Siamese Neural Network and Machine Learning for DGA Classification

Lander Segurola-Gil, Telmo Egues, Francesco Zola and Raúl Orduna-Urrutia
Digital Security Department, Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia/San Sebastian, Spain
lsegurola@vicomtech.org
tegues@vicomtech.org
fzola@vicomtech.org
rorduna@vicomtech.org

Abstract: Domain Generation Algorithms (DGA) are systems used to create immediate multiple and varying domain names. Such "artificial" domains can be then used for siting command and control servers which in turn oversee recruiting/infecting devices, and finally turning them into new resources to be exploited. In this sense, identifying DGA domain names can be crucial, to avoid cyberattacks like Phishing, Spam sending, Bitcoin mining, and many other. Usually, domain names generated by DGAs, are comprised by illegible character strings, but new "intelligent" DGAs tend to generate names using combination of words in dictionaries making its detection a challenging task. For this reason, in this work, we propose to address this problem using a combination of Machine Learning algorithms for improving the classification of DGAs domains. In particular, we propose to combine Siamese Neural Networks and traditional supervised Machine Learning algorithms in order to expand the input domain into separable n-dimensional data points and then achieve the domain classification. The proposed approach can be separated into 3 phases. In a first phase, domain names are encoded, by a one-hot encoder and a variation of this, named probabilistic one-hot encoder, which are implemented separately. Then, in the second phase, Long Short-Term Memory and Convolutional Siamese embedders are tested and compared. In particular, the first one is combined with the one-hot, while the Convolution algorithm is applied with the probabilistic one-hot encoded data. In the final step, five Machine Learning algorithms are tested using the two ways embedded data. Both embedder approaches reach very high results in terms of F1-score and Accuracy (about 91%) depending on the implemented classifier. The promising results obtained by the application of the proposed method shows that it is possible to perform DGA domain classification uniquely over the domain names, without considering external information such as DNS packets features.

Keywords: Siamese Neural Network, DGA classification, cybersecurity

1. Introduction

In the actual world, paradigms like 5G, IoT, Industry 4.0 and so on have increased the interconnectivity among millions of devices, facilitating many operations and increasing the productivity (Rao, 2018). However, being connected everywhere and at every time has triggered increasingly disruptive cyberattacks, which can generate huge impact in the daily life of many people (Bada, 2020). Hacker groups or individuals have the power to create massive botnets and zombie farms that wait until their leader gives the order to attack a website, take down a server farm or DDOS an opposing faction (Mirkovic, 2004). To create such zombie farms, hackers need to infect, recruit and take control of thousands and thousands of devices (Perrone, 2017). This goal can be reached by deploying several cyberattacks like Phishing and Spam sending (Aaron, 2010), which are based on fooling the trustworthiness of the user by using a seemingly legitimate-looking message from a trusted-looking sender (Baykara, 2018).

DGAs (Sood, 2016) comprise a family of algorithms with the aim of immediately creating vast, varying amount of domain names. Created domains often are used for locating command and control servers, targeting devices to be recruited/infected, turning them into new resources to be exploited. Usually, even if DGAs generate domain names comprised by illegible character strings, later trends base these algorithms in word dictionaries, creating human readable domain names. For this reason, in this work, we propose a novel method for DGAs classification, based on combining Siamese Neural Network (SNN) and Machine Learning (ML) algorithms.

The proposed approach is divided into three phases: in the first one, the input data (domain names) is encoded using the one-hot. In the second phase, a Siamese Neural Network is trained for converting the encoded data in embedded features, and finally, in the third phase, these embedded features are used for training and testing different machine learning classifiers.

To the best of our knowledge, this is the first work that addresses the DGA classification combining one-hot encoder and Siamese technologies, for firstly embedding and the classifying data. This study not only shows promising results that can be used for improving the state of the art of phishing detection, but also implement and evaluate different architectures to highlight which is more tied for this specific use case.

The rest of this paper is structured as follows: in Section 2, basic concepts for the compression of the work and related works are introduced. In Section 3, the followed methodology is presented, detailing every component in the process, starting from the encoding phase, passing through the embedding phase, to end with the classification phase. In Section 4, the dataset, experiments and the validation process are introduced. In Section 5, the results of the experiments are illustrated, and finally, in section 6, the conclusions and the future work guidelines are drawn.

2. Background

In this section, several concepts are introduced, which conform the basis of this work. In Section 2.1, a general description of one-hot encoding and Siamese Neural Networks structure are presented and in Section 2.2 the state of the art related to this research is shown.

2.1 Preliminaries

In this paper, two innovative technologies are exploited during the experiments, *one-hot encoder* used for extracting numerical information from the initial domain name dataset and *Siamese Neural Network (SNN)* used for converting such numerical information into embedded features.

One-hot Encoding: The one-hot encoder encodes data by generating a canonical basis for a set of terms, i.e., it is a function mapping every term in the set of terms, to a vector with dimension equal to the set size, where every value of the vector is 0 except for a value, which is equal to 1, and no image of the set of terms is equal. This can be extended for every sentence generated by the combination of these terms, by concatenating terms encodings, generating a matrix of dimensions length of the sentence times the size of the terms set.

When working on categorical data, there is a need of encoding it, due to how Machine Learning works. Several encoding methods can be found, but in the scenario posed in this work, an order preserving encoding might be differential. Due to the nature of the one hot encoding (concatenation of term encodings), the order of terms in each sentence is preserved in the encodings, making this choice especially suitable for this task.

Siamese Neural Networks: The Siamese Neural Networks (SNN) (Chicco, 2021) are specially structured neural networks, usually consisting of two parallel neural networks connected to an output layer. The last one calculates the distance between the outputs of each neural network, trying to minimize distance between points belonging to a same set and maximizing the distance between points belonging to different sets.

Siamese Neural Networks learn how to separate point sets in the training. Each of the networks learns to embed input belonging to a same family in different spatial places. This makes this kind of structure interesting, which can help to enhance classification or clustering tasks.

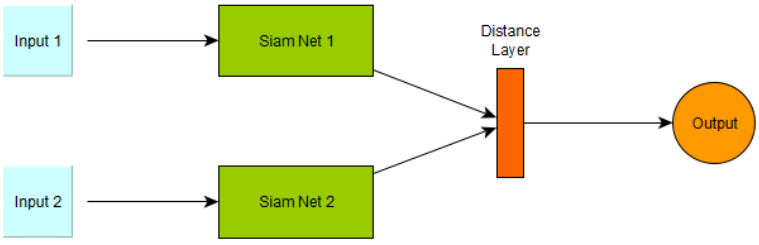

Figure 1: Siamese Neural Network

They belong to the set of supervised learners, receiving data as pairs of points, labelled with the aim of representing equality or inequality between both points. Binary labels are used usually for training these types of structures.

2.2 Related works

In (Ravi, 2021), botnets using DGA domains and DNS homographs are detected using deep learning. They use this technology to detect the DGA and verify the model's viability by testing them against common adversarial attacks. (Vinayakumar, 2019) shows an approach to DGA detection also using deep learning. However, they create a whole detection chain by not only detecting the domains through semantic similarity, but also embedding the domains in case the semantic similarity did not trigger the alarms. On the same line, (Hwang, 2020) presents a method to detect and classify DGA by extracting features and passing them to a CNN-based model that labels the domains as DGA or legit. More deep learning based DGA detection research works can be found, such as (Tuan, 2022) where LSTM based techniques are used, or (Aravamudu, 2022) where various ML classifiers are tested against this task. Techniques involving more conventional Machine Learning techniques can be found too. For example, in (Anand, 2020), lexical and statistical features are extracted for a total of 44 features from domain names, and then conventional algorithms such as the RF are applied. In (Chin, 2018), it is proceed in a similar way. The authors extract some lexical features from the domain names, together with some DNS features, for DGA ML based classification and clustering.

The technology underlying these detectors are commonly related to deep learning (Yu, 2018). In particular, Siamese Networks perform exceptionally well in word embedding projects. (Neculoiu, 2016) is a clear example of this, where a bidirectional LSTM is used to embed the words into a fixed space. (Amin, 2019) uses a Manhattan LSTM Siamese model to extract features and information from domains while in (Benajiba, 2019), the authors tackle the semantic pattern similarity problem by implementing a Siamese model to detect SQL pattern alikeness. (Zhu, 2018) goes even beyond by making a comparison between a D-LSTM model that extracts primary sentence information and generates basic sentence components through a standard LSTM to determine which works better in compositional knowledge. In (Pontes, 2018), a combination of Siamese CNN and LSTM are used in order to detect textual similarity; while CNN notices the local context of words, the LSTM Siamese takes into account the global context of the sentence. (Vinayakumar R. A., 2021) implemented a two-level deep-learning framework involving a Siamese network to detect botnets in Internet of Things networks. Close to these approaches, in (Ling, 2019), the authors implemented a multi-stage banking trojan botnet detection system using deep learning to identify DGA domains and (Kwon, 2016) presents a botnet detection implementation that can be used in large-scale DNS traffic environments.

3. Methodology

In this section, a novel methodology is presented, which encodes and Siamesely learns to embed domain names, with the aim of classifying these into DGA based name or into a legit one. Specifically, in Section 3.1 the motivation of carrying this work and the contributions made are presented. In Section 3.2, the encoding phase is explained, followed by the Section 3.3, which deepens the Siamese training phase and finishes in the Section 3.4, which details the ML algorithms used for the classification.

3.1 Motivation and contributions

When trying to detect DGA created domain names, several techniques have been studied. Common old techniques include blacklisting or Regular Expression (regex) matching. Latest techniques include ML based techniques for filtering anomalous domain names. For example, in (Lin, 2019), they combine both techniques to enhace detection. However, DGA techniques keep evolving, creating more sophisticated DGAs, such as the dictionary-based ones. For this reason, in this work, we propose a classification enhancing approach, based on Siamese embeddings.

3.2 Encoding phase

In a first phase, domain names need to be encoded in order to get numerical features, so the ML algorithms can potentially learn the patterns from data. For this purpose, two approaches have been taken, one-hot encoder and a probabilistic one-hot encoder.

One-hot encoder: As mentioned in Section 2.1, the one hot encoder has been the one chosen for vectorizing natural language data for the training of the LSTM based Siamese. This will encode data by considering lowercase letters, numbers and hyphens as the set of terms to be encoded, for a dictionary of 39 terms, precisely comprising the characters permitted for the domain names.

Probabilistic one-hot encoder: Let D be a set of documents d, and t terms in d. Then the amount of documents d in D containing the term t is defined as the document frequency:

$$\mathrm{df}(t, D) = \frac{|\{d \in D : t \in d\}|}{N}$$

Equation 1: probability of a term appearing in an arbitrarily chosen document

where N = |D|. I.e. the probability of a term appearing in a given random document d in D. For this encoding, instead of encoding each of the terms of the dictionary as the canonical basis of 39 dimensioned Euclidean space, the 1 appearing with a normal one-hot is substituted by df(t,D). This encoding is the one selected for the Convolutional Siamese training.

In both cases matrixes will have 39 columns corresponding to the vocabulary size. Rows amount will vary, matching with domain names length. Then, truncating and padding operations will be performed over the encoded data.

3.3 Siamese training phase

As introduced in Section 2.1, the common SNN are trained by comparing the results of two dense NN. However, in this work, due to the complexity of the input data, as well as the dependency that each letter has with its previous and next character, we explore two different Siamese architectures: one based on Long-Short Term Memory (LSTM) and another based on Convolutional Neural Network (CNN). More specifically, the first one is used together with the one-hot encoder, while the second is used with the probabilistic one-hot encoder. The normal one-hot fits well with the LSTM Siamese due to its structure. However, the Convolutional Siamese can potentially increase its performance when a value is given to each letter, so the max pooling layer does not trivialize data. This is the reason why this election is made for this work.

LSTM based Siamese model: For the LSTM based model, two parallel LSTM models are implemented composed of two LSTM layers where the first one comprises a number of neurons equal to the vocabulary size (39) and a second layer of a hundred neurons. The scalar product between the output of both models is computed in a connecting layer, which is connected to a final neuron with a sigmoid activation, for the minimization of the loss mean absolute error.

Convolutional Siamese Model: For the convolutional model, two parallel convolutional models are developed, both having two blocks of a convolutional layer (with 32 filters for the first layer and 64 for the second, two sized in both cases) and a max pooling layer, connecting both to a flatten layer and finishing in a hundred neuroned layer. The output of both layers is connected to a layer composed of a hundred neurons, which connects with an output neuron with a sigmoid activation function, and a mean squared logarithmic error loss, which replaces the usual distance layer.

Both models will have as input the matrixes generated by the encoding phase. When embedders are extracted, those will embed those matrixes into a 100 dimensional Euclidean space, with the aim of providing this data to the machine learning classifiers.

3.4 Machine learning classifiers

For the classification task, 5 common machine learning classifiers are trained with the embedded data. In this way, it is possible not only to validate the information generated by the two SNNs, but also to evaluate it through different classifiers and choose which is the best in this specific use case. The selected learners comprise Random Forest (RF), Multilayer Perceptron (MLP), Decision Tree (DT), Support Vector Classifier (SVC) and Linear

Regression (LR). More concretely, each model is configured with particular parameters: RF has 50 trees with a max depth of 25 nodes, the MLP is composed by a hidden layer of 100 neuron, with a sigmoid activation function, a batch size of 250, is trained within 100 epochs, and the RMSProp optimizer and mean squared error loss, the DT has max depth of 25 nodes, the SVC is chosen with a radial basis function is used as the kernel, and finally the LR, where the intercept is chosen to be fitted.

4. Experimental study

In this Section, the dataset and the pre-processing operations are introduced (Section 4.1), as well as the used evaluation metrics (Section 4.2). Finally, in the Section 4.3, the two experiments are drawn.

4.1 Data overview and pre-processing

The dataset used in this research was gathered by investigators of the Polytechnic University of Marche in order to develop a DGA detector and DGA-family classifier based on n-gram features (Cucchiarelli, 2021). The dataset is a collection of DGA-based and benign domains, where the DGA-based domains are divided into 25 different DGA families and the benign domains have been provided by Alexa's top site. Both mischievous and legit domains are equally distributed and have been shuffled so none of the DGA families, nor the valid domains are grouped together in the list. Malicious domains were extracted from the 360 Netlab Opendata Project repository, a repository maintained by the 360 Netlab Research Lab (https://netlab.360.com/). The whole dataset consists of 674,898 domain names, where 337,398 are domains labelled as benign domains by Alexa, and the remaining 337,500 comprise the malicious domains set, uniformly distributed for each DGA family, with 13,500 samples.

For the experiments, the initial dataset is split into a Siamese training dataset, a ML train dataset and a test dataset with a proportion of 10%, 30% and 60%, respectively. More specifically, the Siamese train dataset is used for training the SNN, then, the ML train dataset is embedded by the one of the Siamese models, and the output is used for training the classifiers. This process is repeated on the ML test set, which will be used to evaluate the whole chain (SNN and ML classifier). This configuration allows us to obtain validation results over a complete unseen dataset increasing the generalization of the results and decreasing the probability of overfitting results. Furthermore, a 5-fold cross validation process is performed, which means that each evaluation is repeated 5 times changing the used dataset composition but leaving unchanged their overall representation (keeping the indicated proportions). Finally, it is important to highlight that, for the SNN training, pairs of samples are generated from the Siamese training set considering all possible combinations. However, due to the huge amount of possible combinations (the square of the size), just a 0.005% of all combinations are used, randomly selecting samples.

4.2 Evaluation metrics

For the evaluation of the performance of the models, 4 metrics are used, comprising the Accuracy, Precision, Recall and F1-score, which are derived from the confusion matrix (Figure 2) related to binary classification.

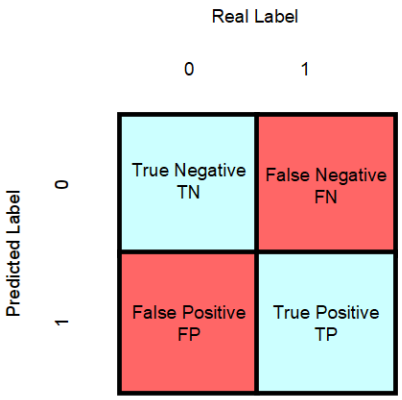

Figure 2: Confusion matrix

The values observed in Figure 2 comprise

- **True Negative (tn)**: The predicted and the real value equals to 0 (negative class)
- **False Negative (fn)**: The predicted value equals to 0 but the real value is 1 (positive class)
- **False Positive (fp**): The predicted value equals to 1 but the real value equals to 0.
- **True Positive (tp)**: Both, the predicted and real values equal to 1.

These four metrics are computed from these values, over the ML models and both Siamese models.

Accuracy: It measures the hit rate. It is computed by summing *tp* and *tn* divided by the total amount of samples (the sum of all four values), as it can be observed in Equation 2.

$$\frac{tp + tn}{tp + fp + tn + fn}.$$

Equation 2: Accuracy

Precision: It measures the positive hit rate in relation of true positives. It is computed by dividing the amount of *tp* by the total amount of positives (the sum of *tp* and *fn*) as in Equation 3.

$$\frac{tp}{tp + fn}.$$

Equation 3: Precision

Recall: It measures the positive hit rate in relation of predicted positives. It is computed by dividing the amount of *tp* by the total amount of predicted positives (the sum of *tp* and *fp*) as it can be seen in Equation 4.

$$\frac{tp}{tp + fp}.$$

Equation 4: Recall

F1-score: The harmonic mean of the precision and recall. It is calculated by the formula shown in Equation 5.

$$2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

Equation 5: F1-score

Those are the four metrics used for the measurement of the performance of the models. In particular, the measure for the ML classifiers is performed over the test set and for the Siamese model, the validation set is used for the computation of these metrics.

4.3 Experiments

First experiment: The aim of this experiment is to prove that a convolutional embedder can potentially work for DGA classification. For this, domain names are firstly vectorized by the probabilistic one hot encoding. For this, the df(t,D) (Equation 1) is calculated over all the domains comprising the training set of the Siamese. In other words, every domain forming the pairs for the Siamese training set are considered for computing the probability of a term of the previously defined characters appearing in a domain is computed. Then probability one-hot is applied to these, which will generate the matrixes for feeding the convolutional Siamese model.

Once the model is trained, one of both Siamese models is arbitrarily extracted to embed the remaining data. Once the embedder is extracted, the remaining data (the training and testing sets for the Machine Learning Classifiers) is one-hot encoded and Convolutionally embedded.

Finally, the embedded training set is provided to the selected five ML classifiers and the performance of these is measured over the test set, considering the four metrics presented in Section 4.2.

Second experiment: This experiment is like the previous one. This time, domain names are encoded using the one hot encoder for the generation of the matrixes. Once again, even in this case they are binary matrixes, those are used to feed the LSTM based Siamese models.

AS in the first experiment, when the Siamese Network has finished its training, an LSTM based embedding is obtained by the subtraction of one of the Siamese models. Again, the splits of the remaining data, the training and testing datasets for the classifiers, is one-hot encoded and LSTM based embedded.

Finally, and keeping the same structure, the embedded training set is used to feed the five ML classifiers, and the four metrics introduced in Section 4.2 are used for evaluating the classifiers over the embedded test set.

5. Results

In this section, the results of the evaluation of the proposed method are presented, in terms of the presented metrics. In the Section 5.1, the results for each experiment will be divided into two tables, showing the values for the respective Siamese model in a table, and the scores of the classifiers in another. The values are computed by calculating a mean and standard deviation among the 5 folds. In Section 5.2 a discussion of the obtained results is written.

5.1 Experiments' Results

- **First experiment**

For the Convolutional Siamese model, the results gotten are the ones observed in Table 1.

Table 1: Convolutional Siamese results

	Accuracy	Precision	Recall	F1-score
Convolutional Siamese	0.946±0.004	0.95±0.004	0.941±0.006	0.945±0.004

As it can be noticed, the Convolutional Siamese discerns with a high rate (95% of accuracy) the DGA domains from the legit domains. In the second phase of the proposed method, the Table 2 shows the performance indicators for all five machine learning classifiers.

Table 2: Performance of the classifiers (after Convolutional embedding)

	Accuracy	Precision	Recall	F1-score
RF	0.893±0.002	0.918±0	0.863±0.003	0.889±0.002
MLP	0.897±0.002	0.918±0.006	0.873±0.008	0.895±0.002
DT	0.839±0.002	0.835±0.002	0.843±0.003	0.83±0.002
SVC	0.888±0.002	0.919±0.002	0.851±0.003	0.883±0.002
LR	0.88±0.001	0.909±0.001	0.854±0.003	0.884±0.001

Results show that the one getting the best overall results has been the Multilayer perceptron, closely followed by the Random Forest, where the first one has an 89.7% of accuracy, and the second a 89.3%, both have 91.8 of precision, 87.3% and 86.3% of recall respectively, and 89.5% in the F1-score for the MLP and 88.9% for the RF. The Supported Vector Classifier and the Linear Regression achieve similar results compared to the best two classifiers, whereas the Decision Three falls behind, getting the scores diminished by a 5%-7%.

- **Second experiment**

For the LSTM based Siamese model, the results can be observed in Table 3: LSTM based Siamese results:

Table 3: LSTM based Siamese results

	Accuracy	Precision	Recall	F1-score
LSTM Siamese	0.822±0.176	0.823±0.177	0.722±0.367	0.855±0.117

The results gotten by the classifiers when LSTM based embedded data is provided, can be seen in Table 4 .

Table 4: Performance of the classifiers (after LSTM embedding)

	Accuracy	Precision	Recall	F1-score
RF	0.915±0.039	0.929±0.037	0.90±0.043	0.912±0.04
MLP	0.885±0.092	0.894±0.097	0.875±0.088	0.884±0.092
DT	0.873±0.061	0.874±0.061	0.871±0.061	0.873±0.061
SVC	0.868±0.116	0.879±0.116	0.85±0.125	0.864±0.121
LR	0.902±0.055	0.917±0.044	0.881±0.072	0.899±0.059

The results are less stable than in the first case, getting high standard deviations specially for the Siamese part (±0.367 in Recall and ±0.177 in the rest). The best classification model in this experiment has been the RF (even if the MLP achieved greater results when the Siamese model learnt well). Scores about 90%-92% can be observed in the four metrics for the RF

5.2 Discussion

As it can be perceived in the first experiment, the Convolutional Siamese discerns quite good the legit domains and the DGA based ones (95%) in the validation set. When classifying with the ML models, the results are still of about a 90% in the best classifiers, even if performance has been diminished. This could be since the distance layer in the Convolutional Siamese is computed by another dense layer. This fact may convert both Convolutional models "lazier" in a sense that this layer may not only fit the distance function, but help in the classification of the pairs, making both models suboptimal. This could potentially explain, even if it is low, the reduction of performance of the classifiers when compared to the Convolutional Siamese.

For the second experiment, higher values can be perceived in the standard deviation and lower in the mean. In fact, the highest values for the LSTM based Siamese model were higher than for the convolutional (about a 97% in the four metrics) and a higher enhancement was observed in the classifiers (about a 95% in all the four metrics for the Multilayer Perceptron). However, the learning phase was not steady in the Siamese part. It learnt well in three of the five folds but behaved strangely in the remaining two. Optimal parameter tunning and structural modifications could be potential solutions for this issue.

6. Conclusion and future work

The idea of this work has been to observe the behaviour of some of the most common and state of the art classifiers when applying Siamese embedders to data for DGA domain classification. The results show that the method should be strongly considered. In fact, and in particular, the good results gotten by the Linear Regression (even if in the first case are a bit worse) classification with both Siamese embedders, indicate good separability of both (DGA and legit) sets. This leads to interesting future work. First, an optimization of the Convolutional Siamese networks could be achieved, by building a better structured network. Optimal parameter tunning could be found for both cases too, in particular for the LSTM case, in which data folds changing produced irregular results. Then, a multiclassification task could be interesting. Differentiating between DGA families may not be such a critical matter, but the separation could be done into legit, random DGAs and dictionary based DGAs. These three subfamilies may be embedded into separable data sets by Siamese embedders.

Acknowledgements

This work has been partially supported by the Basque Country Government under the ELKARTEK program, project TRUSTIND (KK-2020/00054).

References

Aaron, G. (2010). The state of phishing. *Computer Fraud & Security*, 5--8.

Amin, K. L. (2019). Advanced similarity measures using word embeddings and siamese networks in CBR. *Proceedings of SAI Intelligent Systems Conference* (pp. 449-462). Springer.

Anand, P. M. (2020). An ensemble approach for algorithmically generated domain name detection using statistical and lexical analysis. *Procedia Computer Science*, 1129--1136.

Aravamudu, P. a. (2022). Exploring and Comparing Various Machine Deep Learning Technique algorithms to Detect Domain Generation Algorithms of Malicious Variants. *Computer Science and Information Technologies*.

Bada, M. a. (2020). The social and psychological impact of cyberattacks. *Emerging cyber threats and cognitive vulnerabilitie*, 73--92.

Baykara, M. a. (2018). Detection of phishing attacks. *2018 6th International Symposium on Digital Forensic and Security (ISDFS)* (pp. 1--5). IEEE.

Benajiba, Y. S. (2019). Siamese networks for semantic pattern similarity. *2019 IEEE 13th International Conference on Semantic Computing (ICSC)* (pp. 191-194). IEEE.

Chicco, D. (2021). Siamese neural networks: An overview. *Artificial Neural Networks*, 73--94.

Chin, T. a. (2018). A machine learning framework for studying domain generation algorithm (DGA)-based malware. *International Conference on Security and Privacy in Communication Systems*, {433--448.

Cucchiarelli, A. a. (2021). Algorithmically generated malicious domain names detection based on n-grams features. *Expert Systems with Applications*, 114551.

Hwang, C. K. (2020). Effective DGA-Domain Detection and Classification with TextCNN and Additional Features. *Electronics*, 1070.

Kwon, J. L. (2016). A scalable botnet detection method for large-scale DNS traffic. *Computer Networks*, 48-73.

Lin, H. a. (2019). Detection of application-layer tunnels with rules and machine learning. *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage* (pp. 441--455). Springer.

Ling, L. G. (2019). *An AI-based, Multi-stage detection system of banking botnets.* arXiv:preprint arXiv:1:1907.08276.

Mirkovic, J. a. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 39--53.

Neculoiu, P. V. (2016, August). Learning text similarity with siamese recurrent networks. *Proceedings of the 1st Workshop on Representation Learning for NLP.*, (pp. 148-157).

Perrone, G. a. (2017). The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices. *IoTBDS*, 246--253.

Pontes, E. H.-M. (2018). *Predicting the semantic textual similarity with siamese CNN and LSTM.* arXiv:preprint arXiv:1810.10641.

Rao, S. K. (2018). Impact of 5G technologies on industry 4.0. *Wireless personal communications*, 145--159.

Ravi, V. A. (2021). Adversarial Defense: DGA-Based Botnets and DNS Homographs Detection Through Integrated Deep Learning. *IEEE Transactions on Engineering Management.*

Sood, A. K. (2016). A taxonomy of domain-generation algorithms. *IEEE Security & Privacy*, 46--53.

Tuan, T. A. (2022). On Detecting and Classifying DGA Botnets and their Families. *Computers & Security*, 102549.

Vinayakumar, R. A. (2020). A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities. *IEEE Transactions on Industry Applications*, 4436-4456.

Vinayakumar, R. S. (2019). Improved DGA domain names detection and categorization using deep learning architectures with classical machine learning algorithms. *Cybersecurity and Secure Information Systems* , 161-192.

Yu, B. a. (2018). Character level based detection of DGA domain names. *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1--8). IEEE.

Zhu, W. Y. (2018). Dependency-based Siamese long short-term memory network for learning sentence representations. *PloS one*, e0193919.