chowdary
Department of Computer Science, VIT,Vellore

Ambati Abhiram
Department of Computer Science, VIT,Vellore

Harsha Konakalla
Department of Computer Science, VIT,Vellore

Abstract: -

Over the years, the Internet has overwhelmed humankind and increased the dependence of human civilization on it. With the rising number of users, the Internet has become the primary source of security threats to computing systems. These computing systems, viz., computers, mobiles, and now IoT devices, have proliferated in the past decade. Amongst these, mobiles have become nearly ubiquitous with human existence. With the number of people browsing the Internet increasing exponentially, web-based attacks have become the most preferred means of attack. Security firms are now trying to battle these web-based attacks. However, these attacks are evolving rapidly, making it difficult for previous generations of security solutions to keep pace with them. Developed suitable deep learning models to predict malicious webpages using both structured and raw data as input. It has used both supervised and unsupervised approaches for producing these deep learning models. Using state-of-the-art techniques, these Deep learning models have surpassed previous performance metrics.

Keywords: - Deep Learning, Web Page, Web Security, Learning Model

DOI Number: 10.14704/nq.2022.20.13.NQ88314 Neuro Quantology 2022; 20(13):2535-2540

I. INTRODUCTION

The number of users browsing the Internet using web- based applications has grown exponentially over the last few years [1]. Amongst the popular web applications are the browsers like Microsoft's Internet Explorer and Edge [2], Mozilla Firefox [3], Google Chrome [4], etc. Also, with the rising number of mobile users, mobile app users have increased. With the increasing popularity of mobile apps, 'Hybrid mobile apps', which use web-application- based protocols, are getting popular. In fact, most popular apps on mobiles like Facebook, Twitter, Instagram etc., use the hybrid app technology. With the rising number of these web platforms, the web is the preferred route for infecting connected devices. With billions of websites active on the Internet, hundreds of them added each minute, and most of them updating their content frequently, any web security expert seems to be walking in a labyrinth. Most of these web-based attacks are 'Drive-by-download' attacks [5], which inject malicious JavaScript from the server hosting the malicious web applications to the browser or the hybrid app. As per the latest Internet security reports, such drive-by-download attacks have increased manifold over the past years [6].

These attack vectors have also become smarter, making it difficult for older anti-malware technology to detect them. The older anti-malware techniques like static and dynamic heuristics [7] fail to detect most polymorphic and metamorphic malwares [8] (while static heuristics involve decompiling a program and testing its source code, in dynamic heuristics program is run in a controlled virtual environment to see the changes being made at runtime). Further, with automated tools being used to generate new ever-evolving malwares, this task is becoming more and more daunting [9]. The paper attempts to overcome such limitations using AI for web malware detection.

Mobile is the most ubiquitous gadget on Earth today. The most common mobile platforms are Android and iOS, constituting 99.19% of the mobile ecosystem (72.72% Android and 26.47% iOS) [10]. The Android platform has emerged as the most popular computing platform that has

more than three billion devices active across the globe [11].

These devices include not only mobiles and tablets, but even Android auto modules in cars, various Android versions running on televisions, watches, and a host of other smart and IoT devices. What makes things more challenging and interesting for the Android developers and security experts is that various versions of the Android Operating System, from Android 2.3.3 (Ginger Bread) to Android 11.0, coexist in this ecosystem [12]. The thesis has discussed threats that emanate from hybrid Android apps. These apps use Web Viewcomponent for handling web content within Android apps [19]. Web View allows HTML and JavaScript to run and render webpages inside the apps, thereby allowing them to download content from web servers on the Internet. It is used by several popular apps, like Facebook, Twitter, Instagram, Gmail, Amazon, etc. [13].

II. RELATED WORK

Malicious web page detection approaches have evolved from static heuristics, dynamic honey client based detection to ML. Recently, with rapid advances in deep learning, and it is being explored for solving web security tasks. Earlier works using static heuristics include, wherein they used signature-based detection

techniques. Work utilizing dynamic heuristics includehigh interaction honey clients [14].

These approaches had limited capability of detecting new patterns. Conventional ML approaches for malicious webpage detection have used classifiers like SVM, Random Forest, Decision Tree, etc. Eshete [15], Singh [2], have used such techniques. However, their classification results could not surpass 99% accuracy and suffered from high false negatives or false positives [4, 6].

While attempts have been made to use deep learning for malicious web page detection, research outcomes are confined mainly to restricted domains of Google labs [6] or cybersecurity and anti-virus firms with limited information shared in public. Apart from these organizational efforts, few research papers, as discussed below, have been published in this field, but they have been inadequate to address the problem statementholistically. Shrivastava et al. have used a deep learning framework for webpage classification [12].

However, the framework was complex and could not achieve satisfactory accuracy while keeping false positives and false negatives low. Compared to the model proposed in this chapter, their framework underperforms in all metrics, including time. Fang et al. proposed a deep learning based solution to detect cross side scripting (XSS) [12]; however, their solution remains confined to XSS attacks. Vinaya Kumar et al. have evaluated various LSTM, CNN, and RNN layers for feature extraction to classify malicious URLs [13]. Although, their work explored practical feature extraction techniques for such tasks, they failed to propose a suitable end-to-end solution for webpage classification. Wang et al. have proposed a LSTM bidirectional algorithm based on CNN and RNN for expressing the similarity of web content with a malicious page [14]. However, the model underperforms on precision and recall metrics. Keeping related work in mind, the work presented in this chapter attempts to overcome existing limitations and gaps.

III. DEEP LEARNING

Deep learning is a subfield of ML that has gained prominence in the last few years. It uses layered neurons to learn complex non-linear patterns in the data. It can be used in both supervised and unsupervised settings. Further it can handle both structured and unstructured data [13]. Deep Neural Networks (DNN) carry out hierarchical learning, with lower layers learning low- level features and higher layers progressively learning high-level features from them [12]. According to the Universal Approximation Theorem, feed forward deep learning models can represent a nonlinear relationship in dataset better than shallow neural networks and other ML classification algorithms [13]. Typical deep learning techniques include DNN, Deep Belief Networks (DBN), Convolution Neural Networks (CNN), Auto Encoders (AE), Recurrent Neural Networks (RNN), etc.

IV. RESEARCH METHODOLOGY

The deep learning model's design for the

detection of malicious webpages in this research is shown below in Fig. 1.

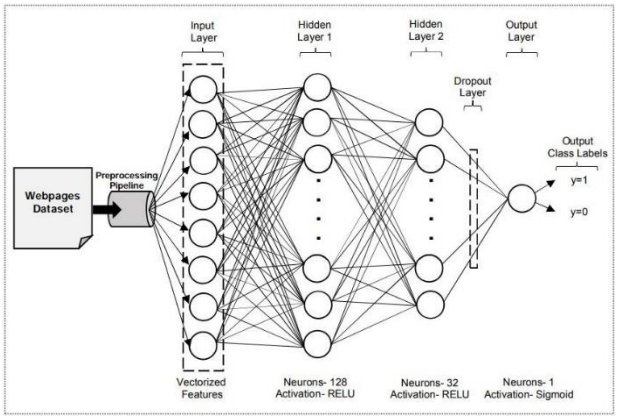

Fig. 1: DNN Model for Malicious Webpage Classification

The dataset was preprocessed and fed to the Input Layer. The Input Layer carried out vectorization, as shown in Table 1.

Table 1: Feature (Input) layer- DNN with Structured Data

#	Features Name	Transformation Carried Out
F1, F2, F8, F9	url_vect, url_len, js_len, js_obf_len	Normalized and fed as a numerical column.
F4, F5	geo_loc, tld	Converted to Hashed Categorical Columns, with bucket size equal to the number of unique values in each.
F6, F7	who_is, https	One Hot coded and fed as Categorical Column.
F11	label	Converted into a single class label column with binary value 0/1.

The vectorized and normalized values given above are condensed into a dense feature layer and fed to the next layer. The next layer is a hidden layer comprising 128 fully connected neurons. The output of this layer is fed to the third layer, which too is a hidden layer of 32 fully connected neurons. RELU (Rectified Linear Unit), which is one of the activation functions used in deep learning, was used for both the hidden layers. The choice of RELU for hidden layers was based on its faster convergence during training. The third layer feeds its output to a fully connected single neuron layer with a Sigmoid activation function.

The sigmoid function is another popular activation function used for binary classification problems that gives a binary 0/1 output for each of the two class labels defined in the F11 feature. Between the third and output

layer, a dropout layer was introduced with a dropout rate set at 10%. Dropout is a technique of randomly excluding few nodes from update cycles. 10% dropout from a layer of 32 nodes means that three nodes are dropped in each update cycle. The use of this technique gave a regularized model and overcame overfitting. The summary of tunable parameters in the generated DNN model is given in Table 2.

Table 2: Layer-wise Tunable Parameters (DNN withStructured Data)

Layer#	Layer (Type)	Output Shape	Param#
1	Input	Multiple	-
2	Dense (Hidden 1)	128	184192 (1438 x 128 Weights + 128 bias)
3	Dense (Hidden 2)	32	4128 (128 x 32 Weights + 32 bias)
-	Dropout	-	-
4	Output	1	33 (32 Weights + 1 bias)
	Total Params: 188,353 Trainable Params: 188,353 Non-trainable Params: 0		

This DNN model is a feed forward network that is trained using 'Gradient Descent'. Gradient Descent is an optimization algorithm that minimizes the cost/loss function by moving in the direction of steepest descent, thereby finding the minima of cost/loss function. 'Binary Cross Entropy' has been used as the loss function in this work. In this model, an extension of 'Gradient Descent' algorithm known as 'Adam Optimization' is used due to its better performance. Adam is a variant of the Stochastic Gradient Descent algorithm which uses an adaptive learning rate. Its name is derived from Adaptive moment estimation. The Gradient Descent algorithm works in two steps - the forward and backward pass. In the forward pass, it computes errors using current parameters. In backward pass, it computes gradients using partial derivatives and amends parameters accordingly.

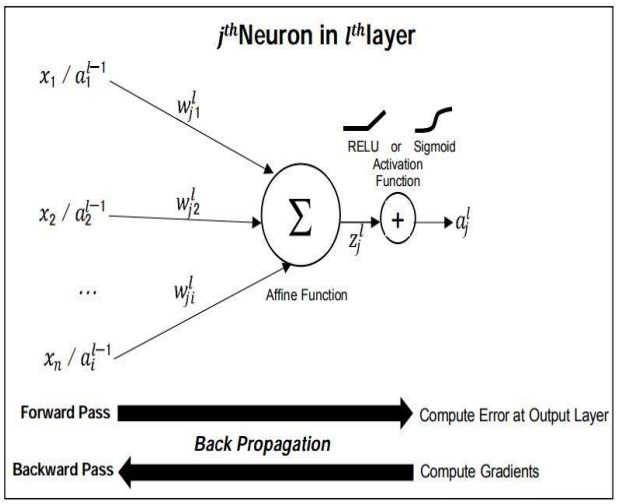

Fig. 2: A Single Computation Unit in DNN

V. SIMULATION RESULT

The DNN model proposed in the last two subsections was implemented using TensorFlow and Keras libraries. TensorFlow is an open-source ML platform in Python that was released by Google [14]. Keras [15] is a deep learning library in Python that can run on top of TensorFlow. Little functionality that was not part of the standard library was coded in Python using NumPy library. NumPy is a Python library that provides functions for vector calculus. For generating results and analysis graphs, TensorBoard and Seaborn libraries are used. TensorBoard library provides storage, retrieval, visualization of machine learning results produced using TensorFlow. Seaborn is a Python data visualization library based on Matplotlib. The code is written to run on CPUs. However, with minor modification, the code can run on GPUs and thereby further improve its time performance. The code is published online on Kaggle

[12] to support further research.

The model proposed in the previous section was trained with 1.0 million samples. For validation, a separate dataset of 0.2 million samples and for testing a different test dataset of 0.35 million samples were used.

The DNN model was trained over 40 Epochs with Early Stopping (with patience set to 20). The accuracy in each epoch during training and validation is plotted below in Fig. 3. The dark red line represents training accuracy, while the light red line depicts validation accuracy. While the training was set to 40 epochs, it is seen that it was stopped at 25 by the early stopping algorithm when the accuracy stabilized.

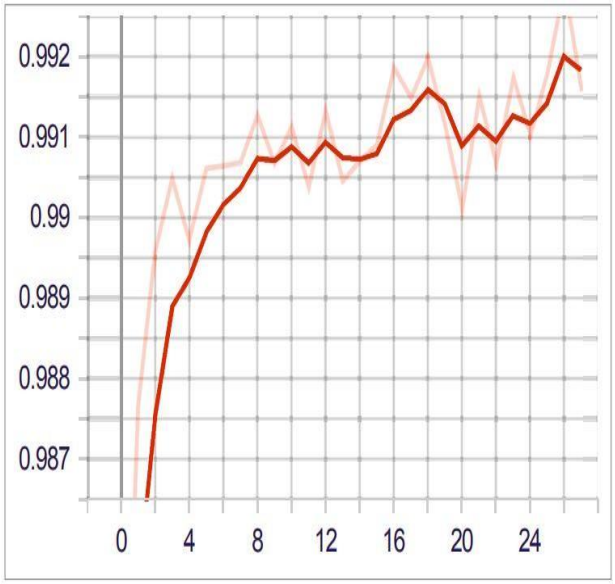

Fig. 3: Training and Validation Accuracy

Similarly, the binary cross entropy loss during the validation and training is shown below in Fig. 3.

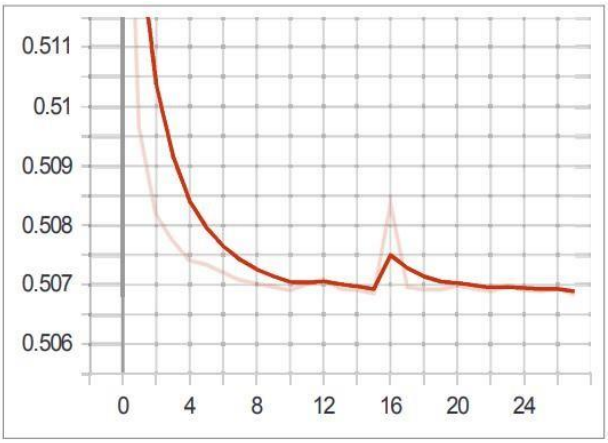

Fig. 4: Training and Validation Loss

It can be seen from Fig. 3 and 4 that the training accuracy had lagged behind validation accuracy, and validation loss has lagged behind training loss. This behavior is attributable to the 10% dropout that was implemented during training for regularization (10% dropout will randomly switch off 1/10th neurons in layer 3, thereby reducing training performance). Since dropout is

not used during validation and testing, training performance always lags validation and test performancein such regularized models.

In the previous section, we had discussed the initial bias used in this model. The modified initial bias resulted in faster convergence, as can be seen in Fig. 5. The graph shows that the loss dropped rapidly within few epochs with the use of initial bias.

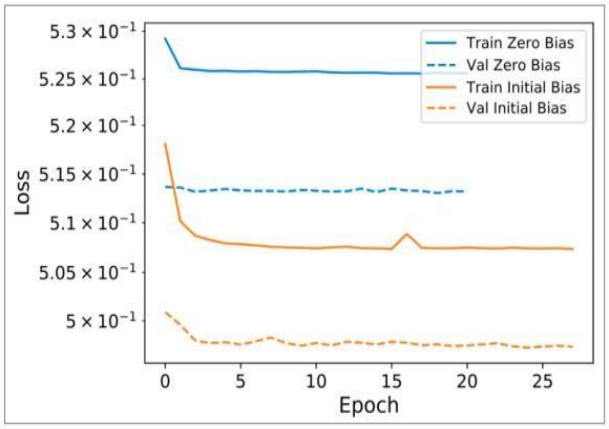

Fig. 5: Impact of Initial Bias

VI. CONCLUSION

This work provides a functional interdisciplinary approach to use deep learning in the field of web security. While other ML techniques have been used to detect malicious webpages, the use of deep learning in this field has been largely unexplored. It was seen that deep learning performs better than earlier ML models in the detection of malicious webpages. The deep learning model has outperformed previous models not only in accuracy, precision, and recall, but also in test response time. The performance of this model makes it suitable for real-time web security solutions on the Internet.

Lastly, we used structured data for deep learning. It would be interesting to know how the accuracy and response time changes with the use of unstructured data as input to the deep learning model, for example, feeding the web content directly to the DNN.

REFERENCES

[1] Prasad, Ramjee, and Vandana Rohokale. "Cyber Threats and Attack Overview." In Cyber Security: The Lifeline of Information and Communication Technology, pp. 15-31. Springer, Cham, 2020.

[2] Vyawahare, M., & Chatterjee, M. (2020). Survey on Detection and Prediction Techniques of Drive-by Download Attack in OSN. In Advanced Computing Technologies and Applications (pp. 453-463). Springer, Singapore.

[3] Sihwail, Rami, Khairuddin Omar, and Khairul Akram Zainol Ariffin. "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis." International Journal on Advanced Science, Engineering and Information Technology 8, no. 4-2 (2018): 1662.

[4] Tahir, Rabia. "A study on malware and malware detection techniques." International Journal of Education and Management Engineering 8, no. 2 (2018):20.

[5] Mirea, Mihnea, Victoria Wang, and Jeyong Jung. "The not so dark side of the darknet: a qualitative study." Security Journal 32, no. 2 (2019): 102-118.

[6] B. Altay, T. Dokeroglu, and A. Cosar, "Context- sensitive and keyword densitybased supervised machine learning techniques for malicious webpage detection," Soft Computing, pp. 1 15, 2018.

[7] McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In Artificial Intelligence and Statistics, pp. 1273-1282. PMLR, 2017.

[8] Li, Wenqi, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust et al. "Privacy-preserving federated brain tumor segmentation." In International Workshop on Machine Learning in Medical Imaging, pp. 133-141. Springer, Cham, 2019.

[9] Rieke, Nicola, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, et al. "The future of digital health with federated learning." NPJ digital medicine 3, no. 1 (2020): 1-7.

[10] Long, Guodong, Yue Tan, Jing Jiang, and Chengqi Zhang. "Federated Learning for Open Banking." In Federated Learning, pp.

240-254. Springer, Cham, 2020.

[11] B Wainakh, Aidmar, Alejandro Sanchez Guinea, Tim Grube, and Max Mühlhäuser. "Enhancing privacy via hierarchical federated learning." In 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 344-347. IEEE, 2020.

[12] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, "Malicious web content detection by machine learning," Expert Systems with Applications, vol. 37, no. 1, pp. 55 60, Jan. 2010.

[13] Pham, Kien, Aècio Santos, and Juliana Freire. "Understanding Website Behavior based on User Agent." Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016.

[14] Singh, A. K., and Navneet Goyal. "Malcrawler: A crawler for seeking and crawling malicious websites." In International Conference on Distributed Computing and Internet Technology, pp. 210-223. Springer, 2017.

[15] A. K. Singh and N. Goyal, "A Comparison of Machine Learning Attributes for Detecting Malicious Websites," 11th International Conference on Communication Systems & Networks (COMSNETS 2019), Bengaluru, India, 2019, pp. 352-358.