Helix: DGA Domain Embeddings for Tracking and Exploring Botnets

Lior Sidi

liorsid@post.bgu.ac.il
Ben-Gurion University
Beer Sheva, Israel

Yisroel Mirsky*

yisroel@post.bgu.ac.il
Georgia Institute of Technology
Atlanta, GA, USA

Asaf Nadler

asafnadl@post.bgu.ac.il
Ben-Gurion University
Beer Sheva, Israel

Yuval Elovici

elovici@bgu.ac.il
Ben-Gurion University
Beer Sheva, Israel

Asaf Shabtai

shabtaia@bgu.ac.il
Ben-Gurion University
Beer Sheva, Israel

ABSTRACT

Botnets have been using domain generation algorithms (DGA) for over a decade to covertly and robustly identify the domain name of their command and control servers (C&C). Recent advancements in DGA detection has motivated botnet owners to rapidly alter the C&C domain and use adversarial techniques to evade detection. As a result, it has become increasingly difficult to track botnets in DNS traffic. In this paper, we present Helix, a method for tracking and exploring botnets. Helix uses a spatio-temporal deep neural network autoencoder to convert domains into numerical vectors (embeddings) which capture the DGA and seed used to create the domain. This is made possible by leveraging both convolutional (spatial) and recurrent (temporal) layers, and by using techniques such as attention mechanisms and highways. Furthermore, by using an autoencoder architecture, the network can be trained in an unsupervised manner (no labeling of data) which makes the system practical for real world deployments.

In our evaluation, we found that Helix can track botnet campaigns, distinguish between DGA families and seeds, and can identify domains generated using the latest adversarial machine learning techniques. Helix is currently being used to track botnets in one of the world's largest Internet Service Providers (ISP), and we include some of the ISP's analysis work using our method.

CCS CONCEPTS

• **Security and privacy** → *Intrusion/anomaly detection and malware mitigation.*

KEYWORDS

DGA, botnet, DNS, autoencoder, embedding, LSTM, CNN

ACM Reference Format:

Lior Sidi, Yisroel Mirsky, Asaf Nadler, Yuval Elovici, and Asaf Shabtai. 2020. Helix: DGA Domain Embeddings for Tracking and Exploring Botnets. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, October 19–23, 2020, Virtual Event, Ireland. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3340531.3416022>

1 INTRODUCTION

Botnets are groups of compromised Internet machines (bots) that are controlled by a single owner to carry attacks such as distributed denial-of-service [9], data exfiltration [17], and cryptocurrency mining [8]. To maintain control over the botnet, the owner configures its bots to connect and receive commands from a command and control server (C&C) in the Internet.

Hardcoding the C&C's IP address makes it easy for firewalls and defenders to blacklist botnet traffic and locate the C&C. Therefore, to make connectivity to the C&C robust, owners register an Internet domain name (e.g., s23kdk.com) and have the bots use the domain name system (DNS) to lookup the current IP address which can be changed if the C&C is taken down. Additionally, modern botnets prevent Internet Service Providers (ISPs) from blacklisting their domain names by using a domain generation algorithm (DGA) [19].

A DGA operates similarly to a random number generator, except it is designed to generate valid domain names. The botnet author configures all of its bots to use the same seed to generate the same random domain, and connect to that domain as their C&C. Periodically, the bots generate the next domain in unison to evade detection of the ISP. Fig. 1 illustrates this process.

Recently machine learning has been applied to the task of DGA domain detection in DNS traffic. To use machine learning, the DGA generated domain names must be represented as a feature vector. In the past, researchers have captured the legitimacy of a domain name by extracting domain registration patterns and DNS resolve statuses [3]. However, ISPs can observe trillions of domains each day. Therefore, it is inefficient to extract these features since it requires constant queries over big data clusters.

Instead, it is preferred to capture domain names by extracting features from the domain name itself. An accepted approach has been to extract textual features such as character entropy, frequency, and counts of specific characters [3, 14, 21, 25, 30]. However, several recent studies have demonstrated camouflage attacks against these textual features, such as mixing benign domain names into the

*This author is also affiliated with Ben-Gurion University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-6859-9/20/10...\$15.00

<https://doi.org/10.1145/3340531.3416022>

Figure 1: An illustration of the process which an attacker performs in order to connect a botnet to a C&C.

Table 1: Summary of how DGA's have been captured for machine learning

Task	Feature Representation			
	Textual	Textual & DNS	Character String	Embedding
Malicious URL detection	Out of scope	[4, 7]	[5, 11, 21]	
DGA Binary Classification	[1, 13, 16]	[22, 24, 32]	[6, 30, 31]	
DGA Multi-class Classification	[14, 29]		[14, 25, 30]	
DGA Representation	[16, 19]			[Helix]

generated ones [2, 12, 18]. Therefore, there is a need for a stronger representation of domain names, one which can capture both the underlying DGA and the presence of camouflage attacks.

In this paper, we propose **Helix**: a method for representing generated domain names by converting them into numerical vectors (embeddings) which capture the essence of their underlying DGAs. To accomplish this, we use a deep neural network autoencoder which uses convolutional layers during the encoding and decoding process to capture local patterns (the textual features), and bi-directional LSTM layers at the center to capture the complex and subtle sequential patterns left by the DGA. The architecture also uses various state-of-the-art techniques for extracting the subtle nuances in sequential data, such as attention mechanisms, denoising, and highway layers. As a result, our embeddings not only can distinguish between different DGA families, but can also differentiate between two botnets which are using the same DGA but with different seeds (i.e., code reuse in separate campaigns).

To the best of our knowledge, this is only work which can capture generated domain names with this level of fidelity, and as such, can be instrumental in tracking and detecting botnets via DNS traffic. Furthermore, we have found that Helix is robust to the camouflage evasion techniques [2, 12, 18] compared a state of the art DGA classifier. Lastly, Helix is practical because autoencoders are trained in an unsupervised manner, meaning that there is no need for manual labeling. As a result, an ISP can easily train the model on trillions of samples gathered from their own DNS traffic.

In our evaluation, we found that Helix (1) outperforms textual features in terms of cluster quality and fidelity, and (2) can form a more robust DGA classifier than the state-of-the-art approach, in the presence of camouflage attacks. Lastly, using Helix, we were able to detect five previously unreported DGA families in one-day of DNS data (8 billion records). We note that Helix is currently being used to monitor botnets in one of the world's largest ISPs, and we have published our code and DGA datasets online for reproducibility.¹

2 THE MOTIVATION FOR HELIX

Previously, the ISP's cyber emergency response team (CERT) was trying to detect botnets by detecting large peaks of traffic on network ports. However this approach was (1) too expensive due to the amount of traffic, (2) ineffective since botnets could only be detected when it was too late (under attack), (3) inaccurate since only large attacks performed in unison (e.g., DDoS) could be detected and not others (e.g., APTs, cryptomining, etc), and incomplete since, the CERT's analysts had no way to determine the malware's source code, author, or variant.

As a result, the ISP turned to us to develop a tool for detecting, monitoring, and tracking botnets in their upper-DNS traffic (easy to obtain) in a manner which is practical and accurate. Helix addressed all of their challenges because (1) it is trained in an unsupervised manner, (2) performs significantly better than textual features, and (3) can track and detect novel botnets based on their DGAs, before any disruptive attacks are launched. In cooperation with the ISP, we integrated Helix into their CERT pipeline over a 6 month period and tuned its performance in cooperation with the CERT's analysts. The CERT currently uses the tool as part of a big data security analytic toolbox which examines the last 24 hours of DNS data, stored in a distributed database, to generate actionable threat intelligence.

Unfortunately, for privacy reasons, we are not privy to the number of botnets detected using the tool since the integration in January 2019. Therefore, in this paper, we only report the results based on the dataset provided to us by the ISP.

3 RELATED WORK

Previous works aimed at DGA classification and detection proposed three common ways for representing DGA-generated domain names as feature vectors: textual features, a mix if textual and DNS features, and the character string itself. Below we briefly describe each of these feature representations. Table 1 summarizes how they have been used for various DGA related tasks.

Textual Features These features capture aspects that can help differentiate between a DGA-generated domain and a natural language string. For example, the domain's (1) length [16], (2) number of

¹<https://github.com/liorsidi/Adversarial-DGA-Datasets>
<https://github.com/liorsidi/Helix>

consonants, number of vowels, digit ratios [22, 24], (3) ratio of meaningful words [7, 32], and (4) domain character entropy (randomness) [13, 32]. Since DGAs are similar to random string generators, character entropy provides good indication whether or not a domain is from a DGA. Let D be a given sample domain name, and let $\{D\}$ denote the set of characters in D . The entropy feature is calculated as $S(D) = -\sum p_i \cdot \log p_i$ where p_i is the probability of the i -th character in the set $\{D\}$ to appear in D .

DNS Traffic-Based Features These are features which capture a domain's meta data provided by the DNS server. For example, in [7], the authors examined the related IP addresses, country of origin, and the DNS time-to-live statistics. Although these features provide a useful context for the machine learning algorithm, acquiring these features may infringe on the users' privacy and incur a significant preprocessing overhead (e.g., IP resolutions and Whois lookups) especially given the massive number of DNS queries made each day.

Character String Instead of summarizing a string as a feature vector, some propose using the character string itself, such as a recurrent neural networks (RNN) which can handle sequential data. There, each of the characters is represented as with one-hot-encoding. For example, $a = (1, 0, 0, \dots)$, $b = (0, 1, 0, \dots)$, and so on.

4 CREATING THE EMBEDDINGS (HELIX)

In our work, we propose using a neural network to find a more meaningful representation of DGA domain names. To obtain the embedding of a DGA domain name, we use a neural network; Specifically, a custom spatio-temporal autoencoder.

An autoencoder is an architecture which consists of two neural networks: an encoder $En(x) = e$ and a decoder $De(e) = x'$, where x is the input and e is the resulted embedding. The network is trained as a single model $De(En(x)) = x$ which tries to reconstruct the same input at the output. However, since e has much fewer dimensions than x , the network En essentially learns how to compress an input into a concise vector (i.e., perform non-linear dimensionality reduction) based on the distribution of the training set.

The spatial aspect of the characters is captured by the autoencoder using multiple convolutional filters of different sizes. These filters essentially search for common patterns and sub sequences. These layers capture the benign domains very well and also capture some of the 'leakage' from the DGA (characters, groupings, etc).

The temporal/sequential aspect of the characters in captured by the autoencoder bi-directional long short term-memory (LSTM) units at the output of En and input of De . The feature maps (outputs from the convolutional filters) are passed through the LSTM instead of the character sequence. This is because (1) domain names can be very long and LSTMs do not perform well on long sequences [20], (2) bi-directional LSTMs have been shown to perform exceptionally well in natural language processing (NLP) and thus captures the benign domains very well, and (3) the LSTM can learn the cyclic patterns which some DGAs use to generate the characters. Therefore, the motivation for this architecture is to capture both local and global patterns in domain name strings which can reflect the originating DGA (see Section 6.2 for an ablation study).

The complete architecture of Helix can be found in Fig. 2. We will now detail each of the layers accordingly:

Input layer The input layer receives the first 63 characters of a second-level domain name (e.g., 'google' from google.com).

Figure 2: The architecture of Helix used to generate embeddings. Note, only the encoder En is used during deployment.

A character can be one of the 37 symbols found in standard domain names (RFC1035 [15]), including English letters, digits, and hyphens. Each character is replaced with an index corresponding to its symbol, and zero padding is added to the end.

Character embedding layer Next, the 63×1 vector of indexes are used to retrieve the corresponding character embeddings (compressed one-hot encodings with a dimensionality of 20). The embeddings are concatenated to produce a 63×20 matrix.

1D convolution layers Four character-aware neural (convolutional) network layers, are used to extract informative patterns from the domain name (i.e., the 63×20 matrix). We use character-aware layers since they have been shown to be very successful at identifying character-level patterns [10]. In contrast to conventional CNNs, the four layers are not executed sequentially on each other's outputs. Rather, the four layers receive the same 63×20 matrix as their input, yet process the matrix using different kernel sizes: (kernel size=1, num. filters=30), (2, 16), (3, 8), (4, 8). The feature maps outputted from each layer is pooled and concatenated together to form a collective feature map.

Highway layer A special gating layer that enables unimpeded information flow across layers of a neural network [23]. Here the highway enables information from the feature maps to be passed directly to the encoding without passing the bi-LSTM.

bidirectional LSTM This layer consists of a dual layer LSTM which maps an input sequence to an output sequence, in a task known as sequence-to-sequence (seq2seq). The first LSTM captures the linguistic dependencies between the characters

from left-to-right, and the latter captures the dependencies from right-to-left.

Attention We employ an attention mechanism to allow the encoder to capture more meaningful information regarding the linguistic patterns of domain names. We use the mechanism of [26] since it is designed for encoder decoder networks used for language processing.

Character softmax encoder This is a time-distributed softmax layer which predicts the character for each “timestep” (character index) based on a maximum likelihood estimation.

We train Helix end-to-end in an unsupervised manner using categorical cross entropy loss defined as $\mathcal{L}_{CCE}(x, y) = -\sum_i^C y_i \log(x_i)$. Therefore, the training objective is to minimize $\mathcal{L}(H) = \mathcal{L}_{CCE}(x', x)$ where H are the trainable network parameters of Helix.

The entire autoencoder network is trained as a denoising autoencoder (DAE). This means that for 20% of the training samples, 35% of the characters were randomly removed during training. This approach helps the network generalize better to unseen domains. This also helps the network become more robust to the malicious manipulation of characters which is common in adversarial attacks. After training the neural network De is discarded and En is used in deployment to generate embeddings.

5 USING THE EMBEDDINGS

There are many different machine learning tasks which can be applied to the embeddings. For example, they can be used as feature vectors for classification models to identify regular domains from generated domains, classify domains according to their DGA family. Doing so may increase detection performance and provide additional robustness to camouflage attacks. Another use is to perform cluster analysis to detect emerging botnets, new seeds, and obtain other threat intelligence.

Typically, an ISP has an internal Cyber Emergency Response Team (CERT) which is responsible for identifying attacks launched against and from within their network. In this section, we discuss how a CERT can deploy Helix for the purpose of botnet campaign tracking. Later, in Section 6, we will evaluate Helix on this task.

5.1 The Pipeline

In order to efficiently track campaigns within their network, the CERT can obtain hourly or daily lists the domain names resolved by their DNS servers. These domain names can then be converted into embeddings by Helix and then added to a clustering model for analysis. There are many different clustering models which can be used. In this paper, we use K-means because many big data clusters offer libraries which can run K-means efficient on trillions of data points (e.g., MLlib in Spark over a Hadoop cluster). Other ISPs may chose a more real-time approach by implementing a stream clustering algorithm on a distributed data stream processing platform such as CluStream over Apache Storm.

We also note that while the training of Helix requires extensive computation (e.g., GPU), the execution of the trained model can be performed on a commodity compute cluster with a standard CPU (Helix has only 226,973 parameters which is relatively small).

In Fig. 3 we present the complete pipeline, which is initialized by the CERT as follows:

(1A) Training

- (1) Collect X_{DNS} : a distinct list of domain names from the DNS over a few months. Note that X_{DNS} contain both benign and malicious domain names.
- (2) Collect X_{DGA} : a list of annotated DGA domain names obtained from community blacklists^a and threat intelligence sharing platforms such as MISP.
- (3) Create the training set: $X_{train} = \{X_{DNS}, X_{DGA}\}$
- (4) Train Helix on X_{train} according to Section 4. We found that our model converged after about 50 epochs.
- (5) Extract En and create the training embeddings as $E_{train} = En(X_{train})$.
- (6) Train a clustering model M_c on E_{train} which captures the DGA families (e.g., Conficker, Virut, etc.) This can be done programmatically by tuning the clustering parameters until $M_c(E_{DGA})$ is fitted to $L(E_{DGA})$, where L retrieves the DGA family names. Concretely, we maximize the Adjusted Rand Index (ARI) [28] between $M_c(E_{DGA})$ and $L(E_{DGA})$ where the ARI is a metric which measures the percentage of correct decisions (cluster mappings) made by the algorithm.

^a<https://nlp.amrita.edu/AmritaDGA> <https://data.netlab.360.com/dga/>
<https://osint.bambenekconsulting.com/feeds/dga-feed.txt>

Note that we train Helix on both benign and malicious generated domain names. One could train Helix on only malicious domains by using a DGA classifier (e.g., Invincea [21]) to curate a malicious dataset from the DNS. Although this would improve the clustering quality, we include the benign domains during training because we expect camouflage and other adversarial attacks aiming to fool DGA classifiers by evading detection.

Periodically (once every few hours or days), the CERT can update their models as follows:

(1B) Model Updates

- (1) Set X_{train}^{t+1} to be X_{train}^t in addition to all new domains obtained from the DNS and DGA lists.
- (2) Continue training En with X_{train}^{t+1} for only a few epochs and compute $E_{train}^{t+1} = En(X_{train}^{t+1})$.
- (3) Cluster E_{train}^{t+1} to obtain M_c^{t+1} , tuned to fit E_{DGA} (as done in the Initialization phase).
- (4) Transfer all annotations from M_c^t to M_c^{t+1} :
For each cluster in M_c^{t+1} , if over 75% of the annotated domains have the same DGA family c , then annotate the cluster as belonging to family c .

During deployment, the CERT can use the models to perform botnet analysis (exploration and campaign tracking). One way to implement botnet analysis with Helix is as follows:

(2A) Deployment: Botnet Analysis

Exploration Because X_{DGA} is annotated, similar DGA domains in X_{DNS} will be clustered with the DGA families in X_{DGA} because of their embeddings will be similar. Therefore, the CERT can identify entire clusters (botnets) in M_c^t using the same DGAs within their network. The CERT can also obtain a situational awareness by tracking the size, origins, and C&C

Figure 3: An example of how an ISP's CERT can use Helix to track and explore botnets in their network by monitoring domain name queries made to their DNS.

locations, of the hosts for each cluster, from M_c^t to M_c^{t+1} and so on. The analysts can also observe the evolution and distribution of DGA usage in their network. For example, a variant (new malware) may be detected when a new cluster is found adjacent/joined to a cluster of a well-known DGA family. Finally, the analysts can also add their own annotations to clusters, such as manually identify outliers and tag novel DGA clusters.

Campaign Tracking When a new botnet is deployed, the bots are given a novel seed. Using Helix, the domains generated from each seed form a micro cluster which is distinct from all other botnets. By monitoring these micro clusters, an ISP can detect and monitor new botnets. This was previously not possible with textual features, since all botnets using the same DGA fall into the same cluster.

To accomplish this, the CERT can:

- (1) Identify DGA clusters using their annotations.
- (2) For the i -th DGA cluster, train a new clustering model $M_{c,i}$ on the embeddings of that cluster.
- (3) Finally, each of the sub-clusters in $M_{c,i}$ captures a new seed/campaign. However, for improved accuracy, the CERT can selecting clusters in $M_{c,i}$ where the majority of host IP addresses are unique with respect to the other micro-clusters in $M_{c,i}$. This is because it is unlikely that the same hosts belong to many other campaigns, and under the same DGA.

The CERT can also use the models to perform DGA detection: the tasks of (1) identifying DGA domains from benign domains, (2) classifying DGA domains by their family, and (3) detecting novel domains generated by a new DGA. In this case, any machine learning classifier model can be fitted to E_{train} by assuming that domains from X_{DGA} are the positive class and those from X_{DNS} are the negative class. However, since we assume that E_{train} is distributed over a big data cluster, we propose the following classifier based on the KNN algorithm. We note that KNN also enables us to perform task #3 (novelty detection) which a classifier cannot perform:²

(2B) Deployment: DGA Detection

When a new domain x (or batch of domains) arrive:

- (1) Compute $e = En(x)$

²Classifiers are closed-world models that can only predict labels seen the training set. Novelty detectors identify new classes/distributions, unseen in the training set.

- (2) Retrieve $E_K = \Gamma_K(e, E_{train})$ where the $\Gamma_K(e, E)$ retrieves the K nearest neighbors of e in E . In our evaluations, we set $K = 5$.
- (3) Classify e :
 - (a) **Check if e is an outlier:**
If $\frac{1}{K} \sum d(e_i, e) > \phi$, where d is the Minkowski distance, then e is labeled as a **Novel Domain Pattern** (potential DGA). Following an analysis of all 5-NN distances in E_{train} , we found that setting ϕ to the distribution's 90-th percentile performed best (see Section 6).
 - (b) **Check if e belongs to the largest malicious class:**
If $\argmax_{\ell \in L} \frac{|\ell \cap L(E_K)|}{|L(E_K)|} > \rho$, where L retrieves labels from X_{DGA} , then x is labeled as **Known DGA Pattern** and e receives the label ℓ (a DGA family name. In our evaluation, we set ρ to 0.75).
 - (c) **Assign e to the majority label:**
In all other cases, x is given the label of the majority class ('benign' or the generic 'DGA'), where all of the DGA family labels in E_K are grouped together.

6 EVALUATION

To justify the architecture of Helix, we first provide results of an ablation study which shows that each component of the network is needed. Next, we compare the quality of our embeddings to the current best method for representing domain names (textual features) through cluster analysis. We then evaluate our suggested implementations for botnet exploration and campaign tracking (Section 5:2A). Finally, we analyze Helix's robustness using several state-of-the-art camouflage attacks on the suggested KNN DGA Detector (Section 5:2B). For reproducibility, we have made the source code and DGA datasets available online (see the footnote in Section 1).

6.1 Datasets

We evaluate Helix using four different datasets:

DGArchive A dataset of 50,000 domains from 50 DGA families taken from the DGA Archive [19]. The DGA families were selected at random to capture a wide variety of behaviors.

AmritaDGA A dataset from 2018 consisting of 100,000 registered benign domain names, and 297,777 DGA generated domain names that were evenly produced from 20 authentic DGAs used in the wild [27].

Figure 4: The character distribution of the domains in ISP vs the generated domains in DGArchive and AmritaDGA.

AdverDGA Our own custom dataset containing samples from three recently published DGAs which use adversarial camouflage techniques to evade detection: DeepDGA [2], CharBot [18], and MaskDGA [12] (available online).

ISP A dataset of domain names collected from the upper-level DNS servers of our partner, one of the world’s largest ISPs. The unlabeled dataset contains approximately eight billion DNS records (22 million distinct domains, 732 GB) collected from a single day, of both benign and malicious nature. Fig. 4 shows the character distribution of **ISP** compared the the generated domains in **DGArchive** and **AmritaDGA**. Note that **ISP** naturally has DGA samples within it. Using the LSTM Classifier from [2] we found that $\sim 4.6\%$ of the distinct domains were from benign and malicious DGAs.

Following the process described in Section 5:1A, we collected the DGA samples from **AmritaDGA** as X_{DGA} and used the benign samples of **AmritaDGA** and the entire **ISP** dataset as X_{DNS} . We split the data for training and testing according to Table 2. Note that since Helix is trained on the ISP’s data in an unsupervised manner, we expect X_{DNS} to have some DGA domains within it. To measure our system’s performance, we used the 50 DGA family labels in **DGArchive** for the ablation, representation, and exploration studies (Sections 6.2, 6.3, and 6.4) and the **AdverDGA** dataset was used to evaluate Helix’s robustness against camouflage attacks (Section 6.6).

6.2 Ablation Study

To validate the architecture of Helix, we performed an ablation study: we evaluated several versions of autoencoders using subsets of Helix’s layers. For each version, we (1) generated embeddings for the **DGArchive** dataset, (2) trained a K-means model with k set to the number of DGA families (50), and then (3) evaluated the clustering performance through classification and clustering metrics. For the classification metrics we used the F1-score, precision, and recall measures. For the clustering metrics we used the Adjusted Rand Index (ARI) and the Normalized Mutual Information measures, where the ARI measures the accuracy of a clustering and NMI measures the entropy reduction for each class.

Table 3 presents the results from this study. We found that the combination of layers used in Helix provides the best results. The reason is because the architecture enables the biLSTM layers to model domains as sequences of patterns (or building blocks) captured by the CharCNN layers. Moreover the attention helps the network focus on the relevant components. As a result the network can learn the distribution of sequences more efficient than CNN or LSTM layers alone.

Table 2: Summary of the Datasets used in the Evaluations:
• Ablation, ◦ Representation, ★ Exploration and ◊ Robustness

	Train	Test	Experiment
DGArchive			
DGA	-	50,000	•◊★
AmritaDGA			
Benign	40,200	13,400	◊
DGA	142,132	10,000	◊
AdverDGA			
DGA (CharBot)	-	10,000	◊
DGA (DeepDGA)	-	10,000	◊
DGA (MaskDGA)	-	10,000	◊
ISP			
Unlabeled Benign & DGA	22,144,594	*	★

Table 3: Ablation Study on the Architecture of Helix

		Layers				Performance					
		CNN		Seq2Seq		Classification			Clustering		
		CNN	CharCNN	Bin-CharCNN	LSTM	biLSTM	F1	Prec	Rec	ARI	NMI
Architecture	Ver1	•					0.36	0.59	0.18	0.29	0.49
	Ver2		•				0.2	0.41	0.06	0.14	0.22
	Ver3			•			0.44	0.74	0.24	0.33	0.55
	Ver4				•		0.47	0.73	0.26	0.37	0.56
	Ver5					•	0.48	0.77	0.29	0.39	0.61
	Helix		•	•		•	0.5	0.78	0.3	0.42	0.63

6.3 Helix as a DGA Domain Representation

To evaluate Helix as a feature representation for machine learning, we compare it to the current best representation: the textual features described in Section 3. We extracted two datasets from **DGArchive**: one using Helix X_H and the other using the textual features X_T . Similar to our ablation study, for each dataset we clustered them, and then computed their performance using the classification and clustering metrics.

Table 4 shows that Helix captures DGAs significantly better than the current best ‘textual features’. This is understandable because the network is able to summarize relevant information (character patterns) based on its observations, whereas the textual features only summarize the given string, adding a significant amount of noise to the feature vector. Since both Helix and the linguistic features were applied to the same dataset, it is clear that the Helix model is the contributing factor and not the size of the dataset.

In Fig. 5, we present scatter plots of the Helix embeddings and the textual feature vectors using the **ISP** dataset. Here only samples which are classified a DGA domain are presented for simplicity. The figure shows that not only are the textual features noisy, but Helix is able to naturally distinguish between the DGA families.

6.4 Exploring Botnets with Helix

Using the embeddings of the **ISP** dataset, we explored the clusters formed by the DGAs. The bottom two subplots of Fig. 5 shows that

Table 4: Helix Compared to Textual Features

	Performance				
	Classification			Clustering	
	F1	Prec	Rec	ARI	NMI
Textual Features	0.36	0.59	0.18	0.29	0.49
Helix Embeddings	0.5	0.78	0.3	0.42	0.63

Figure 5: Scatter plots (PCA reduced) of the textual features (top) and Helix embeddings (bottom) extracted from the ISP dataset, after removing the benign domains. Grey points are unknown DGA domains (not found in public blacklists).

the embeddings were able to capture a wide variety of DGA families active within the ISP’s network.

For this exploration, we observed that Helix generates meaningful embeddings. After PCA reduction, we observe that the largest distinguishing aspect relates to the output length of the DGA (x-axis, pc0: left=short, right=long). The second aspect seems to relate to the algorithmic patterns of the source DGA (the y-axis, pc1). The remaining aspects (other dimensions which are not visible due to the projection) capture entropy and linguistic content. For example, each of the visible clusters can be divided into three distinct macro-clusters: dictionary-based DGAs (e.g., suppbbox, matsnu, and gozi), random letter DGAs (e.g., necurs, cryptolocker, ramdo, and gameover-p2p), and random alpha-numeric DGAs (e.g., urlzone, qadars, and emotet).

Figure 6: Campaign tracking using Helix by summarizing the clusters of the resulting embeddings. Each color represents the domains generated using a selected seed.

To detect novel DGAs in the clustering process, we (1) used a DGA classifier to remove benign domains, and (2) used **DGArchive** as a blacklist to remove known DGA samples. As a result, we had the DGA domains which are currently unknown (the bottom subplot) and were able to identify five new DGA families currently active in the ISP’s network. The fifth group uses Internationalized Domain Names (IDN) that supports foreign languages, and can be tied to phishing attack websites.

6.5 Campaign Tracking with Helix

New botnets often reuse existing DGAs, but each campaign uses a different initial seed. Helix can distinguish between domains generated using different seeds. To demonstrate this, we used existing DGAs to generate a few thousand domain names using 3-4 different seeds. Fig. 6 plots the cluster distributions (pie charts) for each of the seeds. Helix captures dictionary-based DGA very well (top:Suppbbox) and random generators too (bottom:Necrus). Therefore, by clustering the macro-clusters into micro-clusters, we are able to identify new seeds because the distribution shifts.

6.6 Robustness to Camouflage Attacks

To evaluate how robust Helix is to camouflage attacks, we attacked Helix and a recent DGA detector (Invincea [21]) with the state-of-the-art camouflage attacks: CharBot, DeepDGA, and MaskDGA in our **AdverDGA** dataset. CharBot corrupts popular domain names, DeepDGA uses a GAN to generate names that fit the benign distribution, and MaskDGA performs a black box adversarial machine learning attack against arbitrary DGA classifiers.

We then used Helix (Section 5:2B) and Invincea to detect DGA domains in **AdverDGA**. We found that Helix performed better than Invincea on average (F1-score of 0.72 vs 0.67). We also found that Helix performed significantly better than Invincea on the gradient-based attacks (MaskDGA) achieving an F1-score of 0.73 compared to 0.50. The reason why is because the unique perturbations of MaskDGA

Figure 7: A visual mapping (PCA reduced) of the embedded DGA domains from AmritaDGA before (left) and after (right) being camouflaged by MaskDGA.

are captured in the domain name, enabling Helix to distinguish it from the others (visualized in Fig. 7). In the figure, we examine data from AmritaDGA because MaskDGA was trained on it. This is in contrast to CharBot which leaves very little evidence of the attack, resulting in Helix achieving an F1-score of 0.65 vs the 0.66 of Invincea.

7 CONCLUSION AND FUTURE WORK

In this paper we presented Helix: a method for representing DGA domains as feature vectors (embeddings) for botnet exploration, tracking, and many other machine learning tasks. We also presented several ways Helix can be used by an ISP's CERT to explore their daily DNS traffic and automatically detect new botnets in their network over large distributed data clusters (via the KNN algorithm).

We found that not only does Helix outperform the textual features used today, but is also robust to state-of-the-art adversarial attacks which aim to camouflage DGA domains. Finally, we also found that Helix produces high-fidelity embeddings, capable of detecting when new DGA seeds (new campaigns) are used.

As future work, we plan to use Helix to improve current state-of-the-art DGA classifiers, and refine the seed exploration by training Helix on domains from each DGA separately. Helix is currently deployed in the CERT one of the largest ISPs, providing them with emerging threat intelligence, and has significantly improved their situational awareness. Therefore, we believe that this work can help other ISPs and ultimately make the web a safer place.

REFERENCES

- [1] Aashna Ahluwalia, Issa Traore, Karim Ganame, and Nainesh Agarwal. 2017. Detecting Broad Length Algorithmically Generated Domains. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10618 LNCS (2017), 19–34.
- [2] Hyrum S. Anderson, Jonathan Woodbridge, and Bobby Filar. 2016. DeepDGA: Adversarially-Tuned Domain Generation and Detection. (2016), 13–21. <https://doi.org/10.1145/2996758.2996767>
- [3] Manos Antonakakis and Roberto Perdisci. 2012. From throw-away traffic to bots: detecting the rise of DGA-based malware. *Proceedings of the 21st USENIX Security Symposium* (2012), 16. <https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final127.pdf>
- [4] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a Dynamic Reputation System for DNS. *USENIX Security '10: Proceedings of the 19th USENIX conference on Security* (2010), 1–17. http://www.usenix.org/events/sec10/tech/full_papers/Antonakakis.pdf
- [5] Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, and Fabio A. Gonzalez. 2017. Classifying phishing URLs using recurrent neural networks. *eCrime Researchers Summit, eCrime* (2017), 1–8.
- [6] Pavol Bielik, Veselin Raychev, and Martin Vechev. 2017. Character Level Based Detection of Dga Domain Names. (2017), 1–17.
- [7] Leyla Bilge. 2011. E XPOSURE : a Passive DNS Analysis Service to Detect and Report Malicious Domains. V, 4 (2011).
- [8] Jega Anish Dev. 2014. Bitcoin mining acceleration and performance quantification. *Canadian Conference on Electrical and Computer Engineering* (2014), 1–6.
- [9] Nazrul Hoque, Dhruva K. Bhattacharyya, and Jugal K. Kalita. 2015. Botnet in DDoS Attacks: Trends and Challenges. *IEEE Communications Surveys and Tutorials* 17, 4 (2015), 2242–2270.
- [10] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-Aware Neural Language Models. (2015). <https://doi.org/2>
- [11] Hung Le, Quang Pham, Doyen Sahoo, and Steven C. H. Hoi. 2018. URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection. i (2018). <http://arxiv.org/abs/1802.03162>
- [12] Asaf Shabtai Lior Sidi, Asaf Nadler. 2019. MaskDGA: A Black-box Evasion Technique Against DGA Classifiers and Adversarial Defenses. *arXiv preprint arXiv:1902.08909* (2019).
- [13] X Luo, L Wang, Z Xu, J Yang, M Sun, and J Wang. 2017. DGASensor: Fast detection for DGA-based malwares. *ACM International Conference Proceeding Series Part F1280* (2017), 47–53. <https://doi.org/10.1145/3057109.3057112>
- [14] Hieu Mac, Duc Tran, Van Tong, Linh Giang Nguyen, and Hai Anh Tran. 2017. DGA Botnet Detection Using Supervised Learning Methods. *Proceedings of the Eighth International Symposium on Information and Communication Technology* (2017).
- [15] P. Mockapetris. 1987. RFC1035 DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. <https://www.ietf.org/rfc/rfc1035.txt>. (Accessed on 05/26/2020).
- [16] Miranda Mowbray and Josiah Hagen. 2014. Finding domain-generation algorithms by looking at length distribution. *Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014* (2014), 395–400.
- [17] Asaf Nadler, Avi Aminov, and Asaf Shabtai. 2019. Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computers & Security* 80 (2019), 36–53.
- [18] Jonathan Peck, Claire Nie, Raaghavi Sivaguru, Charles Grumer, Femi Olumofin, Bin Yu, Anderson Nascimento, and Martine De Cock. 2019. CharBot: A Simple and Effective Method for Evading DGA Classifiers. *arXiv:1905.01078* (2019).
- [19] Daniel Plohmann, Khaled Yakdan, Michael Klatt, and Elmar Gerhards-Padilla. 2016. A Comprehensive Measurement Study of Domain Generating Malware. *Proceedings of the 25th USENIX Security Symposium* (2016). <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann>
- [20] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. 2017. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078* (2017).
- [21] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv preprint arXiv:1702.08568* (2017).
- [22] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-based botnet tracking and intelligence. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8550 LNCS (2014), 192–211.
- [23] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).
- [24] Van Tong and Giang Nguyen. 2016. A method for detecting DGA botnet based on semantic and cluster analysis. *Proceedings of the Seventh Symposium on Information and Communication Technology - SoICT '16* (2016), 272–277. <http://dl.acm.org/citation.cfm?doid=3011077.3011112>
- [25] Duc Tran, Hieu Mac, Van Tong, Hai Anh Tran, and Linh Giang Nguyen. 2018. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* 275 (2018), 2401–2413.
- [26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
- [27] R Vinayakumar, Prabaharan Poornachandran, and KP Soman. 2018. Scalable Framework for Cyber Threat Situational Awareness Based on Domain Name Systems Data Analysis. In *Big Data in Engineering Applications*. Springer.
- [28] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2009. Information theoretic measures for clusterings comparison: is a correction for chance necessary?. In *Proceedings of the 26th annual international conference on machine learning*.
- [29] Tzy Shiah Wang, Hui Tang Lin, Wei Tsung Cheng, and Chang Yu Chen. 2017. DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. *Computers and Security* 64 (2017), 1–15. <https://doi.org/10.1016/j.cose.2016.10.001>
- [30] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. 2016. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. (2016). <http://arxiv.org/abs/1611.00791>
- [31] Bin Yu, Daniel L. Gray, Jie Pan, Martine De Cock, and Anderson C.A. Nascimento. 2017. Inline DGA detection with deep networks. *IEEE International Conference on Data Mining Workshops, ICDMW 2017-Novem* (2017), 683–692.
- [32] Han Zhang. 2016. BotDigger: Detecting DGA Bots in a Single Network. *Tma* (2016).