2018 IEEE Symposium on Security and Privacy Workshops

S.P.O.O.F Net: Syntactic Patterns for identification of Ominous Online Factors

Vysakh S Mohan, Vinayakumar R, Soman K P[1] and Prabaharan Poornachandran[2]

Abstract— **With more emphasis on internet as a primary mechanism for information access and communication, it is highly important that the platform stays safe and secure for anyone who uses it. Online scams and cybercrimes are becoming a common threat to the technology and systems that help mitigate these issues are in high demand. Businesses all over the world invest heavily to stay secure in the cyberspace and rely on security experts in defending their business from online threats. The immense scale of the internet and the dynamicity of the threat it holds forces the adoption of automated threat detection systems. Several cybersecurity use cases exist, but the two use cases discussed here are DGA detection and Malicious URL detection. This paper addresses the drawbacks of previous rule-based and machine learning based detection methods. Here, embedding concepts from NLP is incorporated into cybersecurity use cases to propose a new in house model christened S.P.O.O.F Net, which is a combination of a Convolutional Neural Network and Long Short Term Memory Network. The proposed model is benchmarked with machine learning algorithm incorporating bi-gram feature engineering techniques and also a conventional CNN with character level embedding (same as the one used for S.P.O.O.F Net). It was observed that S.P.O.O.F Net gave better performance over the aforementioned methods with accuracy scores of 98.3% for DGA detections and 99% for malicious URL detection. This work also aims to demonstrate the possibilities of incorporating NLP concepts to cybersecurity use cases and provide future researches a new thinking curve to develop systems in this domain.**

I. INTRODUCTION

In current scenario there is no denying the fact that people rely on internet as their primary means for communication and data sharing. The internet era is booming and more and more poeple migrate to the cyber world for its vast pool of information. Social networking, E-commerce etc contribute to the traffic experienced in the cyber network. With such advanced technology comes its own issues. Every year major tech companies fall victims to large scale cyber attacks, which also puts their customers at risk of losing personal information. Often times these attacks could deem the user losing both his identity and money just because the system was compromised. Such situations should be taken care of seriously and a secure environment for data transaction should be offered to any end user. Lack of awareness among the users can also escalate the situation.

[1]Vysakh S Mohan, Vinaykumar R and Soman K P is with Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore.
vsmo92@gmail.com, vinayakumarr77@gmail.com, kp_soman@amrita.edu

[2]Prabaharan Poornachandran is with Center for Cyber Security Systems and Networks, Amrita School of Engineering, Amritapuri, Amrita Vishwa Vidyapeetham, India.

Cybercrimes are on the rise, with recent most globally impacted one being the ransomeware attack. All these cybercrimes, both big and small, costs businesses time, money and resources. These attackers could have either a targeted attack strategy or a more generic one. Usual attack strategies involve luring of victims to malicious web-pages, introducing vulnerabilities on victim's device, identity theft, denial of service attacks, complete infestation of the user's system etc. This puts the user at risk of losing data and often find themselves tricked or betrayed by these illegal infringement to privacy. This could raise concerns about the security of an individual, which also could deem the technology unsafe for secure transactions. Careful and accurate detection of threats like a website being malignant or benign is one way to avoid users falling victims to such attacks. As easy as it sounds, this task has its own challenges and difficulties. Impersonation or duplication of such attacks could make the detection quite hard to perform, deeming the system inefficient. The two cases of baleful activities discussed here are malicious Uniform Resource Locators (URL) and Domain Generation Algorithms.

A subset of Uniform Resource Identifier (URI), Uniform Resource Locator (URL), helps the identification of location and fetching of resources from the network of computers. An unsuspecting user could end up in a malicious website that he/she was introduced through an email or a message because they may not be aware of the nature of the URL presented to them. An attacker generally uses compromised uniform resource locator (URL) to stray the users to malicious websites[1]. These URLs are dispensed through social media platforms or via emails. Authors of [2] claim that one third of all websites are malignant, which makes use of malicious URLs to plot cyber crimes. One means of doing this is by making use of rogue websites. The attacker through these website displays unsought information or content in the form of spam, phishing, malware etc to commit fraudulent financial thefts and identity theft of the unknowing user who happens to fall victim to the attack. Diversity of content on the internet is one aspect the attacker uses in his favour to hide these malignant URLs and present it to the unsuspecting user. All these could be avoided, if there is a strategy to detect, identify and isolate these types of malicious URLs and save the user from significant damages.

Existing commercial systems are usually based on either blacklisting, regular expression or signature matching algorithms[3]. These systems may not quickly detect existing malicious URLs and may not scale to entirely new ones, unless subjected to rigorous update schedules. Slow detection

DOI 10.1109/SPW.2018.00041

times could deem the system redundant as this helps the attacker to loot the user of valuable data by the time the system flags any potential threats. Both aforementioned systems require constant monitoring and updation from a domain expert.

The Domain Name System (or Server or Service) (DNS) is a major element in the operation of internet. Its known for its many key characteristics like its distributed nature, scalability, reliability, it is dynamic and its a database that is globally available and helps in mapping domain name to IP address and vice-versa. It is ironic that these characteristics turn out to be the 'Achilles heel' of this technology. Cybercriminals exploit the vulnerabilities in these characteristics to host malicious content or aid in managing phishing websites for theft of user data[4]. Recent times have seen crucial changes in the way these attacks are served. Botnet[5] deployment is one way this is carried out. Distributed denial of service (DDoS) attacks, large-scale spam campaign, identity theft, sniffing traffic, key logging, malware distributions etc are served via these botnets. Most widely used defence mechanism by enterprises or businesses are blacklisting and sinkholing, which shuts the communication link between the coomand and control server (C2C) and bot master[6],[7]. Here too, the requirement to constantly update and maintain the systems manually makes it daunting and cumbersome. Constant improvements to botnets and persistent advancements in evasion strategies means any poorly maintained defenses could be deemed redundant quite quickly. A common attack strategy employed by an attacker to evade blacklisting is DNS agility or fluxing methods, which in most cases may be an IP flux or domain flux service[8]. The case discussed here is the domain flux service. Domain generation algorithms are used by domain flux services to generate domain names on a massive scale and communicate to their CC server in a trial and error fashion. Blacklisting or sinkholing all domain names prior to their deployment is a way to tackle such attacks. Sinkholing involves reverse engineering the malware to detect the seed. The seed may then be used to hack the botnets by registering incoming domain names as a spoof C2C server. Any malicious author needs to re-establish the botnets with an updated seed to reinstate the systems functionality. As the dynamicity of the generated domains increases, it also escalates the difficulty to deploy the aforementioned rule based methods, as it simply fails to identify newer malicious domain names. These methods are only good when there is sound knowledge of the algorithm and the seed used by botnet attack.

Both URL and DGA detection fails when such rule-based mechanisms are employed. This is solved to an extent by employing machine learning based classifiers to classify the incoming URL or domain names as being malignant or benign. One common machine learning approach is the deployment of a DGA/URL classifier that resides within the network to alert the administrator upon detection of a DGA generated domain name or a malicious URL. In a highly dynamic environment these methods simply fails. This could be attributed to the way these machine learning classifiers work. Explicit feature engineering is a primary requirement for these classifiers. Common feature extraction techniques include entropy, string length, alpha numeric characters, vowel to consonant ratio etc. Explicit knowledge of features and constant maintenance and update of these systems are required. Feature generation and representation engineering is often tedious, also finding the best accuracy of the model using chosen representation is quite cumbersome. Requirement for a large labeled training URL corpus and the need for continual analysis of the system to cater to dynamic changes in patterns of URLs, works against the performance and reliability of these classifiers.

Deep learning is a subdivision of machine learning and is a widely used method to reduce the expense of training procedure, while handling raw inputs instead of manually generated feature representations. The deep learning strategies discussed here are extended from [9] and [10]. In this paper we suggest S.P.O.O.F-Net, which uses a combination of Long Short-term Memory (LSTM) and a Convolutional Neural Network (CNN) for the classifier pipeline. The advantage of using deep learning algorithm is that it requires no feature engineering to be done prior to training, as it takes in raw domain names or URLs as input and the non-linearity in the hidden layer helps it to learn highly complex and abstract features.

The structure of the paper is discussed as follows. Section II discusses about the background knowledge for character level encoding of domain names, LSTM and CNN, Section III details the proposed methods, the dataset description is given in Section IV, Section V details the experiments and results and finally Section VI forms the conclusion.

II. Background

This section explains about the various deep learning algorithms used and gives an abstract level explanation about the math that goes behind it. Also discussed here is the way in which the URLs are transformed into sequential inputs.

A. Character level encoding

URL/Domain name encoding involves character level representation of URLs/domain names via preprocessing and tokenization. Preprocessing converts the characters in a URL/domain name, to lowercase and provides a default key allocation of 0 for unknown characters. URLs and domain names are tokenized using character level tokenize, where they are sliced into character tokens. Non-sequential and sequential representations are commonly used for URL and domain names. Here, we consider both representations for a comparative analysis of the results they offer. For non-sequential representation, we use bi-gram representation and for sequential we use dictionary based representation. Non-sequential representations do not preserve the spatial correlation among the characters, whereas in sequential representation, the dictionary creation is done through assigning unique key for each character in the URL or domain name corpus. Now based on the frequency of occurrence, the characters are placed in ascending order in a dictionary. Each character in the domain name and URL is assigned an index of the dictionary. These character vectors are made to be of same length by padding zeroes to shorter ones and discarding vectors that exceeds some fixed length.

B. Logistic Regression

It is considered as one of the most widely used machine learning algorithm for both classification and prediction. A more generic description of logistic regression is that its a statistical tool to analyze data when there are one or more independent variables determining the output. Logistic regression is a special case of linear regression, where in, the logistic regression predicts the probability of outcome by fitting the data to a logistic function given as,

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{1}$$

C. Convolutional Neural Network (CNN)

CNNs are widely used neural net for computer vision tasks. It has been used with character level embeddings for text classification tasks[11]. CNN is efficient and faster for training and predictive analysis on sequential data[12]. Perks of using this method is that they do not require syntactic knowledge of the language. CNN architectures typically comprise an input layer, a couple of convolutional layers, maxpooling layers and fully connected layers with some non-linear activation function. In text based applications 1-D convolutions, 1-D maxpoolings and fully connected layers are used.

Consider a URL or domain name $D = \{c_1, c_2, ...c_l\}$, where c depicts the characters and l is the length of URL or domain name. An embedding matrix $V^D \in R^{dx1}$ is used for character level representation of a URL or domain name, where d is the dimensionality of the character embedding.

A 1-D convolution layer includes a filter operation $H \in R^{dx1}$, which is operated on a domain name or URL character to form a feature map f_m. Mathematical formulation of this for a window of characters $V[*, j : j+c]$ is as shown below,

$$f_m^D[j] = f(\Sigma(V[*, j : j+c]^{\odot})) \tag{2}$$

where $b \in R$ is the bias term, f is the activation function, usually a *ReLU* or *tanh*, $\odot$ is the element-wise multiplication between two matrices. Also the windows of characters in the domain name or URL $D = \{c_1, c_2, ...c_l\}$, is subjected to a convolutional filter operation.

The resultant feature map is subjected to 1-D pooling operation to obtain more significant features. Pooling is simply a down-sampling operation to reduce the dimension of the feature map. Finally we have a fully-connected layer and a classifier layer which has its activation function as a *sigmoid*.

D. Long-Short Term Memory

LSTMs are special type of Reccurent Neural Networks (RNN), which introduced the concept of a memory cell. The function of these memory blocks are to remember previous information about the entity its learning. Blocks can decide how much information it needs to retain on the basis of gates within them. These memory blocks contain a memory cell and a couple of gates. A memory cell is like a container and has a constant error carousel (CEC) component. While the cell do not receive any input, the CEC has a fixed value 1. An LSTM contains an input gate (ig), forget gate (fg), output gate (og), memory cell m and a hidden state vector (hi) at each time step t. The output of the aforementioned gates are between 0 and 1. The transition function for each LSTM unit is written below,

$$ig_t = \sigma(w_{ig}x_t + P_{ig}hi_{t-1} + Q_{ig}m_{t-1} + b_{ig}) \tag{3}$$

$$fg_t = \sigma(w_{fg}x_t + P_{fg}hi_{t-1} + Q_{fg}m_{t-1} + b_{fg}) \tag{4}$$

$$og_t = \sigma(w_{og}x_t + P_{og}hi_{t-1} + Q_{og}m_{t-1} + b_{og}) \tag{5}$$

$$m1_t = tanh(w_m x_t + P_m hi_{t-1} + b_m) \tag{6}$$

$$m_t = fg_t^i \odot m_{t-1} + ig_t \odot m1 \tag{7}$$

$$hi_t = og_t \odot tanh(m_t) \tag{8}$$

where x_t is the input at time step *t*, σ is the *sigmoid* non-linear activation function and $\odot$ denotes the element wise multiplication.

III. Proposed Method

This work proposes S.P.O.O.F Net, which stands for Syntactic Patterns for identification of Ominous Online Factors. It's a combination of a CNN and an LSTM net. Also the proposed architecture is pitched against a normal 1-D CNN and logistic regression to analyze its performance and advantages. Basic layout can be broadly broken down into three: a character embedding (both for URL and domain name), feature extraction phase and a binary classifier. Following sections details the structure of the suggested pipeline and Fig. 1 shows its overall layout.

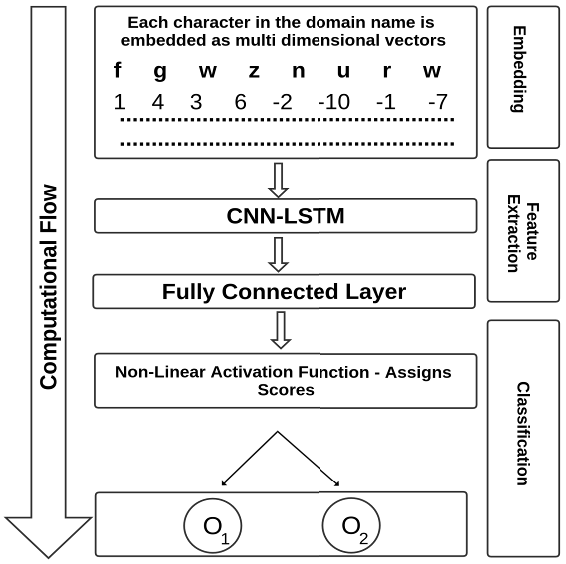

Fig. 1. Proposed architecture of S.P.O.O.F Net

A. Embedding

The train and test corpus for both URL and DGA are subject to some pre-processing steps. Uppercase characters in the URLs and domain names are converted to lowercase value as it is computationally costly to make neural net learn features to make itself case sensitive. Vector representations of the corpora are obtained using a dictionary. Followed by this the vectors are transformed to be of same length. This returns the train and test data that can be subjected to embedding. The *Keras* deep learning library provides powerful embedding functionality, which help obtain character level embedding to serve as input to the convolutional layer of S.P.O.O.F Net. One advantage of using *Keras* embedding is that it adjusts the embedding style it follows as the training progresses, which ensures optimal embedding approach for given data. The 128 dimensional vectors obtained from these embeddings is visualised in a 2-D linear projection through PCA with t-SNE and this is displayed in the Fig. 2. It is evident from the figure that characters that share similar features are clustered together, which means that the model has captured the contexual and semantic similarity of characters that have mutually dependent characteristics.

B. Feature Learning

Feature learning in neural nets refer to the process of the net learning optimal weight parameters to make sense of the data or in other words its the technique through which the net automatically discovers representations needed for understanding the raw input data. Here, the feature learning is handled by a 1-D convolutional neural network and an LSTM. The various elements in the feature learning pipeline is discussed below.

1) Convolutional Neural Network and Long Short Term Memory Network: CNNs are well known for their ability to learn spatial attribute of the given input data. Here, this quality of CNN is exploited to learn spatial co-relation among characters in the input data. A 1-D CNN is used, which incorporates within it multiple 1-D convolutional layers. Several filter numbers were tried out, like 8, 16, 32, 64, 128 and 256 out of which 128 and 256 filters gave the best result and were almost comparable, so 128 was chosen to reduce computational complexity. Each filter is made to slide over the character embedding vector sequence to output a continuous value at each step. This gives representation of, the extent to which the pattern has matched in the character embedding sub-sequence. A pool length of 2 is used here and output is normalized. Regularization of the output is done by using dropout of 0.01. These steps ensure the model over-fitting is less likely and makes the training process a whole lot faster. Output of the former operations is supplied as input to the LSTM. LSTM in the proposed architecture uses 50 memory blocks. Resulting output of the LSTM is passed to the classifier layers.

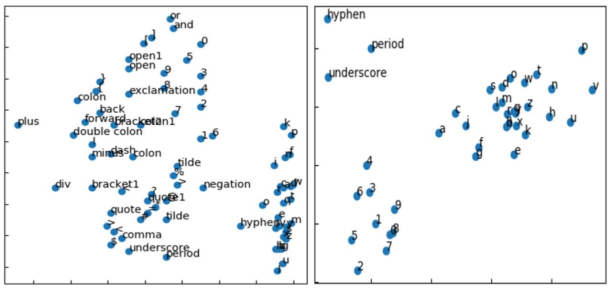

Fig. 2. Embedded character vectors learned by CNN-LSTM binary classifier for URL (left) and DGA (right) is represented using 2-dimensional linear projection (PCA) with t-SNE.

C. Classification

A binary classifier is intended where it classifies any incoming domain name or URL as malignant or benign. Various layers of the classification phase is detailed below.

1) Fully-connected layer: Features obtained from the previous layers are made to pass to the fully-connected layer. This layer is a cheap way of learning the deep features learned by the deep layers. A distinguishing characteristics of fully-connected layers are the connections it has to all the neurons in its preceding layer. The fully-connected layer is followed by the classifier.

2) Sigmoid classifier: The problem in hand demands for a binary classifier and this leads to the choice of *sigmoid* as the activation function for the final output layer. It gives a confidence score of 1 for malignant and 0 for benign input. Binary crossentropy is the loss function used for optimizing the classifier output and it is shown below,

$$loss(p,e) = -\frac{1}{N}\sum_{i=1}^{N}[e_i log(p_i) + (1-e_i) log(1-p_i)] \quad (9)$$

where p is the vector of predicted labels and e is the truth or expected label vector.

IV. Dataset Description

The dataset used for this work is proprietorially developed by crawling several websites, the details of which are discussed below.

A. Domain Generation Algorithm (DGA) Dataset

Domain generation algorithms (DGA) is a prominent method for malwares to build efficient attack strategies. This is achieved through DGAs ability to pseudo-randomly generate millions of domain names to connect to C2C servers on a periodic basis for information access and malware infiltration. For this work, a legitimate list of benign domain names were assembled by using Alexa[13] and OpenDNS[13] and malicious domain names are generated using publically accessible algorithm[14] and OSNIT DGA feeds[15]. Split of DGA data is shown in Table I.

B. Universal Resource Locator (URL) Dataset

Crawling for legitimate URLs were done from sources such as Alexa.com and DMOZ directory and malicious URLs were sourced from malwareurl.com, Phishtank.com, OpenPhish.org, malwaredomainlist.com and malwaredomains.com. Splits for URL data is shown in Table I.

TABLE I
STATISTICS OF DGA AND URL DATASET

Use case	**Training**	**Testing**
DGA	232511	125012
URL	160101	90101

V. EXPERIMENTS AND RESULTS

The work proposes a hybrid neural net architecture which is a combination of a CNN and an LSTM, christened S.P.O.O.F Net. TensorFlow[16] deep learning library was considered as the software framework acting as the backend for the Keras API. To speed up the gradient computations, the model was trained on a single NVIDIA GK110BGL Tesla k40 GPU. This section explains the various details regarding the model parameters and experimental methods followed.

A. S.P.O.O.F Net Architecture

An overview of the proposed model is discussed below, where we detail the embedding strategy, the feature extraction mechanism and finally the classification.

1) Character Embedding: The Keras API provides proprietory character level embedding provisions built-in to itself. The advantage of using this method is that it tunes the embedding layer while training to obtain the best possible character embedding for the domain name and URL corpora. Train matrix for the DGA dataset has a size of 232511x37, while the test matrix is of size 125012x37, where the column size is the length of the domain name in the corpus. Similarly the URL dataset is split into a train matrix of size of 160101x1235 and test matrix of size 90101x1235, where column size is the length of the URL. These matrices are fed to the embedding layer of size 128, which maps every individual character to a 128 length vector. Embedding size is a hyperparameter and this embedding layer works in tandem with other deep layers of the S.P.O.O.F Net attempting to group similar characters into separate clusters. This embedding style preserves the semantic and contextual similarity in the structure of characters in URL or domain names.

2) CNN-LSTM Layers: CNN-LSTM configuration in the proposed pipeline handles the feature extraction part. The advantage of such a configuration is its capability to learn abstract features from dynamic input data to provide highly rich feature representations. Also, this ensures that the feature engineering strategy remains secure against any malicious adversaries, thus making the system defenses reliable compared to its competitions. For DGA detection architecture, it starts with a 1-D convolution layer of size 1x35 with a depth of 128, a batch normalization layer followed by a maxpooling that returns vectors of size 1x8 with a depth of 128 which is succeeded by a layer applying a dropout of 0.01. The output vector of this CNN is fed to an LSTM with 50 memory blocks , which is followed by a fully-connected layer and then the final classifier. The URL detection architecture differs from the aforementioned configuration only in the CNN architecture. Here the 1-D convolution layer has a size of 1x1233 with 128 depth layers followed by batch normalization and a maxpooling that returns 1x308 dimensional vectors which is succeeded by the application of a dropout of 0.01. Rest of the configuration is similar to the DGA detection pipeline. The LSTM output is fed to a binary classifier that classifies the domain name or URL as malignant or benign. The proposed architecture gave really good performance compared to its competitors. It managed 98.3% accuracy for DGA detection and 99% for malicious URL detection. S.P.O.O.F Net also outperformed its peers in the benchmarks performed, where it was pitched against a logistic regression using bi-gram feature engineering and a conventional 1-D CNN.

B. Evaluation of Results

The performance of trained model is evaluated on the test split (for both DGA and URL dataset) described in Table II. Various other hybrid configurations were tested during the course of this work. Following are some of the model architectures that were tried out,

- Stacked convolutional 1-D layer followed by RNN/GRU/IRNN/CWRNN layer in hybrid.
- Stacked convolutional 1-D layer to capture non-linear convolutional activations architecture.
- Stacked RNN/LSTM/GRU/IRNN/CWRNN layers.

Out of this the CNN-LSTM hybrid performed best. Comparison of two operating characteristics of the model like the true positive and false positive rate across varying threshold in the range [0.0 - 1.0] is demonstrated in the form of an ROC curve, which can be found in Fig. 3. This metric is not associated with the amount of malicious and benign samples in the corpus. Details of performance scores obtained for logistic regression, CNN and S.P.O.O.F Net are shown in Table II. Deep learning models have outperformed the bi-gram based machine learning algorithms, which means that the embedding with deep layers can be used to obtain robust and relevant features which makes it a good feature extraction strategy. The runtime for S.P.O.O.F Net was promising for both DGA and URL detection. It managed 41.6 seconds for DGA and 35.57 seconds for URL detection on the test split. Good performance was delivered by this hybrid framework, giving accuracy of 98.3% for DGA detection and 99% for malicious URL detection. The S.P.O.O.F Net is designed to

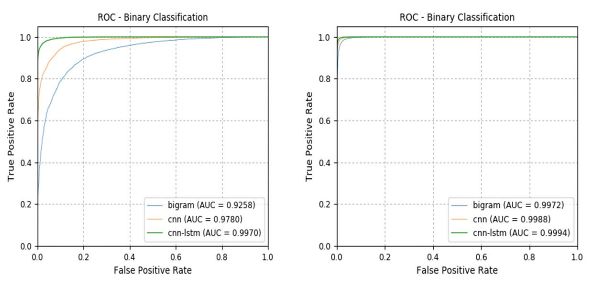

Fig. 3. ROC Curves for DGA (left) and URL detection (right).

TABLE II
PERFORMANCE SCORES FOR VARIOUS MODELS

Algorithm	Accuracy	Precision	Recall	F-score
DGA				
CNN	0.956	0.965	0.985	0.975
CNN-LSTM	0.983	0.985	0.995	0.990
bigram-logistic regression	0.915	0.923	0.985	0.953
URL				
CNN	0.982	0.991	0.974	0.982
CNN-LSTM	0.990	0.985	0.995	0.990
bigram-logistic regression	0.976	0.982	0.969	0.976

deliver a scalable platform to serve near real-time situational awareness to the user by safeguarding him/her from online phishing attacks, malwares etc. It is capable of providing early warning signals well before a large scale attack or malware propogation happens. The proposed framework can analyze and correlate DNS information at multiple Tier-1 Internet Service Provider scale as well as, it can be deployed on a consumer grade server to provide analysis on more than 2 million events per second at near near real-time. With more powerful resources, the performance of the model can be enhanced exponentially. It is this scalability and real-time performance that sets the model apart from its peers.

VI. CONCLUSION

This paper proposes S.P.O.O.F Net (*Syntactic Patterns for identification of Ominous Online Factors*), a CNN-LSTM hybrid to detect and classify malicious domain names as well as malignant URLs. The proposed architecture was found to outperform existing threat detection strategies like blacklisting, sinkholing and machine learning based classifiers for two cybersecurity use cases, namely DGA detections and malicious URL detection. S.P.O.O.F Net overcomes drawbacks of the aforementioned methods, like the requirement of a domain level expert for constant maintenance of the database the classifier is trained on, because the threats are ever changing. Also the knowledge of the feature engineering strategy employed by the classifier lets the attacker bypass the system's defenses. Dynamic generation of domain names makes these classifiers redundant in real life. Deep learning algorithms are well known for their abstract feature learning capabilities and can adapt well to the dynamic nature of the inputs. They can obtain optimal feature representations themselves and they are often considered as a *black-box* when it comes to their feature engineering strategies, which makes them secure against malicious adversaries who intend to circumvent their detection mechanism. The research also benchmarked the proposed architecture against classic machine learning algorithm (logistic regression with bi-gram feature engineering) and a conventional deep learning method employing a 1-D CNN with Keras character level embedding strategy. S.P.O.O.F Net outperformed its competitors in the benchmarking, where it returned detection accuracies of 98.3% for DGA detections and 99% for malicious URL detection. The proposed hybrid architecture proved a better performer than conventional deep learning algorithms and machine learning classifiers for both use cases considered. Logistic regression with bi-gram feature engineering managed a score of 91.5% for DGA detection and 97.6% in URL detection whereas the CNN managed a score of 95.6% and 98.2% for DGA detection and URL detection respectively. The computational bottleneck with the available hardware restricted this work to smaller strings and less complex architectures, but with more powerful hardware, more complex architectures can be tried out for other use cases like, spam detection, log analysis etc. This work tries to demonstrate the relevance of incorporating concepts of natural language processing towards cybersecurity use cases and provide researches with a new direction for clubbing concepts from multiple domains toward deep learning based applications in cybersecurity.

REFERENCES

[1] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012.

[2] B. Liang, J. Huang, F. Liu, D. Wang, D. Dong, and Z. Liang, "Malicious web pages detection based on abnormal visibility recognition," in *E-Business and Information System Security, 2009. EBISS'09. International Conference on.* IEEE, 2009, pp. 1–5.

[3] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious url detection using machine learning: A survey," *arXiv preprint arXiv:1701.07179*, 2017.

[4] Y. He, Z. Zhong, S. Krasser, and Y. Tang, "Mining dns for malicious domain registrations," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on.* IEEE, 2010, pp. 1–6.

[5] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on.* IEEE, 2009, pp. 268–273.

[6] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *International Workshop on Recent Advances in Intrusion Detection.* Springer, 2014, pp. 1–21.

[7] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, and G. Vigna, "Analysis of a botnet takeover," *IEEE Security & Privacy*, vol. 9, no. 1, pp. 64–72, 2011.

[8] G. Ollmann, "Botnet communication topologies," *Retrieved September*, vol. 30, p. 2009, 2009.

[9] R. Vinayakumar, P. Poornachandran, and Soman.K.P, *Scalable Framework for Cyber Threat Situational Awareness based on Domain Name Systems Data Analysis.* Springer (In Press), 2017.

[10] R. Vinayakumar, Soman.K.P, and P. Poornachandran, "Evaluating deep learning approaches to characterize, signalize and classify malicious urls," in *Journal of Intelligent Fuzzy Systems (In Press)*, 2017.

[11] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[12] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[13] "Does alexa have a list of its top-ranked websites," available at https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites, Accessed: 2017-10-02.

[14] "Github repo of publically accesible algorithms," available at https://github.com/baderj/domain_generation_algorithms Accessed: 2017-10-28.

[15] "Bambenek consulting - master feeds," available at http://osint.bambenekconsulting.com/feeds, Accessed: 2017-10-06.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

VII. ACKNOWLEDGMENT

This research was supported in part by Paramount Computer Systems. We are also grateful to NVIDIA India, for the GPU hardware support and to Computational Engineering and Networking (CEN) department for encouraging the research.