Using Deep Learning to Detect Malicious URLs

Yuchen Liang
Shady Side Academy
Pittsburgh, PA, United States
20liangy@shadysideacademy.org

Xiaodan Yan
Beijing University of Posts and Telecommunications
Beijing, China
xdyan@bupt.edu.cn

***Abstract*—With the development of energy internet, cyber-attacks pose a great threat to the distribution of electricity. Specifically, malicious domains can cause tremendous damages to network security. Many researchers have tried to tackle the task of detecting DGA-generated domains with different approaches. In this paper, an algorithm based on the Deep Bidirectional LSTM model is constructed to address this problem. Before the final experiment with the deep learning model, this paper also presents traditional machine learning methods that classify malicious domains based on lexical features for comparison. The result shows that the DBLSTM classifier can achieve a 98.6% accuracy, performing much better than conventional machine learning approaches.**

***Index Terms*—energy internet, network security, DGA detection, DBLSTM, Deep Learning.**

I. Introduction

In the past a few decades, the Internet was developing at an unprecedented speed. Recently, progress was not only made in the development of telecommunication network but also energy network which aims to efficiently supply electricity to anyone anywhere. Energy Internet is a new power generation developing a vision of evolution of smart grids into the Internet [K. Wang]. The objective for energy internet is to build a smart and connected energy system that enable the integration of energy flow, information flow, and business flow [1]. In 2016, Weifei Zhongs research group proposed the SDEI architecture which put the control, data, and energy planes into separate structures [2]. This architecture enables the programmability of energy flow and P2P energy delivery in the future [2]. It is also interesting to see what roles the prevailing deep learning algorithms play in the development of energy internet. In 2019, Haochen Hua and his research group adopted a model-free deep reinforcement learning algorithm to obtain the desired control scheme for the energy internet [3].

Since an energy internet system might have extensively distributed consumers and devices, it is crucial to gurantee the security of the system [4]. Without network security, it is hard for the energy internet technology to advance further. In 2016, Ukraine experienced a power cut that resulted in a loss of about one-fifth of the energy consumption that night [5]. The cyber attackers intruded three power centers and left more than 230,000 residents in the dark [5]. Moreover, the cyberattack caused tremendous damage to the energy network that the control centers were still not fully operational two months after the cyberattack [5]. If the cyber attackers already know how to attack the power grid, the worlds electricity system is at stake because most major economies rely on the energy internet. Therefore, it is necessary to come up with an effective approach to address the potential cybersecurity threat to the energy network.

II. Related Works

As network security has become an issue, many companies and individuals implemented firewalls control the incoming and outgoing network traffic. Although firewalls are effective to block unauthorized access, they enforce restrictive policies that limit users legitimate operation. Application-based firewalls can bring inconvenience to the usage of a computer since they need to run constantly and take up processor power and RAM memory which could be used for other application to perform their function. Therefore, network security experts have developed different approaches to counter intrusions. Many researchers approach this issue through blacklisting, a method providing a list of URLs that contain malware or phishing contents. In particular, Zhang, Porras, and Ullrich introduce a relevance ranking algorithm that produces individualized blacklist, which increases the hits on the potential intrusion [6]. However, blacklisting is vulnerable to distributed denial of service attacks since it cannot handle the traffic generated by the cyber-attackers using thousands of fake connections. In other words, clients might access to the phishing site before its domain is manually updated into the database. Other efforts have been made to solve security problems. Karasaridis, Rexroad, and Hoeflin created a method to detect botnets by analyzing traffic data through scalable, non-intrusive algorithms and obtained less than 2% false positive rates [7]. Besides, Antonakakis et al. developed a new technique that examines the Non-Existent Domain responses to detect DGA-generated domains characterized them based on the fact that domains with the same DGA algorithm would have similar NXDomain traffic. They discovered twelve DGA families during their evaluation of DNS traffic, in which six out of twelve DGAs are not revealed before [8]. The rocketing of utilizing machine learning has made it one of the most influential technique to solve security issues. Even back in 2006 Livadas et al. have applied conventional machine learning approaches to differentiate botnet Internet Relay Chat(IRC) traffic from real IRC traffic. Their results suggested that the naive Bayes classifiers work well with real life flows, but not with botnet testbed traffic [9]. In 2011, Bilge, Kirda, Kruegel, and Balduzzi used J48 deci-

978-1-7281-1493-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICEI.2019.00092

sion trees to analyze passive DNS, categorizing the different properties of DNS and the different ways of the query [10]. Their system EXPOSURE was tested in real life for two weeks and proved that it has the ability to recognize unknown malicious domains [10]. Many researchers have used machine learning to explore the relationship between the malware and their network traffic features. In 2011, Saad et al. designed a method based on machine learning algorithms to detect P2P bots before the attacks [11]. In their research, they adopted five different machine learning models to detect attacks only by analyzing network traffic behaviors [11]. The result shows that a 90% accuracy can be achieved by using SVM [11]. In 2013, Feizollah et al. attempted to examine the most effective classifiers among Naive Bayes, k-nearest neighbor, decision tree, multi-layer perceptron, and support vector machine to detect mobile botnet malware based on network traffic features [12]. A false positive of 0.06% has proved k-nearest neighbor to be the optimal classifier in their experiment [12]. In 2014 Beigi et al. also reexamined the productiveness of using flow-based features [13]. Other seized to follow the path of using machine learning to classify domains based on the features of the URL, which largely reduce the cost of the detection process. In 2009, Ma et al. used machine learning models such as Naive Bayes, SVM, and Logistic Regression to classify suspicious domains and attain 95-99% efficiency [14]. Likewise, Vanhoenshoven, Napoles, Falcon, Vanhoof, and Koppen in 2016 treated the detection of malicious URLs as a binary classification problem by employing many other machine learning techniques such as Multi-Layer Perceptron, Decision Trees, Random Forest, and k-Nearest Neighbors [15]. Unlike the approaches in this paper which only uses lexical features for basic machine learning models, both research groups, however, take other features such as blacklist and WHOIS info into account. Conventional machine learning models do improve their performance progressively, but they still need human intervention. For instance, if a machine learning algorithm returns an inaccurate prediction due to the adding of a bad feature, an engineer needs to get involved to make adjustments. However, with a deep learning model, the algorithm itself can determine the accuracy of prediction and make self-modifications to the neural network. The recent study of network security has started to deploy this approach on a large scale. In 2013, Dahl et al. improved the performance of their neural network on malware classification by using random projection to reduce the dimension of the original input so that the number of possible features would not be too large to train, which helped them to achieve a 0.49% error rate for a single neural network and 0.42% for a group of networks [16]. In 2014, Yuan et al. used a deep learning method to detect malware on the Android platform [17]. Their deep learning model was proved to perform better than other models and attained a 96% accuracy [17]. Two years later the same research group proved that deep learning is suitable for classifying Android malware with a large training set by implementing an online malware detector [18]. In 2015, David and Netanyahu proposed a new approach to identify the malware signature generation and classification automatically by using a deep belief network [19]. By deploying DBN, this research group presented that the features generated by this method also apply to new variants of malware, achieving a 98.6% accuracy for the classification [19]. In the last a few years, a lot of success has been achieved by the recurrent neural network due to its ability to connect between related information and the prediction. In 2016, Kolosnjaji et al. built a neural network based on CNN and RNN layers to classify malware with the best features. Compared to formerly using methods, their combined neural network model achieved an 85.6% accuracy [20]. In the same year, Chengs research group used a multi-scale LSTM model to detect anomalous Border Gateway Protocol (BGP) and achieved a 99.5% accuracy with optimal time scale 8 [21]. In 2017, Athiwaratkun and Stokes presented a novel approach that employs Echo state networks and recurrent neural networks to learn the behavior of the malware through analyzing the time domain features, improving the true positive rate to 98.3% compared to the former trigram of events model [22]. These works based on deep learning models gave inspiration to this paper that a method with DBLSTM model can be applied to detect malicious URLs.

III. Core Method

Researchers has come up with conventional machine learning methods to identify malicious domain. However, with those methods, the classifier would use features subject to human influence. To keep the objectivity of the features, this research adopted a deep learning method and let the computer generate features itself. Since a domain can be perceived as a time series, this research made use of Long Short-Term Memory (LSTM), a specific kind of recurrent neural network for modeling. Although LSTM allows information from the past to persist, it fails to preserve or utilize information from the future. Hence, the study decided to employ a Deep Bidirectional Long Short-Term Memory (DBLSTM) classifier to gather context from both the past and the future. Moreover, a three-layer structure was designed for this research to capture abstract features that are difficult to detect with a single layer of DBLSTM. The structure of the classifier is illustrated in Fig. 1.

Basically, every character in a domain is an input feature in a time sequence. These features are then processed separately in a forward LSTM-cell sequence and a backward LSTM-cell sequence. The output of the forward LSTM-cell sequence is calculated using inputs in the positive direction while the output of the backward LSTM-cell sequence is calculated using the inputs in the reversed direction. The two outputs are then concatenated and placed in a SoftMax function to normalize the values into a probability distribution, producing the final output.

In each LSTM cell that form the DBLSTM layers, the gates, each composed of a sigmoid neural net layer and a pointwise multiplication operation, are set to add or remove information from the cell state. In each module, there are four interacting

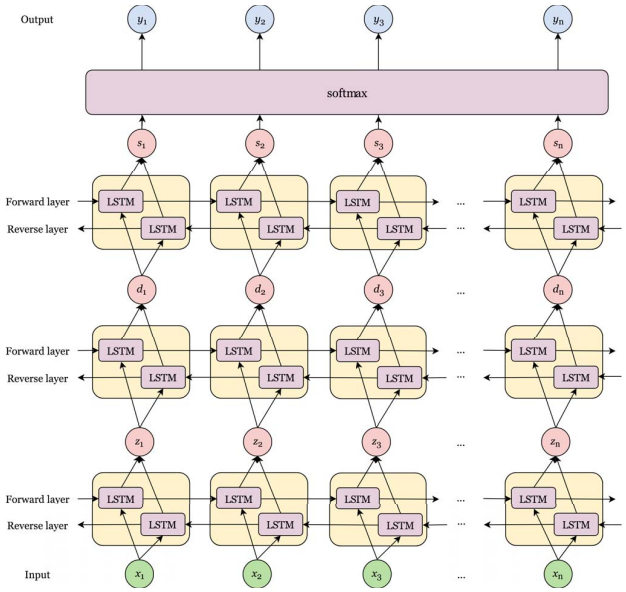

Fig. 1. Illustration of the DBLSTM Classifier

neural network layers [23]. The forget gate layer is a sigmoid layer that decides the information that would be thrown away from the cell state by examining the output h_{t-1} from the previous module and new input x_t [23]. This function returns a value between 0 and 1, labeling the possibility of keeping the data (0 suggests discarding it, while 1 suggests keeping it completely). The function is defined below:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

where W_f represents a weight vector and b_f represents a scalar bias.

Then, the input gate layer i_t determines the values to update, and a $tanh$ layer creates a vector of potential values $\bar{C}_t$ add to the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\bar{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

Next, the product of f_t and C_{t-1} and the product of i_t and $\bar{C}_t$ are added to output the new cell state.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \bar{C}_t \tag{4}$$

Finally, the output gate layer selects the parts of the cell state that are going to the output [23]. The output of this sigmoid gate is eventually multiplied to value returned by the tanh function that the cell state goes through to return the final output of the module. Fig. 2 illustrates the structure of an LSTM cell.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \cdot tanh(C_t) \tag{6}$$

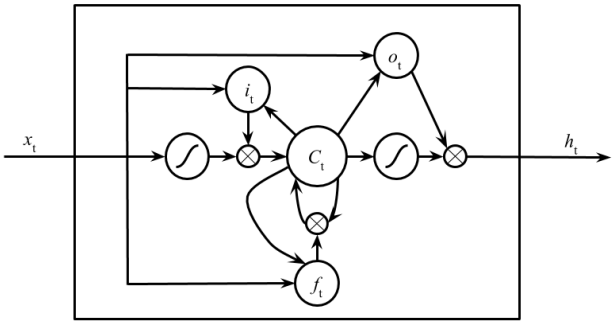

Fig. 2. Illustration of an LSTM Cell

IV. Experiment

A. Data

This section discusses the open datasets used in the experiment from two sources: 360 NetLab and Alexa.

1) DGA URLs: 1,145,093 malicious domains are collected from 360 NetLab, which can be characterized into 39 DGA Families [24]. Some of the DGA Families and domain examples are illustrated in Table I [24].

2) Normal URLs: top 1 million domains are collected from the Alexa dataset [25]. A few domains from Alexa top 1M dataset are listed in Table II [25].

B. Feature Engineering

In the process of selecting features for conventional machine learning methods, lexical features of URLs are chosen because they are easy to analysis and applied widely to any kinds of DGA families. The lexical features are listed as follow (the label in the parentheses would be used in the figures of the following discussion):

- *Length(l)*: a normal URL usually has a shorter length than a DGA URL because it keeps the style simple.
- *Vowels* and *percentage(v%)*: vowels make the URLs readable for clients, thus should be taken into consideration.
- *Digits(d)* and *percentage(d%)*: Possessing a lot of numbers in a URL often means they have been generated automatically because they become less comprehensible when digits are involved, especially when they are present without order.
- *Dot(.) in URL(dot)*: The dots separate different portion of a URL (TLD, SLD, etc.). If a URL has too many dots in there, the credibility of that domain is undermined.
- *Normal suffix test(nT)* and *malicious suffix test(mT)*: The suffix is also taken into consideration, for an ordinary website generally has a familiar suffix, such .com, .org, .net, .co, etc. Hence, all the suffixes from both the most popular domains and from the DGA-generated domains are extracted to serve as features for traditional machine learning methods. Suffixes from DGA domains are labeled with 1 while suffixes from normal domains are labeled with 0.

TABLE I
DGA EXAMPLES FROM 360 NETLAB

DGA family	TLD features	SLD features		# domains generated daily	Examples
		length	*Char*		
conficker.a	com, net, org, info, biz	5-11	a-z	250	ydqtkptuwsa.org
conficker.b	cc, cn, ws, com, net, org, info, biz	5-11	a-z	250	bxyozfikd.ws
virut	com	fix length of 6	a-z	10,000	hgznsb.com
chinad	com, org, net, biz, info, ru, cn	fix length of 16	mix a-z and 0-9	1,000	29cqdf6obnq462yv.com
suppobox_v1	net	Combine two word from a wordlist		∼254	childrencatch.net
suppobox_v2	net, ru	Combine two word from a word list		∼783	thinkgoodbye.ru
gameover	com, org, biz, net	20-28	mix a-z and 0-9	1,000	2id0lapmam6w1799w7315zaqj5.com

TABLE II
NORMAL URL EXAMPLES FROM ALEXA TOP 1M

Top #	Domain Name
1	google.com
2	youtube.com
3	facebook.com
4	baidu.com
5	wikipedia.org
6	yahoo.com
7	qq.com

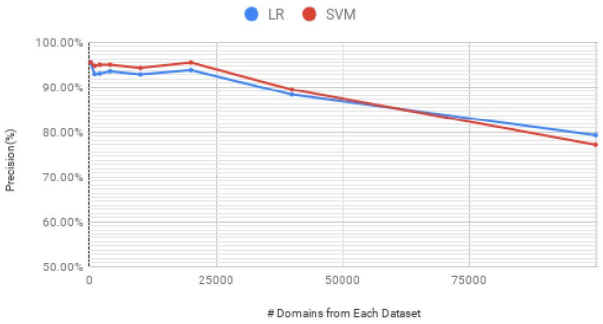

Fig. 4. LR and SVM Precision (%) for Different Amount of Testing Data with Full Features

C. Feature Comparison

During the experiment, the precision of an individual feature is measured when Logistic Regression (LR) and Support Vector Machine (SVM) models are applied to 2000 domains from each source. This comparison is presented in Fig. 3.

Fig. 3 shows that every feature, no matter which model is used, can individually produce a precision rate above 40%. The length, the percentage of vowels, and the malicious suffix test features of the URL stand out from the rest because each one of them achieves a precision rate above 60%. These results demonstrate that different features have a different influence on the overall precision.

D. Testing Set Size Comparison

Following the feature comparison is an experiment designed to compare the performance of different learning models. For this purpose, the size of the testing set is modified while holding other components constant by using full features and the same training set of 40,000 domains. The precision trend is illustrated in Fig. 4.

There is a general trend of decline for both models as the size of the testing set increases. Moreover, while the performance of SVM is better than Logistic Regression when the testing set size is relatively small, LR performs slightly better than SVM when a larger testing set is adopted.

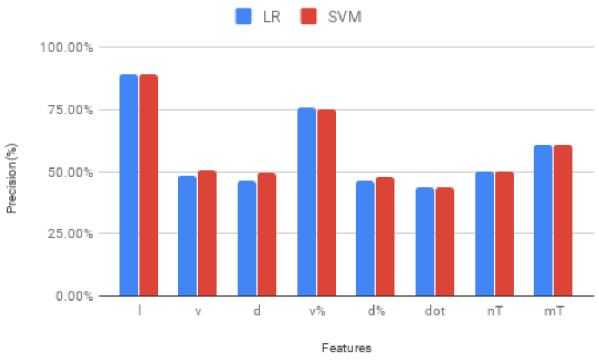

Fig. 3. LR and SVM Precision (%) for Each Single Feature with 4000 Domains

E. Classifier Comparison

In this section, a juxtaposition between the performance of conventional machine learning models and that of DBLSTM model is analyzed. For each trial, the size of the testing set is held constant at 4000 domains. When applying DBLSTM, a testing set randomly selected from the two sources is also used to make the testing set more objective. The performance of LR, SVM, and DBLSTM are compared in Fig. 5.

Fig. 5 shows that there is an evident improvement in precision rate when LSTM model is adopted. When the data from the testing set is randomly selected, the precision rate rises to 98.6%.

F. Long Short-Term Memory newwork (LSTM) Adjustment

In order to explore the effect of different parameters on DGA-generated domains detection, this section identifies the influence of the adjustment made to the model by changing the embedding size of each character in the domain and the

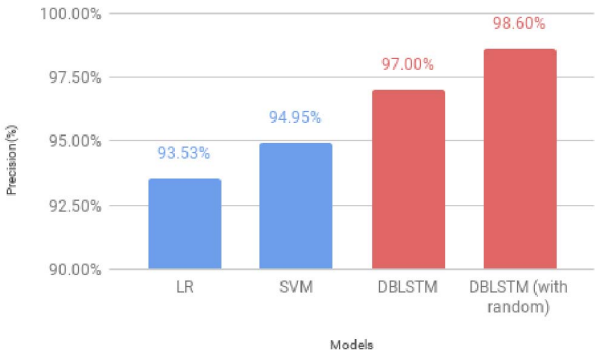

Fig. 5. Performance of Different Models with 4000 domains

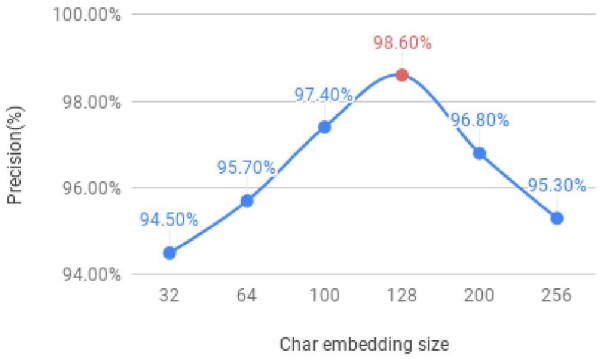

Fig. 7. Precision (%) vs. Char Embedding Size

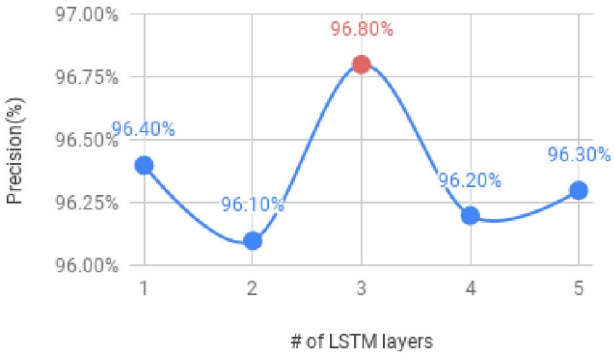

Fig. 6. Precision (%) vs. # of LSTM Layers

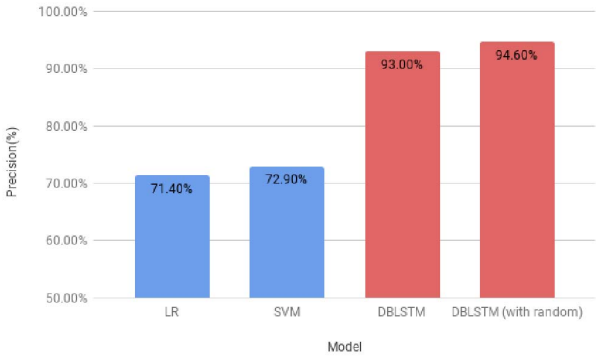

Fig. 8. Performance of Different Models in Unknown DGA Simulation

number of LSTM layers. The specific effects are illustrated in Fig. 6 and Fig. 7.

Fig. 6 shows that given the same task, the model has the best performance when the hidden size equals to 3 while holding the other parameters constant before adjusting embedding size.

Fig. 7 shows that given the same task, the model has the best performance when char embedding size equals to 128, holding the other parameters constant.

G. Unknown DGA Simulation

The previous sections tested algorithms with same data source for training set and testing set. In this section, the proposed algorithm is tested with a dataset consisting of 4000 domains generated by the following method in order to simulate unknown DGAs. The method automatically creates a URL of length twenty in a format of (char, digit, digit, char, ...), in which each component is generated randomly. The precision of algorithms is demonstrated in Fig. 8.

Fig. 8 shows that the precision of the proposed DBLSTM algorithm still remains high at 93-95% while that of conventional models such as LR and SVM dropped significantly to lower than 71-73%.

V. Conclusion

This paper presents different approaches to detect DGA-generated domains based on the features of URLs. The result proves that the DBLSTM algorithm is superior to other conventional machine learning methods. The source code is posted on GitHub for other groups to use or to reproduce the same result (https://github.com/liangy2001/Using-Deep-Learning-to-Detect-Malicious-URLs). The deep learning technique presented in the paper can be widely utilized in the realm of cybersecurity, especially for energy network security, to detect attacks initiated by different domain generation algorithms.

Admittedly, improvements can be made to produce better results. First, the drastic fall in precision rate for the conventional machine learning method can be minimized by selecting more lexical features. Additional research can be done to study other computational linguistic features such as n-gram. However, it also proves the advantage of deep learning technique because with DBLSTM algorithm feature engineering is unnecessary. In the future, research will also be carried out to simulate cyberattacks and examine the efficiency of the deep learning algorithm in a real-time situation.

REFERENCES

[1] K. Zhou, S. Yang, and Z. Shao, "Energy internet: the business perspective," *Applied Energy*, vol. 178, pp. 212–222, 2016.

[2] W. Zhong, R. Yu, S. Xie, Y. Zhang, and D. H. Tsang, "Software defined networking for flexible and green energy internet," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 68–75, 2016.

[3] H. Hua, Y. Qin, C. Hao, and J. Cao, "Optimal energy management strategies for energy internet via deep reinforcement learning approach," *Applied Energy*, vol. 239, pp. 598–609, 2019.

[4] K. Wang, J. Yu, Y. Yu, Y. Qian, D. Zeng, S. Guo, Y. Xiang, and J. Wu, "A survey on energy internet: Architecture, approach, and emerging technologies," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2403–2416, 2018.

[5] K. Zetter, "Inside the cunning, unprecedented hack of ukraine's power grid," Jun 2017.

[6] J. Zhang, P. A. Porras, and J. Ullrich, "Highly predictive blacklisting.," in *USENIX Security Symposium*, pp. 107–122, 2008.

[7] A. Karasaridis, B. Rexroad, D. A. Hoeflin, *et al.*, "Wide-scale botnet detection and characterization.," *HotBots*, vol. 7, pp. 7–7, 2007.

[8] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: detecting the rise of dga-based malware," in *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pp. 491–506, 2012.

[9] C. Livadas, R. Walsh, D. E. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic.," in *LCN*, pp. 967–974, Citeseer, 2006.

[10] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis.," in *Ndss*, pp. 1–17, 2011.

[11] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting p2p botnets through network behavior analysis and machine learning," in *2011 Ninth Annual International Conference on Privacy, Security and Trust*, pp. 174–180, IEEE, 2011.

[12] A. Feizollah, N. B. Anuar, R. Salleh, F. Amalina, S. Shamshirband, *et al.*, "A study of machine learning classifiers for anomaly-based mobile botnet detection," *Malaysian Journal of Computer Science*, vol. 26, no. 4, pp. 251–265, 2013.

[13] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*, pp. 247–255, IEEE, 2014.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1245–1254, ACM, 2009.

[15] F. Vanhoenshoven, G. Nápoles, R. Falcon, K. Vanhoof, and M. Köppen, "Detecting malicious urls using machine learning techniques," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, 2016.

[16] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3422–3426, IEEE, 2013.

[17] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 371–372, ACM, 2014.

[18] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[19] O. E. David and N. S. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2015.

[20] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence*, pp. 137–149, Springer, 2016.

[21] M. Cheng, Q. Xu, J. Lv, W. Liu, Q. Li, and J. Wang, "Ms-lstm: A multi-scale lstm model for bgp anomaly detection," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pp. 1–6, IEEE, 2016.

[22] B. Athiwaratkun and J. W. Stokes, "Malware classification with lstm and gru language models and a character-level cnn," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2482–2486, IEEE, 2017.

[23] C. Olah, "Understanding lstm networks."

[24] Netlab.360.com, "DGA data."

[25] "Alexa Top 1M Sites."