OXFORD

The Computer Journal, 2024, **67**, 3094–3104
https://doi.org/10.1093/comjnl/bxae072
Advance access publication date 29 July 2024
Original Article

DGA domain embedding with deep metric learning

Yifan Yang[1,‡], Xionglve Li[1,‡], Tao Yang[1,‡], Bingnan Hou[1], Lingbin Zeng[1], Zhiping Cai[1,*], Wenyuan Kuang[2]

[1]Department of Computer Science, National University of Defense Technology, No. 109, Deya Road, Changsha City, Hunan Province, 410073, China
[2]360 Digital Security Group, Beijing, 100015, China

[*]Corresponding author. E-mail: zpcai@nudt.edu.cn
[‡]Yifan Yang, Xionglve Li and Tao Yang contributed to the work equally and should be regarded as co-first authors.

Abstract

Botnets currently use domain-generation algorithms to produce fast-flux domains that enable them to evade detection. Accurately categorizing these botnet domains is crucial to develop cybersecurity solutions against botnet threats. However, existing methods, requiring labeled data, are ineffective against new botnets. To address this issue, we propose Domain2Vec, a metric learning-based approach that can explore new botnets. Domain2Vec integrates a framework of metric learning, which uses individual domains from known botnets for categorization of unknown botnet domains. The training involves an attention-based encoder, and it includes a constraint to ensure that samples with the same labels are closer in the embedding space. The categorization uses the encoder to project domain names into appropriate representations (numerical vectors), even for domains from new botnets. Finally, Domain2Vec uses numerical vectors to explore botnets. Experiments showed that Domain2Vec performs well on domain retrieval and clustering tasks without labeled data, outperforming the state of the art by 13% and 100%, respectively. Real-world tests demonstrate that Domain2Vec can effectively identify unreported malicious domains and monitor botnet activities.

1. INTRODUCTION

Botnets, complex networks of compromised devices orchestrated by cybercriminals [1, 2], have emerged as a significant cybersecurity threat in modern times. These networks serve as platforms for various malicious activities, including malware injection [3, 4], spam distribution [5], distributed denial of service attacks [6], and phishing [7]. In 2021, the impact of botnet activities was stark, with over 1.6 million computing devices compromised, leading to considerable financial losses amounting to billions of dollars globally [8]. The broad scope and severity of these attacks underscore the ongoing challenge in countering botnets—a challenge that persists despite advances in cybersecurity technologies.

Extensive research has been devoted to behavioral detection of botnets, i.e. identifying benign or malicious domain names [9, 10]. Yet, the binary classification of domain names is insufficient to cope with botnet threats fully. This is because modern botnets commonly leverage the domain fluxing technique to escape network monitoring and avoid Internet takedown [2, 11]. With domain fluxing, the bots (clients) injected with malware codes can generate a large number of domain names, i.e. fast-flux domains. Those botnet domains serve as the potential rendezvous stubs for the command-and-control (C&C) connections between the bots (clients) and the botmaster (server). Domain fluxing is very cost-effective to botmasters because it merely requires a few botnet domains registered where the bots can receive the instructions and fulfill the attacks. In this case, rough behavioral detection on real-world networks, where the domains are divided into benign and malicious classes, cannot relieve the botnet threats thoroughly because a few fugitives of fast-flux domains could still maintain the C&C channels for attacks.

Given the escalating complexity and prevalence of botnet threats, traditional approaches that merely identify and filter all malicious botnet domains are proving to be inadequate. Consequently, researchers and cybersecurity vendors are now shifting their focus toward the rapid categorization of botnet domains to facilitate the issuance of timely security alerts within actual network environments [12, 13]. This strategic shift is crucial as it enables network asset owners to proactively prepare for impending botnet attacks by deploying targeted security patches and configuring robust firewalls customized to the specific threat [14]. More importantly, the meticulous, fine-grained categorization of botnet domains into specific categories aligned with their botnet types allows for more accurate tracking and effective mitigation of these threats [15]. This approach not only enhances the effectiveness of the security measures but also optimizes the allocation of defensive resources. By precisely identifying the nature and origin of the threat, this method fosters a more strategic and informed response, thereby significantly reducing the potential damage these botnets [9, 16] can cause.

In botnet domain categorization, several research efforts have been made to broaden our insights of domain fluxing mechanisms in botnets. A common solution framework is to collect the botnet domains from reverse engineering or bot-infected hosts as train sets, and exploit the supervised classifiers, e.g. CNN [17], RNN [18], and LSTM [19], for divisions of botnet domains. In addition, some feature-based methods [9, 20] for behavioral detection can be extended for the multi-classification of botnets. To perform well, the above methods require labeled training data covering at least partial domains created by the botnets. In other words, they are less effective if a botnet domain family is never seen before and no labeled data are available [21]. This requirement greatly limits the generality of these methods since botnet threats grow along with the arms race between cybercriminals and defenders, making the complete collection of malicious fast-flux domains impossible.

Received: January 23, 2024. **Revised:** May 5, 2024. **Accepted:** July 3, 2024

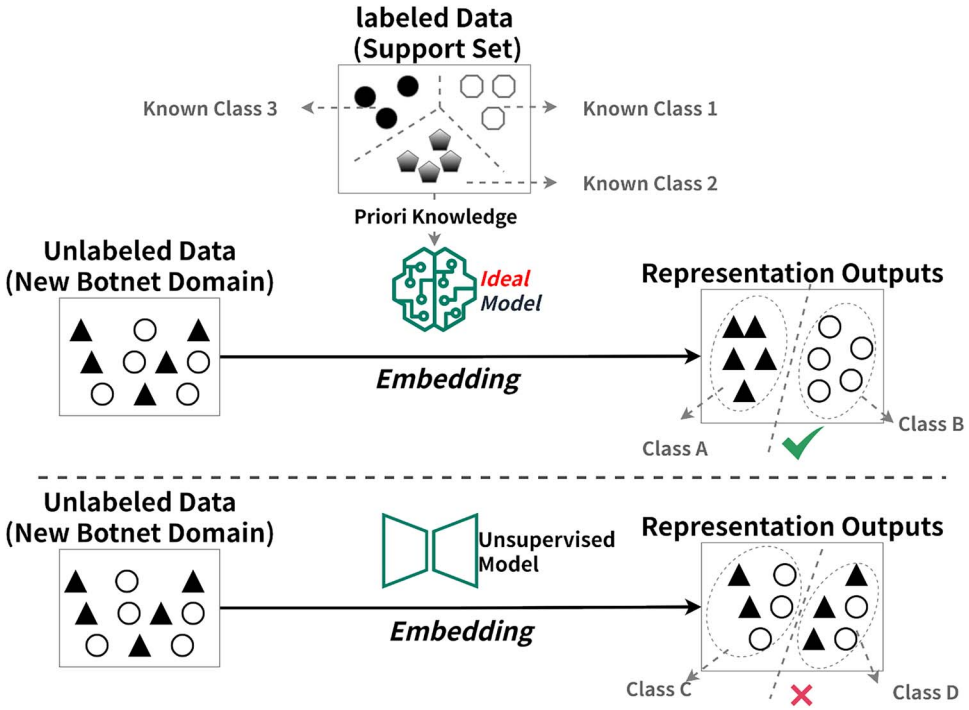


Figure 1. The illustration of weaknesses in existing methods and an ideal model for a solution; in the ideal model, the prior knowledge is utilized, and new botnet domains can be embedded into a proper representation to achieve an appropriate categorization, i.e. classes *A* and *B*, while the existing unsupervised method without the prior knowledge would perform weakly, i.e. classes *C* and *D*.

To address the challenges associated with effectively categorizing botnet domains, a straightforward approach involves utilizing unsupervised learning. An innovative method in this domain is the Helix model [22], which employs a spatio-temporal deep neural network autoencoder (An autoencoder is a type of neural network designed to learn efficient representations of data. Its function is to compress input data into a more compact encoding.) to transform domain names into numerical vectors. These vectors are subsequently clustered using algorithms such as KMeans to identify distinct groups of botnet domains. However, the efficacy of these unsupervised approaches often falls short, particularly when dealing with the complex nature of fast-flux botnet domains. As detailed in Section 5, these domains exhibit highly irregular behaviors and are distributed across botnets in a manner that makes traditional clustering challenging [23]. Without the integration of prior knowledge or advanced feature engineering, embedding these domains into a coherent feature space that accurately reflects their underlying associations proves difficult. This challenge is visually illustrated in the bottom right of Fig. 1, highlighting the involved complexities. Consequently, achieving a fine-grained categorization of such diversified botnet domains without labeled data remains a substantial challenge and an ongoing area of research within real-world network security contexts.

Our motivation to develop a novel botnet domain categorization method originates from the limitations of existing approaches, which primarily rely on labeled data. To address this issue, we introduce Domain2Vec, a metric learning-based model designed to generalize effectively across both known and previously unseen botnet domains by embedding them into vector representations. Metric learning refers to a category of machine learning algorithms designed to derive an optimal distance metric from provided data. This method not only capitalizes on the rich information available from labeled datasets but also pioneers the identification of new, previously unrecognized botnet families. Domain2Vec employs a feature extraction framework (an encoder) that captures complex patterns in known botnet domains to predict and categorize unknown malicious domains effectively, as illustrated in Fig. 1. Our model utilizes a multi-head attention-based encoder (A technique employed in deep learning architectures, enabling the model to concurrently attend to multiple aspects of the data during processing.), detailed in Section 4.3, adept at identifying subtle character patterns crucial for recognizing botnet domains. Furthermore, our approach introduces a novel metric learning constraint, detailed in Section 3.2, which enhances the model's ability to generalize across varied domain types. The refined encoder acts as a robust feature extractor, transforming botnet domains into precise and interpretable numerical vectors. These vectors are subsequently employed in advanced classification, retrieval, and clustering tasks, facilitating the effective categorization of emerging botnet domains, thereby substantially broadening the scope and utility of our method.

Contribution. Our contributions can be summarized as follows:

- We propose a deep metric learning model to validly convert the malicious botnet domains into representative vectors. To the best of our knowledge, our model is the first to apply deep metric learning to the fine-grained categorization of botnet domains.
- Based on the model, we implement Domain2Vec which can be used for botnet domain retrieval and clustering even without corresponding labeled data. Experiments carried out on benchmarks show that Domain2Vec not only outperforms traditional supervised multi-classification tasks but also achieves state-of-the-art performance on the categorization of never-before-seen botnet domains.

- Furthermore, real-world tests on DNS infrastructures indicate that our approach can uncover never-before-seen domains and track new botnets in the wild.

2. RELATED WORK

The research on botnet domain categorization roughly falls into two broad categories: DNS traffic-based and bare domain-based methods.

2.1. DNS traffic-based approach

In the DNS traffic-based methods, the domain's metadata from raw DNS traffic, a.k.a. side information, are captured for analysis of botnet characteristics. *Exposure* [24], *Winning with DNS Failures* [25], *NetFlow* [26], and *Phoenix* [11] are representative methods in this category. Specifically, *Exposure* [24] is a malicious DNS traffic monitoring system, which exploits the information from possibly infected hosts for discovering domain names used in phishing, malicious code, or botnets. Yadav and Reddy [25] focused on both successful DNS queries and DNS failures (e.g. NXDomain) and proposed a method called *Winning with DNS Failures*, which analyzes the information regarding IPs and domain names to discover a set of potential C&C server IP addresses. Grill *et al.* [26] classify normal DNS traffic and botnet DNS queries using information acquired by *NetFlow*, e.g. the IP addresses, time stamps, port numbers, byte counters, and packet counters of DNS queries. Similarly, *Phoenix* [11] filters out the benign domains based on the linguistic features and statistical features of DNS traffic and then clusters the domains and IPs as either botnet-related or benign.

In a nutshell, the above DNS traffic-based methods require extensive tracking of DNS traffic, which may cause the serious concern of privacy leakage [10] and expensive deployment costs. For this reason, they are excluded from the baselines, and we mainly focus on the other type of methods, i.e. bare domain-based methods.

2.2. Bare domain-based approach

In this group of methods, the traffic-independent domain names, a.k.a. bare domains, are utilized as inputs of the botnet domain categorization. Those bare domain-based methods focus on learning the character-level features with either manual feature engineering or deep neural networks. Then the problem of categorizing botnet domains is solved with classification or clustering, depending on whether or not labeled domain classes are available.

FANCI [9] and HAGDector [27] adopted a series of manually selected lexical features and used supervised learning classifiers, e.g. random forest (RF) and support vector machine (SVM), to classify benign and botnet domains. Since manually selecting features is tedious, a series of deep learning-based methods were proposed and achieved better accuracy and false positive rate results in the classification and behavioral detection of bare domains. Generally, those deep learning-based methods adopt the convolutional neural network (CNN) [17], recurrent neural network (RNN) [19], or a hybrid architecture [18, 28, 29] as the backbone network in the model and can adjust the activation function of the output layers according to the task types, e.g. *softmax* for botnet domain classification and *sigmoid* for domain behavioral detection. Nevertheless, the above methods fail to cope with the categorization of new botnet domains without labeled data. Recently, *Helix* [22] utilized an autoencoder architecture for unsupervised clustering of botnet domains without requiring labeled data. Its performance, however, is weak.

Table 1. Summary of related works on botnet domain categorization; some traffic-based methods require the captured DNS traffic and additional side information, which being used as baselines might not be fair for character-level (a.k.a. bare domain-based) methods, e.g. FANCI

Name	Δ	Detail	Comments
Exposure	x	Raw DNS Queries	Privacy leakage
Win Fail	x	IPs and NXDomains	
NetFlow	x	NetFlow Records	
Phoenix	x	Features on DNS traffic	
Endgame	✓	LSTM Classifier	Lacking extensibility
Invincea	✓	Parallel CNN Classifier	
CMU	✓	Bidirect LSTM Classifier	
MIT	✓	LSTM-CNN Classifier	
NYU	✓	Stacked CNN Classifier	
FANCI	✓	Manual Lexical Features	Weak performance
HAGDector	✓	Mixed Manual Lexical Features	
Helix	✓	Unsupervised Autoencoder	
Ours	-	Metric Learning Framework	Botnet domain representation without data annotation

Δ: As Baseline

In summary, existing methods are either fronted with privacy leakage problems or perform weak in unsupervised conditions, as listed in Table 1. Thus, there is an urgent demand and a challenging research gap in categorizing botnet domains. This work is dedicated to filling this gap.

3. PRELIMINARIES

To help better understand the paper, we first define the problem to be solved. Then, we briefly introduce the metric learning theory adopted in Domain2Vec. Finally, we describe the categories of domain fluxing techniques and their characteristics.

3.1. Problem statement

The problem of categorizing new botnet domains could be abstracted as a task of knowledge transfer, which assumes the availability of two datasets: a labeled support dataset D^l and a query dataset D^u, containing the label (class) set of C^l and C^u, respectively, with the setting of $C^u \cap C^l = \emptyset$. Therefore, the goal is to associate the domain data in D^u with the class set C^u, leveraging the knowledge from D^l, which is different from the existing tasks of supervised classification [18, 19] and unsupervised clustering [22]. Note that the class set C^u of the query set might not be known in real-world scenarios, and we include it here to help understand the problem formulation.

Difference from behavioral detection. Behavioral detection [9, 10] aims to divide the bare domains or non-exist domains (NXDomains) into benign domains and fast-flux (malicious) ones, i.e. binary classification. As stated in Section 1, behavioral classification is not enough to deal with botnet threats.

3.2. Metric learning

For the knowledge transfer, Domain2Vec adopts a metric learning framework rather than pure classifiers, where the outputs of our models are not specific categories/classes but the representation vectors of input data, which can be validly used for input of existing algorithms (e.g. RF and K-Means). In other words, we focus on adjusting inter-class and intra-class distances of botnet domain

Table 2. Categories of domain generating algorithms.

Pattern	Example	Comment

Arithmetic	Botnet *pushdo*
	`qiclafux.kz`
	`lebusobafwyo.kz`
	`xerrucehel.kz`
		Combination of arithmetic operations, e.g. multiply, divide, and modulo. Unfixed length and alphabets ∈ {0-9*a*-z}

Hash	Botnet *ud2*
	`eb616ba683e7dd4bba24.info`
	`3122b1948d87ee39af71.info`
	`b5daea4485986519b4d4.info`
		Hashing values on given seeds, usually random numbers or timestamps. Fixed length and alphabets ∈ {0-9*a*-*f*}

Wordlist	Botnet *suppobox*
	`rightneither.net`
	`christmasbotwright.ru`
		Samples from a wordlist (dictionary), e.g. {right, nethier, christmas, bot, wright}

Permutation	Botnet *banjori*
	`andersensinaix.com`
	`xjsrrsensinaix.com`
		Permutation of given sequences, i.e. common suffix `rsensinaix`

embeddings to project the datasets into appropriate metric space where the samples with the same label are smaller in distance than those with different labels. Generally, the advantages of metric learning are as follows:

- **Avoidance of overfitting.** The distance metric not only measures the labeled botnet domains but also provides a new data representation with interpretability and discriminational capacity using the similarity between samples.
- **Flexibility for dynamic threats.** As reported, cybercriminals have been proposing new botnets to resist Internet monitoring. Hence, the number of botnet domain families is constantly growing. The outputs of the metric learning framework do not depend on the number of classes and only need to reflect the similarity between the input data. This is sharply different from pure classifier models, whose outputs heavily rely on the total number of classes. In other words, Domain2Vec can be easily expanded for unknown botnets with simple tuning, while the classifier models need to be retrained using entire known datasets whenever we detect new botnet domain families.

More details of the metric learning framework are presented in Section 4.3.

3.3. Domain fluxing technique

Domain fluxing is a cost-effective technique for botnet-related cybercriminals. Early botnets use hardcoded domains or IP addresses for the C&C channels. In this case, defenders can easily block the malicious domains or IP addresses by blacklisting and taking down the C&C servers. To avoid this problem, domain fluxing can keep a malicious botnet in operation by constantly changing the domain name of the botnet owner's C&C server [30]. Since the botnet domains constantly change, it is difficult for defenders to blacklist botnet domains. We need an automatic method to categorize botnet domains.

Domain fluxing uses Domain Generating Algorithms (DGA) to generate domain names, e.g. simply drawing characters uniformly at random or imitating character or word distributions of real domains [12, 31]. The DGA can be generally classified in Table 2.

4. DESIGN OF Domain2Vec

We first present the system overview of Domain2Vec and then describe its key technical components: sample selection, network structure, and metric loss function. Finally, we discuss the application scenarios of Domain2Vec.

4.1. System overview

Figure 2 illustrates the main workflow of Domain2Vec. In the offline phase, it first selects a proper number of samples from the labeled datasets of known botnet domains, to generate the balanced data inputs. Domain2Vec will exploit those balanced data to train a neural network model for learning appropriate similarity metrics. Then, the well-trained model is utilized as a feature encoder for efficient representation of botnet domains and those numerical vectors (embeddings) could be used for classification, retrieval, or clustering tasks, depending on specific online use scenarios.

4.2. Sample selection

Direct utilization of all data from label datasets would result in the challenge of unbalanced prediction. Because it is not rare that there are only a few domain names in some botnet families (classes) of public data, learning on the data inputs derived from those entire imbalanced domain datasets may result in weak performance, i.e. the classes with sufficient data may have higher accuracy, but classes with fewer data may have lower accuracy. To avoid the problems, we oversample the domains based on botnet families (classes) to generate balanced data inputs.

4.3. Network structure

In this section, we focus on learning the appropriate representations of botnet domain inputs using a multi-head attention-based neural network in Domain2Vec. To be brief, the employment of attention mechanisms is to uncover which positions and alphabets in the domain names have conclusive contributions to their attributions.

The complete architecture of Domain2Vec's network structure can be found in Fig. 3. Next, we describe the implementation details of its feature encoder G.

- Input layer. This layer receives the byte streams of entire domains with a maximum length constraint s ($s = 256$ is the max number of tokens in this work according to the maximum length of domains [32]). Thus, zero padding is added to those vector tails of short domains.
- Character embedding layer. Each character represented with the integers by the Input layer would be compressed as the m-dimensional (32 in Domain2Vec) numerical vectors. Thus, those character embeddings are concatenated to produce a $s \times m$ matrix for a given domain name.
- Positional encoding. Considering the position-insensitive multi-head attention architecture, we introduce the absolute positional encoding for the employment of positional information in domain names according to the following formulation:

$$\mathbf{p}_{pos,2i} = \sin(pos/10000^{2i/m}) \tag{1}$$

$$\mathbf{p}_{pos,2i+1} = \cos(pos/10000^{2i/m}), \tag{2}$$

where pos denotes the encoded position and i is the embedding dimension index in the positional encoding tensor.

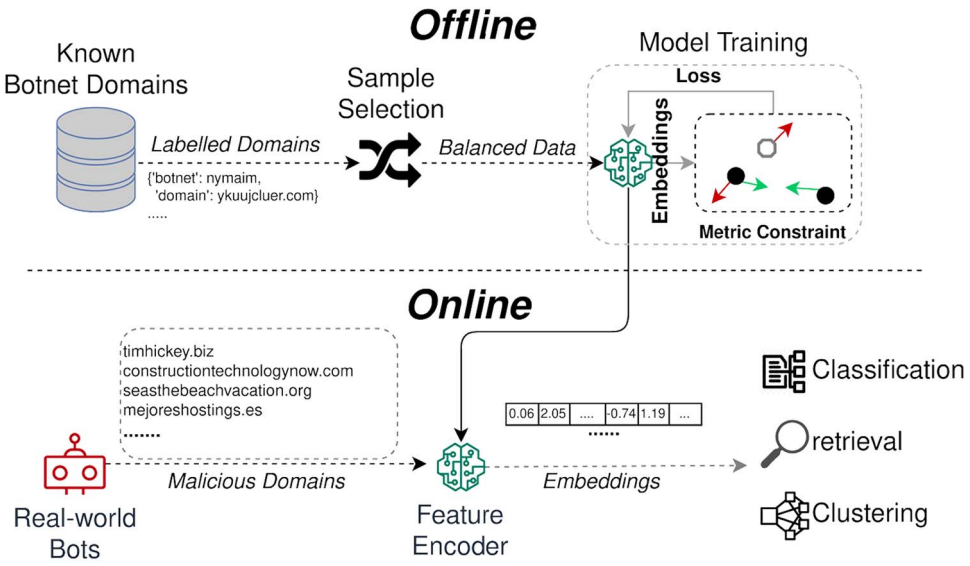


Figure 2. The workflow of Domain2Vec.

- Attention mechanism. Direct utilization of domain embeddings above might suffer low performance of categorization because some specific positions and alphabets in the data inputs (e.g. `xx` in two beginning positions of domains `xx4b836033.ru` for botnet *xxhex*) should be focused on due to their most pertinent information to accomplishing the task, while others should be given less attention to avoid additional noise. To this end, the multi-head attention architecture is leveraged for realizing the automatic weighting of the character embeddings via simple multiply operations. In this study, 8-head attention mechanisms are utilized to improve the metric learning of botnet domain empirically.
- Normalization. For better generalization accuracy, layer-wise normalization is utilized in the feature encoder of Domain2Vec. Intuitively, normalization provides an appropriate opportunity for character embeddings even if they are lightly weighted by attention scores. In this way, we can prevent the possible overfitting caused by biased attention and make the training faster.

Some deep metric learning frameworks explicitly employ the shared-weight mechanisms of siamese networks [33] to make sure the generalizability and consistency of representation outputs. In Domain2Vec, we implement it because the data will be embedded by the same weights in a batch until the gradient updates. This design decision is due to the following considerations: (1) low complexity: compared with multiple backbones in siamese networks, a single backbone network can significantly reduce the size of the entire model; (2) flexibility: for the training of siamese networks, the complex selection of data inputs is necessary, i.e. apart from the balanced datasets, the fine-crafted data inputs of all the backbones are additionally required. In other words, Domain2Vec is adaptive to the various domain datasets.

4.4. Loss function

Domain2Vec aims to learn a metric for appropriate representations of irregular botnet domains, as introduced in Section 4.2. In this case, the traditional loss function frequently used in existing supervised methods, e.g. categorical cross entropy [19, 22], is infeasible. Hence, we need to define a new loss function.

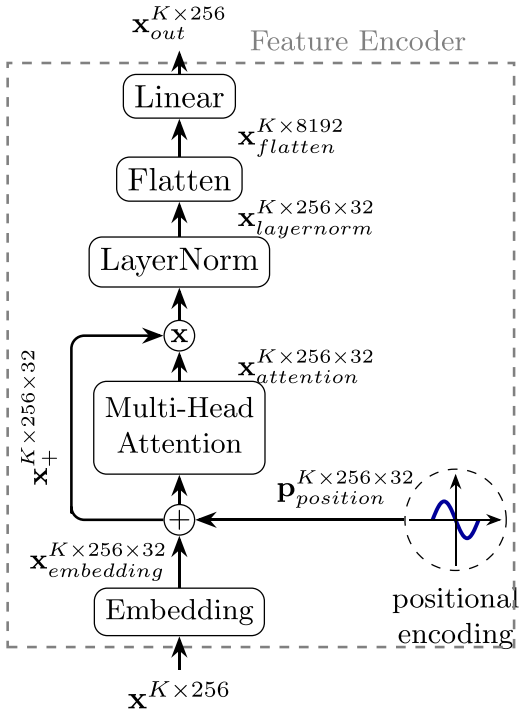


Figure 3. The neural network model in Domain2Vec; note that K denotes the number of input domains in a batch.

Our loss function considers the straightforward similarity/distance constraint, i.e. the similarity between samples from the same domain family should be bigger than the similarity between samples from different domain families. Specifically, given a data batch $\mathbf{x}$ and corresponding labels $\mathbf{y}$, the outputs of one data input x_i could be presented as $G_\theta(x_i)$, where $x_i \in \mathbf{x}$, $y_i \in \mathbf{y}$, and θ is the parameters of feature encoder G.

For one data x and its label y in the batch $\mathbf{x}$, its similarity from any data representation x_δ could be roughly estimated by $G_\theta(x)^\intercal G_\theta(x_\delta), \forall x_\delta \in \mathbf{x}$. However, merely the inner product on representation outputs could not reflect the real similarity due to the following consideration: the embeddings of two domains A, B from the same class usually have a smaller inner product (less similar)

than the embeddings of two domain C, D from different classes i.e. $G_\theta(x_A)^\intercal G_\theta(x_B) < G_\theta(x_C)^\intercal G_\theta(x_D), y_A = y_B, y_C \neq y_D$, if the former class is complex and hard. In other words, the domains from one irregular botnet family would be always far away from each other in the metric space. To address this, a smooth normalization method is necessary and we thus adopt the *softmax* function for illustration of (x, x_δ) similarity:

$$s_x(x_\delta) = \frac{\exp(G_\theta(x)^\intercal G_\theta(x_\delta))}{\sum_{x_i \in \mathbf{x}} \exp(G_\theta(x)^\intercal G_\theta(x_i))}, s_x(x_\delta) \in (0, 1)' \tag{3}$$

Intuitively, $s_x(x_\delta)$ represents the confidence that x_δ belongs to a similar domain (the positive in the same class) as x in the data batch $\mathbf{x}$. The above similarity constraint could thereby be presented as

$$s_x(x_+) > s_x(x_-) > 0 \quad \forall x_+, x_- \in \mathbf{x}, y = y_+ \neq y_- \tag{4}$$

Nevertheless, the above hard constraint may be too restrictive since it requires the inequality holds for $\forall x_+, x_- \in \mathbf{x}$. We thus relax it into a soft one and rewrite the objective as

$$\arg\max_\theta \sum_{y_+ = y} \sum_{y_- \neq y} \frac{s_x(x_+)}{s_x(x_-)} \tag{5}$$

Assuming that there are K_+ positives (including x itself) of the anchor x in the batch $\mathbf{x}$ consisting of K domain names, the soft constraint could be

$$\arg\max_\theta \sum_{y_+ = y} \frac{1}{K_+} s_x(x_+) - \sum_{x_{\prime} \in \mathbf{x}} \frac{1}{K} s_x(x_{\prime}) \tag{6}$$

Note that $\sum_{x_{\prime} \in \mathbf{x}} \frac{1}{K} s_x(x_{\prime}) = \frac{1}{K}$ because $\sum_{x_{\prime} \in \mathbf{x}} s_x(x_{\prime}) = 1$, and we thus rewrite the objective as unified format of loss:

$$\arg\min_\theta -\sum_{x_\delta \in \mathbf{x}} p_x(x_\delta) s_x(x_\delta), \quad p_x(x_\delta) = \begin{cases} \frac{1}{K_+}, & y_\delta = y \\ 0, & y_\delta \neq y \end{cases} \tag{7}$$

However, the loss is always a negative value that cannot be directly utilized for training the backbone network. To this end, we replace $s_x(x_\delta)$ with $\log s_x(x_\delta)$ and the loss function on the entire batch could be presented as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\frac{1}{K} \sum_{x \in \mathbf{x}} \sum_{x_\delta \in \mathbf{x}} p_x(x_\delta) \log s_x(x_\delta) \tag{8}$$

Improvement for hard samples. In the categorization of diversified botnet domains, some are easily embedded into a metric space (e.g. domains from hash-based and permutation-based botnets), while the others from those irregular botnets, a.k.a. hard samples, are difficult to correctly predict the labels (botnet families) even with the balanced data inputs. The underlying reason is that the model considers the contribution of all samples equally, which means that the vast number of easy samples would overwhelm it and thus make it perform weakly on those irregular botnet domains (hard samples). To this end, the dynamical scaling factor $(1 - s_x(x_\delta))^\gamma$ is adopted to automatically down-weight the loss assigned to well-categorized samples, and rapidly focus the model on the hard samples as shown in the following:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\frac{1}{K} \sum_{x \in \mathbf{x}} \sum_{x_\delta \in \mathbf{x}} p_x(x_\delta)(1 - s_x(x_\delta))^\gamma \log s_x(x_\delta)$$

where $\gamma > 0$ is a hyperparameter. Intuitively, for those positives (domains in the same class) $p_x(x_\delta) > 0$, if the confidence $s_x(x_\delta) \approx 1$, which indicates that the samples with same classes are similar in the representations (easy samples), the scaling factor will reduce their contributions to the loss with $(1 - s_x(x_\delta))^\gamma \approx 0$. In contrast, those hard samples $s_x(x_\delta) << 1$ with the low-confidence similarities would be given more weights for efficient metric learning.

4.5. Use scenarios

After training, Domain2Vec works online and requires no additional information but the individual domain names. Thus, Domain2Vec is a versatile and flexible system applicable in various scenarios. Below, we briefly introduce the deployment of Domain2Vec (i.e. the online phase in Fig. 2) on real-world networks.

For privacy concerns, most ISPs strictly restrict the access of DNS-related traffic, e.g. source IPs, time stamps, DNS response codes, because otherwise the ulterior can easily infer the sensitive information of the customers, e.g. browsing history. In this case, bare domain names are the only available information. Nevertheless, this information is enough for Domain2Vec to work well. Domain2Vec embeds the bare domain names into appropriate numerical vectors with the powerful feature encoder shown in the online phase of Fig. 2, and uses those embeddings as the inputs of classification, retrieval or clustering tasks. Furthermore, Domain2Vec's embedding function can be used *as a service*, for example, accessible via an API or a web service usable by security software or tools.

5. EVALUATION

We compare the performance of Domain2Vec and other existing methods, including FANCI [9], HAGDector [27], Endgame [19], Invincea [17], CMU model [28], MIT model [18], and NYU model [29], with well-known public datasets [31, 34]. Furthermore, we field-test Domain2Vec on real-world DNS records to find new botnets.

5.1. Experiment setup

The data sources used in the paper are collected from the 360 DGA repository, DGArchive service, and Rapid7 Foward DNS data.

- **360 DGA repository.** This is massive passive DNS data of 360 DNS services and malware samples in real-time [34], containing about 1 million botnet domains across 64 families until 31 December 2021.
- **DGArchive service.** DGArchive service [31] is a public service providing the malicious domains generated based on reverse-engineered and known seeds. At the time of our experiment, DGArchive contains about 120 million botnet domains and 106 families.
- **Rapid7 Foward DNS dataset.** The Rapid7 Foward DNS dataset [35] contains the response results to DNS requests from Rapid7's DNS infrastructure, which will be used for the real-world field test of Domain2Vec.

To conduct a comprehensive evaluation of benchmarks, we divide the categorization of botnet domains into three tasks:

Table 3. Characteristics of botnet domain datasets.

	Name	Botnet	Number
Classification	ξ_1^u, ζ_1^u	10	10K
	ξ_2^u, ζ_2^u	30	30K
	ξ_3^u, ζ_3^u	50	50K
Retrieval	ξ_4^u, ζ_4^u	10	10K
	ξ_5^u, ζ_5^u	50	50K
Clustering	ξ_6^u, ζ_6^u	10	10K
	ξ_7^u, ζ_7^u	30	30K
	ξ_8^u, ζ_8^u	50	50K
Support Data	$\xi_1^l - \xi_8^l, \zeta_1^l - \zeta_8^l$	50	500K

Table 4. Classification accuracy on labeled datasets.

Approach	ξ_1^u	ξ_2^u	ξ_3^u	ζ_1^u	ζ_2^u	ζ_3^u
$\text{Helix}_{(\text{KNN})}$	0.898	0.787	0.754	0.926	0.833	0.789
$\text{FANCI}_{(\text{SVM})}$	0.725	0.523	0.434	0.657	0.455	0.364
$\text{FANCI}_{(\text{RF})}$	0.913	0.801	0.765	0.924	0.818	0.771
Endgame	0.871	0.826	0.815	0.846	0.799	0.785
CMU	0.859	0.789	0.776	0.825	0.767	0.746
NYU	0.961	0.913	0.897	0.952	0.916	0.897
Invincea	0.949	0.901	0.879	0.927	0.896	0.867
MIT	0.936	0.869	0.849	0.908	0.843	0.816
Embedding	0.948	0.893	0.874	0.927	0.885	0.859
Domain2Vec$^{\triangle}$	0.974	0.932	0.909	0.986	0.945	0.929
Domain2Vec*	0.980	0.946	0.929	0.988	0.960	0.946
Domain2Vec	**0.987**	**0.956**	**0.941**	**0.990**	**0.964**	**0.951**

$\triangle$: attention layer removal $\star$: scaling factor removal

classification, retrieval, and clustering. For all the tests, the entire datasets are divided into two parts, i.e. support sets $\xi_1^l - \xi_8^l, \zeta_1^l - \zeta_8^l$ of class sets $\mu_1^l - \mu_8^l, \nu_1^l - \nu_8^l$ and query sets $\xi_1^u - \xi_8^u, \zeta_1^u - \zeta_8^u$ of class sets $\mu_1^u - \mu_8^u, \nu_1^u - \nu_8^u$. Note that the class sets of support sets and query sets should be disjoint, i.e. $\mu_i^l \cap \mu_i^u = \emptyset, \nu_i^l \cap \nu_i^u = \emptyset, i \in [1, 8]$. And, the support sets are extracted from the *mixture* of DGArchive and 360 DGA repository to ensure that there are sufficient categories (classes $>$ 50), while the query sets are derived from two data sources ($\xi_1^u - \xi_8^u$ for DGArchive and $\zeta_1^u - \zeta_8^u$ for the 360 DGA repository) for the generality of experiments. Those support sets are only employed for offline training the Domain2Vec's feature encoders after the sample selection in Section 4.2, and then the trained models are utilized to convert the domains of corresponding query sets into vector representations for the online categorization scenarios. More details of datasets are presented in Table 3.

We highlight that Domain2Vec in real-world deployment can work without labeled data, i.e. retrieval and clustering on unlabeled query sets $\xi_4^u - \xi_8^u, \zeta_4^u - \zeta_8^u$. We test its performance on the classification task on labeled query sets $\xi_i^u, \zeta_i^u, i \in \{1, 2, 3\}$ purely for the comparison purpose with several supervised baselines that only work with labeled data, e.g. Invincea [17], CMU model [28], and MIT model [18].

5.2. Performance of classification

To investigate the categorization capability of our framework with labeled data, we conduct the classification tests on the supervised baselines. To avoid sampling biases from the class sets' difficulty, we repeat the classification experiments 50 times and take the average of the results for evaluation.

We implement the RF algorithm in Domain2Vec for this task. As shown in Table 4, most methods, e.g. the NYU model and Invincea model, achieve high performance on the classification task of label datasets. Meanwhile, the classification performance degrades generally for all the methods when the number of categories increases. Nevertheless, Domain2Vec outperforms all the baselines and still achieves 0.94–0.99 accuracy results even on the classification task of 50 classes. This demonstrates Domain2Vec's capacity and robustness in supervised categorization of domain names.

5.3. Performance of retrieval

The domain retrieval is to fulfill this scenarios, i.e. researchers and security teams could usually acquire a few new botnet domains using the injected hosts or honeypots. However, such few samples fail to be used for training a supervised classifier, where the domain retrieval is a practical choice. In retrieval tests, the query sets are embedded into representation vectors using Domain2Vec encoder and other baselines. Then, the retrieval scores could be calculated as $\frac{x}{k}$, i.e. precision@k (**P@k**), while the top-k nearest neighbors in query sets have x relevant items (domains in same classes) for a given embedding. As mentioned above, supervised classifier-based methods cannot learn on only few samples of the query class sets. Therefore, we exclude these approaches that only work with sufficient labeled domain classes from the baselines in the retrieval, except for FANCI and Helix. We also repeat the experiments for avoiding sampling biases and present the results in Table 5.

Almost all the methods achieve high performance on the botnet domain retrieval tasks. Nevertheless, domain can still take a leading position over the baselines and realize 89.0–98.2% of **P@k** which is a 13% improvement on average over the state of the art.

5.4. Performance of clustering

Domain clustering can address the botnet domain categorization without any labeled data, where most existing methods fail to classify the unlabeled domains due to their supervised classifier architectures. Therefore, we exclude these approaches that only work with labeled domain classes from the baselines in the clustering experiments, except for FANCI and Helix.

For fair comparison, we also repeat the experiments to avoid sampling biases and adopt the K-Means algorithm in Domain2Vec and other baselines for clustering tasks. As shown in Table 6, the categorization results of existing methods on unlabeled domain datasets are unsatisfying, where the baselines usually achieve 0.30–0.52 adjusted rand index (ARI) scores and suffer weak performance commonly on all the query sets, because FANCI and Helix do not have a mechanism to leverage prior knowledge from the support set. In contrast, Domain2Vec, benefitting from the prior knowledge from the support set, can realize the effective categorization of botnet domains and fulfill 0.64–0.94 ARI scores on all the unlabeled datasets, which is a 100% improvement on average over the existing works.

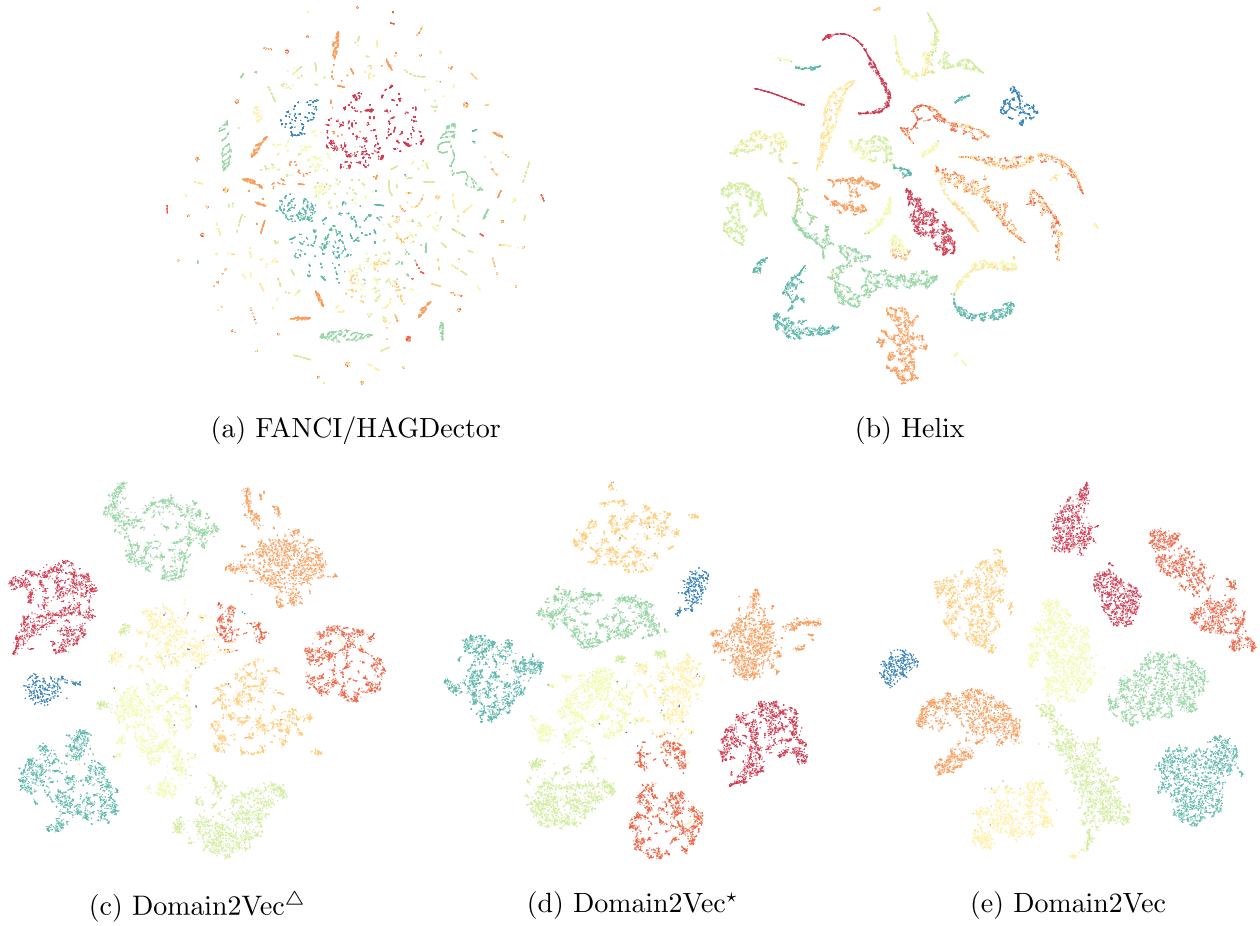
5.5. Ablation analysis

As stated in Section 4.3, the key components of Domain2Vec consist of the attention mechanism in feature encoder and the scaling factor $(1 - s_x(X_\delta))^\gamma$ in the loss function. To validate the architectures of Domain2Vec and learn the contribution of each component, we performed the ablation study: we evaluate two versions of Domain2Vec with the removal of the multi-head attention layer and the scaling factor, respectively. Then, those versions will be utilized as the additional baselines in the classification,

Table 5. Retrivel results (Precision@k %) on unlabeled datasets.

Approach	ξ_4^u				ξ_5^u				ζ_4^u				ζ_5^u			
	P@1	**P@2**	**P@4**	**P@8**	**P@1**	**P@2**	**P@4**	**P@8**	**P@1**	**P@2**	**P@4**	**P@8**	**P@1**	**P@2**	**P@4**	**P@8**
Helix	90.9	89.3	87.0	84.3	78.5	76.4	73.6	70.2	94.7	93.2	91.1	88.5	88.3	85.1	81.0	76.4
FANCI	88.3	87.8	87.0	85.8	72.5	71.9	70.8	69.4	89.4	88.6	87.2	85.2	75.7	74.3	72.5	70.1
Domain2Vec$^{\triangle}$	96.3	95.7	95.1	94.4	89.1	88.3	87.3	86.2	97.5	96.6	95.4	93.9	93.3	91.3	88.6	85.4
Domain2Vec*	97.0	96.9	96.7	96.4	90.8	90.4	90.0	89.7	97.9	97.4	96.5	95.4	94.4	92.9	90.8	88.2
Domain2Vec	**97.8**	**97.7**	**97.6**	**97.4**	**92.0**	**91.8**	**91.6**	**91.4**	**98.2**	**97.6**	**96.8**	**95.8**	**94.8**	**93.4**	**91.4**	**89.0**

△: attention layer removal ⋆: scaling factor removal



(a) FANCI/HAGDector (b) Helix

(c) Domain2Vec$^{\triangle}$ (d) Domain2Vec* (e) Domain2Vec

Figure 4. Scatter plots (*t-distributed stochastic neighbor embedding* reduced) of the baselines' and Domain2Vec's embeddings on the common data inputs (△ for attention layer removal and ⋆ for scaling factor removal).

Table 6. Clustering results (ARI) on unlabeled datasets.

Approach	ξ_6^u	ξ_7^u	ξ_8^u	ζ_6^u	ζ_7^u	ζ_8^u
Helix	0.518	0.365	0.299	0.491	0.418	0.381
FANCI	0.481	0.328	0.266	0.420	0.316	0.280
Domain2Vec$^{\triangle}$	0.821	0.743	0.646	0.617	0.532	0.504
Domain2Vec*	0.874	0.825	0.760	0.696	0.632	0.597
Domain2Vec	**0.940**	**0.909**	**0.861**	**0.747**	**0.676**	**0.642**

△: attention layer removal ⋆: scaling factor removal

retrieval, and clustering experiments as shown in Tables 4, 5, and 6.

The results illustrate that combination of multi-head attention and scaling factor used in Domain2Vec provides the best performance on all the tasks because the attention mechanism can focus on the critical positions and alphabets rather than equally view all the characters in a botnet domain and the scaling factor can effectively improve the resistance to those hard botnet domain families. In other words, removing some Domain2Vec components may leave out the useful (latent) features, resulting in weaker performance.

To trace Domain2Vec's high performance, we utilize the embeddings using different methods on the same data inputs of 10 typical botnets (e.g. *makloader* and *rovnix*) as shown in Fig. 4. Intuitively, the embeddings of Domain2Vec are naturally distinguishable where there are clear boundaries between classes. Nevertheless, the embeddings with FANCI and Helix are relatively irregular and weak in representation, where the data from different classes may overlap. In this case, the categorization algorithms could hardly perform on the embeddings. The phenomenon shown in Fig. 4 could also be observed with all

```
xwa.carrosserie-ammann.ch, pop.glarner-sommercup.ch
xas.hundeschule-marybo.ch, pop.baumgartner-motos.ch
xwa.naehatelier-samuel.ch, ctx.carrosserie-fuchs.ch
```

(a) New Botnet 1

```
00d0p000000dfmxuas.com, 00d1n000002duvuuae.com
00d1n0000021prpmac.com, 00d1n0000021prpuak.com
00d3000000001ayeau.com, 00d500000007mzqea2.com
```

(b) New Botnet 2

```
direwolf-03f4028373.*.com, direwolf-58f9664c9e.*.com
direwolf-91e4a46f10.*.com, direwolf-03f3a16a71.*.com
direwolf-58f4228f64.*.com, direwolf-58f4206042.*.com
```

(c) New Botnet 3

```
boats-kost.com, ekt-shirts.com, influ-life.com
paris-troc.com, poliy-pure.com, rosin-kuch.com
sking-nerd.com, stock-ware.com, tco-motors.com
the-fishle.com, think-city.com, truck-kort.com
```

(d) New Botnet 4

```
middelburg-ongediertebestrijding.nl,wageningen-ongediertebestrijding.nl
amstelveen-ongediertebestrijding.nl,leeuwarden-ongediertebestrijding.nl
roosendaal-ongediertebestrijding.nl,amersfoort-ongediertebestrijding.nl
```

(e) New Botnet 5

`bolzie.com`♦, `datohs.com`
`evarid.com`, `fitota.com`
`hubefy.com`, `redohs.com`
`robupi.com`, `unotor.com`
`vanodi.com`, `xupari.com`

(f) *virut*

`tradestrike.net`♦
`baechinger.net`
`leadtolove.net`
`wagenterprise.net`
`soulctcher.net`

(g) *suppobox*

Figure 5. Illustration of (partly) malware domains on unknown and known botnets discovered in the real-world DNS records; note that `bolzie.com` and `tradestrike.net` (♦) have been reported in the DGArchive service.

botnets in DGArchive and 360 repository. We omit these figures to save space.

5.6. Real-world test

To demonstrate that Domain2Vec can address botnet threats in real-world deployment, we apply Domain2Vec on domains from Rapid7's real network DNS traffic to mine the botnet domains and track the botnet actions.

We collected 1-month records from 1 February 2022 to 28 February 2022, from the Rapid7 Foward DNS dataset, which contains ∼7.7 billion DNS records, where we filtered out about 21 million non-existing domains (NXDomains). Then, we utilize a Domain2Vec encoder, which has learned on all known botnet domains, for representations of those NXDomains as feature vectors. And those vectors could be used for domain retrieval based on the known botnet domains to find the unrecorded malicious domain from known botnet families, and the others would be clustered for the discovery of never-before-seen botnet domain families. As the final result, we detected a number of suspicious new botnet families in the real-world data, and Fig. 5 illustrates several representative examples of discovered botnet domains. Meanwhile, we also found that there are still 30 000 botnet domains (18 botnet families), e.g. `tradestrike.net` in *suppobox*, that have been reported in the DGArchive service [31] and 360 DGA repository for a long time but still occurred in Rapid7's real-world DNS records for 2022. This finding confirmed that fighting botnets is still a long way to go.

Additionally, different from the direct combination of algorithm-generated subdomains and top-level domains, there are emerging botnet domains that take the private domains of some domain name registrars as their suffixes to further escape from monitoring (e.g. New Botnet 3 in Fig. 5), and we have obscured the suffixes with the wildcard (∗) to avoid conflict of interest. Therefore, it is suggested that the private domain registrars might block out those botnet domains timely and all the findings have been reported to the security vendors, related ISPs, and the DGArchive service.

6. CONCLUSION

In this work, we tackled the challenge of categorizing previously unobserved botnet domains, a critical task for monitoring botnet activities and mitigating associated threats. Botnets currently employ DGA to swiftly generate fast-flux domains, thereby evading detection. Precise categorization of these botnet domains is essential for the development of effective cybersecurity solutions aimed at mitigating botnet threats. However, existing methods, which rely on labeled data, fall short in effectively addressing newly emerging botnets. We introduced "Domain2Vec," a metric learning-based methodology tailored for the fine-grained categorization of malicious domains. By employing a multi-head attention-based encoder, Domain2Vec efficiently extracts features and transforms botnet domains into appropriate numerical vectors. Comprehensive evaluations on public datasets and real-world deployments demonstrate that Domain2Vec significantly outperforms existing state-of-the-art solutions, achieving improvements of 13% and 100% in the tasks of new domain retrieval and clustering, respectively. Furthermore, Domain2Vec has successfully detected previously unreported botnet domains, affirming its efficacy and applicability in real-world scenarios.

Domain2Vec demonstrates extensive applications, notably assisting cybersecurity firms in the early detection and prevention of potential attacks targeting government networks. This methodology supports organizations by meticulously monitoring the generation and transformation patterns of domain names, which facilitates the timely identification of emergent network threats. Furthermore, Domain2Vec capitalizes on its sophisticated feature extraction capabilities to offer Internet Service Providers (ISPs) enhanced traffic analysis and management. By scrutinizing anomalous domain access activities, the tool aids in fortifying the network security architectures of businesses. Additionally, it empowers legal authorities to augment law enforcement measures by tracking domains linked to illicit activities, ultimately aiding governments and organizations worldwide in their endeavors to monitor and counteract burgeoning botnet threats and other cybersecurity challenges. Future research will be directed toward enhancing the computational efficiency of the model to accommodate larger datasets and facilitate real-time analysis across diverse network environments. In response to the continuous evolution of botnet tactics, we intend to consistently refine and augment Domain2Vec to address new threats and challenges, thereby extending its capabilities to detect various types of cybersecurity threats.

FUNDING

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF1203001; in part by the National Natural Science Foundation of China under

Grants 62072465, 62172155, 62102425, and U22B2005; and in part by the Science and Technology Innovation Program of Hunan Province under Grant 2022RC3061.

DATA AVAILABILITY

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also form part of an ongoing study.

REFERENCES

1. Thomas M and Mohaisen A. Kindred domains: detecting and clustering botnet domains using dns traffic. In: *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*. Association for Computing Machinery, New York, NY, USA, pp. 707–712, 2014. https://doi.org/10.1145/2567948.2579359.
2. Antonakakis M, April T, Bailey M. *et al.* Understanding the mirai botnet. In: *Proc. USENIX Security*, pp. 1093–1110, 2017.
3. Bisio F, Saeli S, Lombardo P. *et al.* Real-time behavioral dga detection through machine learning. In: *Proc. ICCST*, pp. 1–6. IEEE, 2017. https://doi.org/10.1109/CCST.2017.8167790.
4. Mohaisen A and Alrawi O. Unveiling zeus: automated classification of malware samples. In: *Proceedings of the 22nd International Conference on World Wide Web*, pp. 829–832, 2013.
5. Yan Q, Zheng Y, Jiang T. *et al.* Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis. In: *Proc. INFOCOM*, pp. 316–324. IEEE, 2015. https://doi.org/10.1109/INFOCOM.2015.7218396.
6. Wang T-S, Lin C-S, and Lin H-T. Dga botnet detection utilizing social network analysis. In: *Proc. IS3C*, pp. 333–336. IEEE, 2016. https://doi.org/10.1109/IS3C.2016.93.
7. Drichel A, Drury V., von Brandt J. *et al.* Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs. In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, New York, NY, USA, pp. 1–12, 2021. https://doi.org/10.1145/3465481.3470111.
8. of Justice, D. Emotet botnet disrupted in international cyber operation. 2021.
9. Schüppen S, Teubert D, Herrmann P. *et al.* Fanci: feature-based automated nxdomain classification and intelligence. In: *Proc. USENIX Security*, pp. 1165–1181, 2018.
10. Li S, Huang T, Qin Z. *et al.* Domain generation algorithms detection through deep neural network and ensemble. In: *Proceedings of the International Conference on World Wide Web*, pp. 189–196, 2019. https://doi.org/10.1145/3308558.3316498.
11. Schiavoni S, Maggi F, Cavallaro L. *et al.* 2014. Phoenix: Dga-based botnet tracking and intelligence. In Dietrich S (ed). *Detection of Intrusions and Malware, and Vulnerability Assessment*. Cham: Springer International Publishing. 192–211. https://doi.org/10.1007/978-3-319-08509-8_11.
12. Drichel A, Faerber N and Meyer U. First step towards explainable dga multiclass classification. In: *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 1–13, 2021. https://doi.org/10.1145/3465481.3465749.
13. Wu L, Fu S, Luo Y. *et al.* A robust and lightweight privacy-preserving data aggregation scheme for smart grid. *IEEE Transactions on Dependable and Secure Computing* 2024;**21**:270–83. https://doi.org/10.1109/TDSC.2023.3252593.
14. Shi Y, Xi J, Hu D. *et al.* Raymvsnet++: learning ray-based 1d implicit fields for accurate multi-view stereo. *IEEE Trans. Pattern Anal. Mach. Intell.* 2023;**45**:13666–82. https://doi.org/10.1109/TPAMI.2023.3296163.
15. Zhang H, Gharaibeh M, Thanasoulas S. *et al.* Botdigger: Detecting dga bots in a single network. In: *Proc. TMA*, 2016.
16. Antonakakis M, Perdisci R, Nadji Y. *et al.* From throw-away traffic to bots: detecting the rise of dga-based malware. In: *Proc. USENIX Security*, pp. 491–506, 2012.
17. Saxe J, Berlin K. Expose: a character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys *ArXiv*, **abs/1702.08568**. 2017. https://doi.org/doi:10.48550/arXiv.1702.08568.
18. Vosoughi S, Vijayaraghavan P, and Roy D. Tweet2vec: learning tweet embeddings using character-level cnn-lstm encoder-decoder. In: *Proc. SIGIR*, pp. 1041–1044, 2016. https://doi.org/10.1145/2911451.2914762.
19. Woodbridge J, Anderson H, Ahuja A. *et al.* Predicting domain generation algorithms with long short-term memory networks. 2016. *ArXiv*, **abs/1611.00791**.
20. Drichel A, Meyer U, Schü"ppen S. *et al.* Making use of nxt to nothing: the effect of class imbalances on dga detection classifiers. In: *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 1–9, 2020. https://doi.org/10.1145/3407023.3409190.
21. Wu L, Fu S, Luo Y. *et al.* Sectcn: privacy-preserving short-term residential electrical load forecasting. *IEEE Trans. Industr. Inform.* 2024;**20**:2508–18. https://doi.org/10.1109/TII.2023.3292532.
22. Sidi L, Mirsky Y, Nadler A. *et al.* Helix: Dga domain embeddings for tracking and exploring botnets. In: *Proc. CIKM*, pp. 2741–2748, 2020. https://doi.org/10.1145/3340531.3416022.
23. Zhou T, Cai Z, Liu F. *et al.* In pursuit of beauty: aesthetic-aware and context-adaptive photo selection in crowdsensing. *IEEE Trans. Knowl. Data Eng.* 2023;**35**:9364–77. https://doi.org/10.1109/TKDE.2023.3237969.
24. Bilge L, Kirda E, Kruegel C. *et al.* Exposure: finding malicious domains using passive dns analysis. *Proc. NDSS* 2011;**16**:1–28. https://doi.org/10.1145/2584679.
25. Yadav S and Reddy A. Winning with dns failures: Strategies for faster botnet detection. In: *Proceedings of the International Conference on Security and Privacy in Communication Systems*, pp. 446–459. Springer, 2011.
26. Grill M, Nikolaev I, Valeros V. *et al.* Detecting dga malware using netflow. In: *Proc. IM*, pp. 1304–1309. IEEE, 2015. https://doi.org/10.1109/INM.2015.7140486.
27. Liang J, Chen S, Wei Z. *et al.* Hagdetector: heterogeneous dga domain name detection model. *Comput. Secur.*. Elsevier, 2022;**120**:102803. https://doi.org/10.1016/j.cose.2022.102803.
28. Dhingra B, Zhou Z, Fitzpatrick D. *et al.* Tweet2vec: Character-based distributed representations for social media. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 269–274, 2016. https://doi.org/doi:arXiv.1605.03481.
29. Zhang X, Zhao J, and LeCun Y. Character-level convolutional networks for text classification. In: *Proc. NIPS*, 28, 2015.
30. DeCarli L, Torres R, Modelo-Howard G. *et al.* Botnet protocol inference in the presence of encrypted traffic. In: *Proc. INFOCOM*, pp. 1–9. IEEE, 2017. https://doi.org/10.1109/INFOCOM.2017.8057064.
31. Plohmann D. Dgarchive. 2018.
32. Mockapetris P. Domain names - implementation and specification. STD13. RFC editor. 1987.

33. Chicco D. 2021. Siamese Neural Networks: An Overview. In Cartwright H (ed). *Artificial Neural Networks*. New York, NY: Springer US. https://doi.org/10.1007/978-1-0716-0826-5_3.
34. Network Security Research Lab at 360, N. S. R. L. Netlab dga project. 2021.
35. Sonar P. Rapid7 forward dns dataset. 2021.