EAI Endorsed Transactions
on Security and Safety
Research Article EAI.EU

deMSF: a Method for Detecting Malicious Server Flocks for Same Campaign

Yixin Li[1,*], Liming Wang[1], Jing Yang[1], Zhen Xu[1] and Xi Luo[2]

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]Cyberspace Institute of Advanced Technology, Guangzhou University

Abstract

Nowadays, cybercriminals tend to leverage dynamic malicious infrastructures with multiple servers to conduct attacks, such as malware distribution and control. Compared with a single server, employing multiple servers allows crimes to be more efficient and stealthy. As the necessary role infrastructures play, many approaches have been proposed to detect malicious servers. However, many existing methods typically target only on the individual server and therefore fail to reveal inter-server connections of an attack campaign.
In this paper, we propose a complementary system, deMSF, to identify server flocks, which are formed by infrastructures involved in the same malicious campaign. Our solution first acquires server flocks by mining relations of servers from both spatial and temporal dimensions. Further we extract the semantic vectors of servers based on word2vec and build a textCNN-based flocks classifier to recognize malicious flocks. We evaluate deMSF with real-world traffic collected from an ISP network. The result shows that it has a high precision of 99% with 90% recall.

Keywords: Malicious web infrastructure, Server flock, Word embedding, textCNN.

Received on 31 May 2020, accepted on 14 September 2020, published on 07 October 2020

doi: 10.4108/eai.21-6-2021.170236

* Corresponding author. Email: liyixin@iie.ac.cn

1 Introduction

Malicious web activity is still a major threat to Internet. Nowadays, cybercriminals build malicious web infrastructures to supply their crimes, which makes attacks complicated and diversified. Dark infrastructures today contain multiple servers (e.g., exploit servers, command & control servers, redirect servers, payment servers). Adversaries growingly combine these servers as the platform to spread malicious content, launch attacks and monetize from crimes. A typical example is malicious redirection, which leverages exploit servers to redirect visitor to another website.

Many approaches have been proposed to identify malicious servers. Most detection systems detect malicious webs by analyzing web content [5,11,16,24], identify malicious servers by building a reputation system for an individual server [3,4,9] or find popular techniques adversaries used to avoid evasion [18,25,30,32, 33]. Unfortunately, these works focus only on a single server which makes them lack the panoramic view of attacks. In addition, some servers may not easily be discovered by only analyzing the single server. Some works [2,15,19,27,34] notice the relation of servers by identify malicious redirections. Zhang et al. [34] indicate that malicious servers tend to be invisible and propose a method by analyzing redirections from visible to invisible server. Li et al. [15] leverage redirect-chains to build the topology of dark infrastructures and further recognize the dedicated malicious hosts. However, the collection of redirections is not easy while relations of malicious servers are various which not only limited to redirect.

Different from existing works, we focus on relations of servers given that increasingly attacks are conducted with multiple servers. We aim to identify servers involved in the same malicious activity, which we called a server flock, without relying on redirections. In particular, we find two features of server flocks to help distinguish flocks from DNS traffic. First, the completion of the attack requires a victim to access multiple servers continuously, in other words, servers of a flock tend to co-exist in the user's access list within an interval. Second, the flock used for crimes only serves certain victims,

which means that servers of a flock probably have stable clients.

Based on the above observations, in this paper, we propose a mechanism, deMSF, to detect malicious server flocks on a local network with only three fields: timestamp, clients and servers. We generate sever flocks in two steps: (a)we cluster servers within a client access list according to the timeline; (b)we extract final server flocks based on the similarity of clients. Inspired by a hypothesis that servers occur in the same contexts tend to be similar, we extract semantic vectors as features of servers based on word2vec and further design a convolution neural network based on textCNN to classify malicious flocks.

It should be noted that deMSF is a complementary approach to existing works. We believe that it can help detect servers that may be ignored by only analyzing a single server. In addition, it helps describing relations of servers within a malicious activity.

In summary, the contributions can be described as follows:

- We present a system, deMSF, to detect malicious campaigns by recognizing malicious server flocks. Focusing on flocks rather than an individual server makes deMSF be capable of revealing the relation of malicious servers.
- We design a two-step method to discover malicious flocks. In the first step, we generate flocks by clustering servers from both sequential and spatial dimensions. In the second step, we extract semantic vectors of servers and design a convolutional neural network to classify flocks based on these vectors.
- We evaluate the effectiveness of deMSF with real-world data collected from an ISP network, and the result demonstrates that deMFS performs well in discovering associated servers involved in malicious campaigns.

The rest of this paper is organized as follows: section 2 introduces the background knowledge of our work. Section 3 describes the critical components of deMSF. Then we evaluate the effectiveness of deMSF in section 4. Section 5 presents the related works, and section 6 is our conclusion.

2 Background

Machine learning has been widely used in many fields and gets significant advantages. Our work leverages machine learning to describe semantic features of servers and identify malicious flocks. In this section, we describe the related machine learning techniques employed in our system.

2.1 word2vec

Word2vec, proposed by Mikolov [20] in 2013, is one of the most widely used techniques for learning high-quality word vectors from huge data sets with billions of words. The resulting vectors can reflect subtle semantic relationships between words, for example, $vector(King)-vector(Man)+vector(Woman)$ results in a vector that is closest to the vector representation of the word Queen [22].

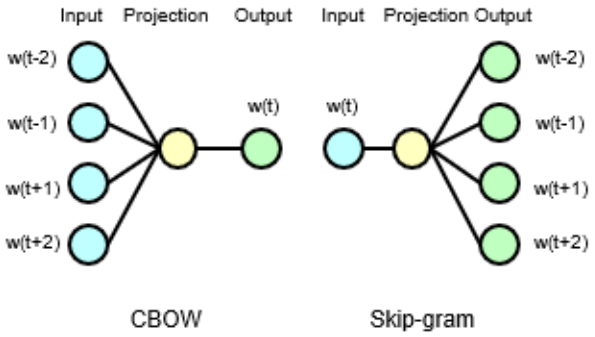

Fig.1. Two models of word2vec

Word2vec takes text corpus as input and generates word vectors. It includes two learning models, Continuous Bag of Words (CBOW) and Skip-gram. As shown in Figure 1, both are simple neural network model with one hidden layer. The former predicts the word given its context, while the latter predicts the context given a word. Compared to the one-hot encoder, word2vec generates dense vectors. Another significant advantage of word2vec is that words with similar meanings will be mapped to similar positions in the vector space.

2.2 textCNN

Convolutional neural networks (CNNs) are a specialized kind of neural network for processing data that has a known, grid-like topology [8]. CNNs are originally used in computer vision [13], while in recent years, they have been found to perform well for NLP. In 2014, Kim proposed a network named textCNN [12] for sentence-level classification tasks with pre-trained word vectors. As shown in Figure 2, textCNN is a simply neural network with an input layer, an output layer, a convolution layer and a max-pooling layer. It takes texts as input and usually leverages word embedding to increase performance.

In this paper, we design a network based on textCNN for our task. This model achieves superlative performance in malicious flocks detection.

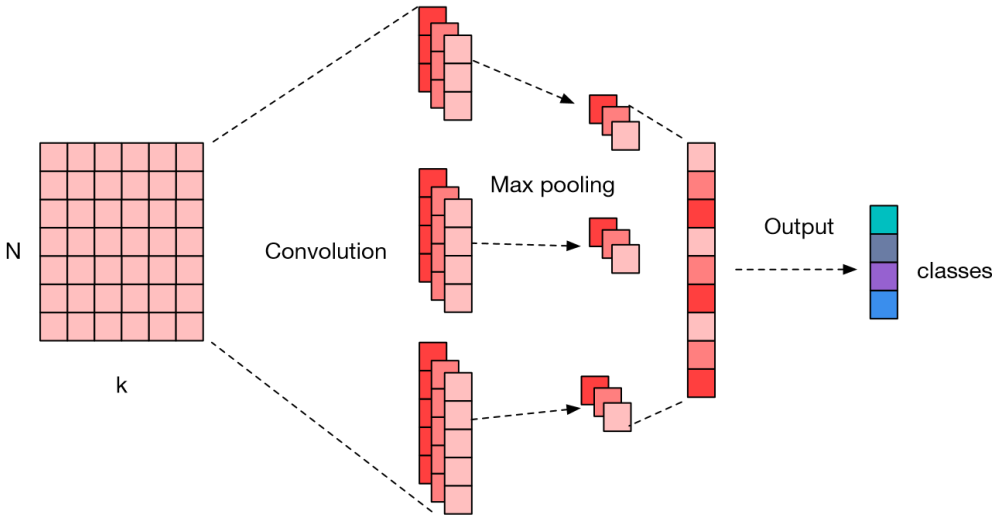

Fig.2. textCNN

3 System Description

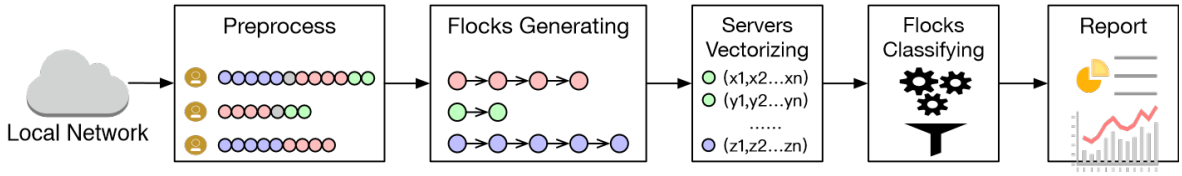

Fig.3. Architecture of deMSF

In this section, we describe our design of deMSF. The intuition of deMSF is that servers involved in one activity have strong relationships: (a) servers of a flock tend to co-occur within an interval in a client access list; (b) servers for the same campaign have similar clients. As shown in Figure 3, deMSF takes network traffic as input, and has four components: Preprocessing, Flocks Generating, Servers Vectorizing, Flocks Classifying. It leverages only three fields (client, timestamp, server) to analyze. In this paper, we take DNS logs as raw data. After process raw data and extract related fields, we generate server flocks from both temporal and spatial dimensions. Then we vectorize servers according to word2vec [21]. Finally, we build a deep learning classifier to recognize malicious flocks based on semantic vectors of servers. In the following, we will explain each component in detail.

3.1 Preprocessing

The primary goal of this step is to formalize the dirty raw data, extract valid fields and generate visit-sequences of clients. In order to reduce the data to be processed and improve the system efficacy, we first filter records according to the following rules.

- Irregular domain. There are some records in raw data with irregular domains(domains that do not conform to domain naming rules, for example, *google,com*), which is probably caused by mistyping or misconfiguration.
- Invalid domain. An invalid domain here indicates that its TLD(Top Level Domain) is not in the list of registered TLDs presented by IANA [10]. We filter records with these domains.
- Hyperactive clients. There are some hyperactive clients whose queries are greatly more than others, which are usually proxies forwarding requests for many users. In order to improve the performance of deMSF, we remove these clients cause they behave significantly different from regular clients. In detail, we remove the top H% most active clients. In this experiment, H is set to 1% empirically.

Then we formalize the data by extracting three valid fields: client, server and timestamp to generate request sequences. The form is defined as follows: $R = \cup C_i$ is the set of visit-sequences, where $C_i = \{(s_1,t_1),(s_2,t_2),...,(s_n,t_n)\}$ represent the visit-sequence of client i and (s_n,t_n) indicates that client i query server s_n at time t_n.

3.2 Flocks Generating

Based on the collected sequences of each client, deMSF further mines related servers that are involved in the same activity. We explore two steps to find server flocks from temporal and spatial dimensions. We give an example in Figure 4.

First, we execute clustering according to querying time. We analyze dns queries of ten clients within 900 seconds, Figure 5 shows the result: client's requests show obvious clustering phenomena in timeline. The result accords with our expectations as many network activities require more than one domain. For example, when query a web page, clients usually query other domains to download images. Besides, some programs have a static domain query list and order. We implement time clustering in a simple way: for two adjacent visits $(s_j,t_j),(s_{j+1},t_{j+1})$ of client C_i, if the time interval $\Delta T = t_{j+1} - t_j$ greater than a certain threshold τ (we set $\tau = 5$ in this article), we divide them into different clusters. After these step, we get a time-clustered sequence of client C_i as $\{s_1,s_2,...s_n\}$.

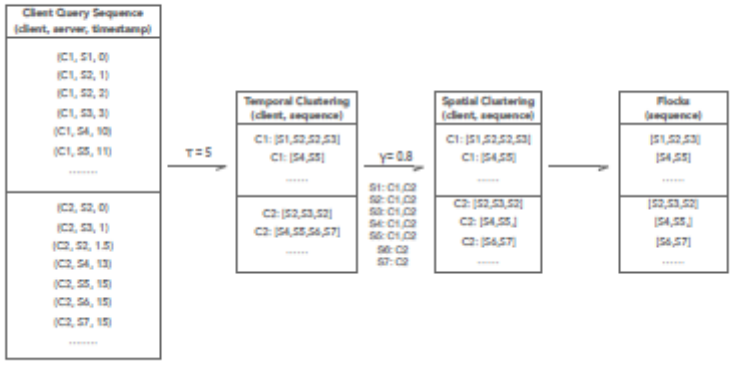

Fig. 4. An example of flocks generating

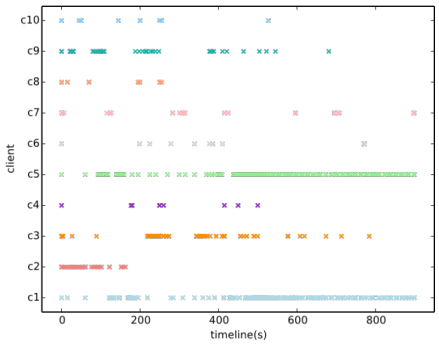

Fig.5. Domain queries of ten clients in 900 seconds

Second, we perform clustering in terms of the client similarity of servers. It depends on the intuition that normal clients usually don't query malicious servers while infected clients of a same malicious campaign usually query same suspicious servers. In other words, servers sharing similar client tend to belong to same flocks. We leverage Jaccard similarity to measure the connection of server s_i and s_j:

$$Similarity(s_i, s_j) = \frac{|Client_{s_i} \cap Client_{s_j}|}{|Client_{s_i} \cup Client_{s_j}|}$$

Specifically, for a time-clustered sequence $\{s_1,s_2,...s_n\}$, we calculate the client similarity of adjacent servers s_j and s_{j+1}. If the $Similarity(s_j,s_{j+1})$ is less than a certain threshold γ (γ is set to 0.5 empirically), we divide them into different clusters.

Finally, as our goal is to find the correlation among different servers, the small flocks with only one server are removed. In addition, if the adjacent two servers are the same, we only keep one.

3.3 Servers Vectorizing

The goal of this step is to map servers into a low-dimensional feature vector while keeping the context information as much as possible. We find that a technique named word embedding in natural language processing (NLP) is very helpful for learning features of servers. Word embedding based on a hypothesis: words that occur in the same contexts tend to have similar meanings. The same applies to servers: servers that occur in the same contexts tend to be similar. Thus we regard servers as words, a flock as a sentence, then we can learn features of servers the same as word embedding. Based on this, we leverage word2vec to learn feature vectors of servers, which can effectively describe the relationship among different servers.

Considering the time consuming and effect, we experiment with CBOW model. We implement it in Python, using the *Gensim*[†] package to generate server-vectors:

(a) the input layer contains 2a context servers, in this article, we set a=5.

(b) the output layer contains a vector, which is the server probability predicted according to the context. In the experiment, we set the size of vectors as 128.

3.4 Flocks Classifying

As we mentioned earlier, the server can be regarded as a word and the flock as a sentence. Then identifying malicious server flocks can be seen as a text classification task. Based on this perception, we design the neural network based on the textCNN [12] proposed by Kim in 2014. The structure is shown in Figure 6, consisting of an input layer, an embedding layer, convolution layers, max-pooling layers, a concatenate layer and an output layer. And we show the parameter settings in Table 1.

(a) Input Layer. The input layer takes flocks as input. A flock can be represented as a sequence $Seq_{flock} = \{s_1, s_2, \ldots s_n\}$, where n is the length of sequence.

(b) Embedding Layer. Let x_i be the k-dimensional server vector corresponding to the i-th server in the sequence. A sequence with n servers can be represented as $x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n$. The output of the embedding layer is a n*k matrix composed of server vectors of each sequence, where k is the length of vectors.

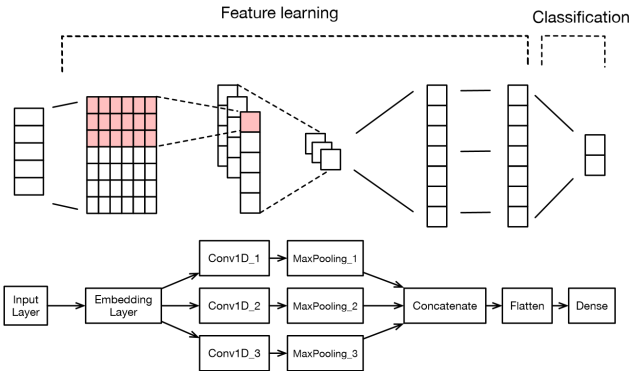

Fig.6. textCNN-based flocks classification system

Table 1. Parameter settings

Layer	Parameters
Input Layer	shape = (None,200)
Embedding Layer	server semantic vectors
Conv1D_1	filters = 100 kernal_size = 3 activation = 'relu'
MaxPooling_1	pool_size = 198
Conv1D_2	filters = 100 kernal_size = 4 activation = 'relu'
MaxPooling_2	pool_size = 197
Conv1D_3	filters = 100 kernal_size = 5 activation = 'relu'
MaxPooling_3	pool_size = 198
Concatenate	axis = -1
Flatten	N/A
Dense	units = number of categories activation = 'softmax' droupout=0.2

(c) Convolution Layer. There are three convolution layers with different window sizes of filters: 3,4,5. Each type of filters has 100 filters with different values. A feature c_i is generated with a filter which window size h:

$$c_i = f(w * x_{i:i+h-1} + b)$$

[†] https://radimrehurek.com/gensim/

In this work, we use Rectified Linear Units(ReLU) as f. A filter is applied to each possible window of servers in the sequence $\{x_{1:h},x_{2:h+1},...,x_{n-h+1:n}\}$ to produce a feature map $c = [c_1,c_2,...c_{n-h+1}]$.

(d) Max-pooling Layer. We apply a max-pooling operation over the feature map and take the maximum value $\hat{c}$ to capture the most important feature for a feature map. In this step, we get 300 features from 300 filters.

$$\hat{c} = max(c)$$

(e) Concantenate&flatten & output. All features are passed to a fully connected softmax layer whose output is the probability distribution over labels.

4 Evaluation

In this section, we evaluate the performance of deMSF using the real word DNS traffic captured from an ISP network. We first introduce the dataset used in our experiment. Then we analyze the results of server vectorizing and the effectiveness of deMSF.

4.1 DataSet

DNS traffic We obtain DNS traffic collected on the edge of an ISP network from December 20th, 2018 to December 26th, 2018. The summary of dataset is presented in Table 2. As we filter hyperactive clients in preprocessing step, we don't count them in Table 2.

Table 2. Dataset

Date	Clients	Domains	Queries	Flocks
2018-12-20	14.3k	412k	30,433k	119k
2018-12-21	14.2k	437k	25,497k	116k
2018-12-22	12.7k	279k	23,921k	79k
2018-12-23	12.3k	288k	27,000k	74k
2018-12-24	13.7k	424k	44,531k	116k
2018-12-25	13k	385k	34,221k	120k
2018-12-26	13k	386k	24,517k	119k

Ground Truth We get the ground truth from two popular online blacklists, Malware Domain Block List [7] and URLhaus [1]. Except above two blacklists, we also leverage a threat intelligence platform named ThreatBook [29] to scan all servers appeared in flocks and get their report. ThreatBook marks a server with three labels: clean, suspicious and malicious. Also some special clean servers will be marked as whitelist in ThreatBook.

4.2 Labelling

We first label servers according to ground truth we collect by following steps:

(a) a server is labeled as white if it is marked with whitelist by ThreatBook.
(b) a server is labeled as malicious if it is listed in any blacklists or is marked with malicious by ThreatBook.
(c) a server is labeled as suspicious if marked with suspicious by ThreatBook.
(d) a server is labeled as clean if it is marked with clean by ThreatBook and not listed in any blacklist.

Then we label flocks with harsh conditions. A flock is labeled as clean if all servers are labeled as white. A flock is labeled as malicious if its threat score is greater than 3. The threat score of a flock is the average score of all its servers and is calculated by the following formula.

$$score_{flock} = \frac{score_{s_1} + score_{s_2} + ... + score_{s_n}}{n}$$

$$score_{server} = \begin{cases} 0 & \text{white} \\ 1 & \text{clean} \\ 3 & \text{suspicious} \\ 5 & \text{malicious} \end{cases}$$

4.3 Server Vectorizing Results & Analysis

We expect semantic vectors of servers can effectively represent the internal relationship among servers, which means similar servers tend to have similar vectors. The internal relationship here indicates that servers have

similar content, provide similar service or have other connections like belonging to the same web.
We trained the vectors with one day data (2018-12-20) to check if the results meet our expectations. For a popular domain, we extract top 10 servers similar to it according to the semantic vectors and manually check whether they are similar in practical world. We give partial results in Table 3:

- *www.jd.com* is one of the biggest online shops in China. The top 10 servers similar to it are most the subpages of jd.com. It should be noted that there are three special examples *pf.3.cn, f.3.cn, dx.3.cn*, which are all related to *jd.com* cause *3.cn* is another domain of JD Inc. and can be redirected to *www.jd.com*. Another exception is *mediav.com*, which is a popular online advertising provider. It is reasonable that *mediav.com* is similar to *jd.com*
- *www.google.com* is the most popular search engine around world. The top 10 servers similar to it all belong to Google Inc.
- *sohu.com* is a popular portal web in China. Among the top 10 similar to itservers, there are four portal webs (*hao123.com, sina.com, qq.com, 163.com*), one popular search engine in China (*baidu.com*), the biggest online shop in China (*taobao.com*), and four servers related to a popular tool named Kingsoft Antivirus.

It can be seen that the semantic vectors can reflect internal connections of servers. Thus it is feasible to use the semantic vectors as features of servers.

Table 3. Top 10 servers similar to *www.jd.com, www.google.com, sohu.com*

Server	Similarity	Vector	Correlation
jcm.jd.com	0.9894	[0.04064134,-0.016787084...-0.050261103]	Subpage
cm.mediav.com	0.9891	[0.04064134,-0.016787084...-0.050261103]	Advertisement
ccc.jd.com	0.9875	[0.04064134,-0.016787084...-0.050261103]	Subpage
pf.3.cn	0.9872	[0.04064134,-0.016787084...-0.050261103]	Subpage*
f.3.cn	0.9859	[0.04064134,-0.016787084...-0.050261103]	Subpage*
api.m.jd.com	0.9846	[0.04064134,-0.016787084...-0.050261103]	Subpage
list.jd.com	0.9844	[0.04064134,-0.016787084...-0.050261103]	Subpage
ai.jd.com	0.9838	[0.04064134,-0.016787084...-0.050261103]	Subpage
dx.3.cn	0.9833	[0.04064134,-0.016787084...-0.050261103]	Subpage*
floor.jd.com	0.9831	[0.04064134,-0.016787084...-0.050261103]	Subpage
www.googletagmanager.com	0.9689	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
fonts.googleapis.com	0.9548	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
stats.g.doubleclick.net	0.9515	[0.09148411,-0.23106502...-0.007657597]	Advertisement of google
fonts.gstatic.com	0.9473	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
googleads.g.doubleclick.net	0.9450	[0.09148411,-0.23106502...-0.007657597]	Advertisement of google
www.googletagservices.com	0.9411	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
adservice.google.com	0.9399	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
www.googleadservices.com	0.9368	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
pagead2.googlesyndication.com	0.9330	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
ssl.google-analytics.com	0.9328	[0.09148411,-0.23106502...-0.007657597]	Google Inc.
hao123.com	0.9994	[0.36417392,-0.58563906...-0.12286949]	Portal web
sina.com	0.9992	[0.36417392,-0.58563906...-0.12286949]	Portal web
rq.upgrade.cloud.duba.net	0.9988	[0.36417392,-0.58563906...-0.12286949]	Anti-virus**
rq.cct.cloud.duba.net	0.9987	[0.36417392,-0.58563906...-0.12286949]	Anti-virus**
rcmd.pop.ijinshan.com	0.9987	[0.36417392,-0.58563906...-0.12286949]	Anti-virus**
taobao.com	0.9987	[0.36417392,-0.58563906...-0.12286949]	Online shopping
qq.com	0.9983	[0.36417392,-0.58563906...-0.12286949]	Portal web
cv.duba.net	0.9982	[0.36417392,-0.58563906...-0.12286949]	Anti-virus**
163.com	0.9982	[0.36417392,-0.58563906...-0.12286949]	Portal web
baidu.com	0.9981	[0.36417392,-0.58563906...-0.12286949]	Search engine

* *3.cn* is another domain of JD and is redirected to *www.jd.com*
** *duba.net* is a domain of Kingsoft Antivirus, which is one of the most widely used software in China

4.4 Classification Results & Analysis

Evaluation index

1. Accuracy&Precision&Recall&F1score.
   – TP: the number of malicious flocks deMSF detected as malicious.
   – TN: the number of clean flocks deMSF detected as clean.
   – FP: the number of clean flocks deMSF detected as malicious.
   – FN: the number of malicious flocks deMSF detected as clean.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall}$$

10-fold cross-validation We experiment with one day data (2018-12-20) to evaluate the effectiveness of the classifier. There are 49,760 different labeled flocks in 2018-12-20 data, including 603 malicious samples and 49,157 clean samples. In order to eliminate the randomness that may exist during some experiments and improve the reliability of results, we use 10-fold cross-validation for performance evaluation. Each experiment contains 39,808 samples in training sets, 4,976 samples in validation sets and 4,976 samples in test sets. The result shows in Table 4. It can be seen that the flock classification model based on server embeddings can achieve a high accuracy rate over 99%.

Table 4. 10-fold Cross Validation

No.	TP	TN	FP	FN	Accuracy	Precision	Recall	F1_socre
0	48	4925	0	3	99.94	100.00	94.12	96.97
1	53	4916	1	6	99.86	98.15	89.83	93.81
2	62	4912	0	2	99.96	100.00	96.88	98.41
3	60	4909	0	7	99.86	100.00	89.55	94.49
4	61	4908	0	7	99.86	100.00	89.71	94.57
5	45	4929	0	2	99.96	100.00	95.74	97.83
6	62	4907	0	7	99.86	100.00	89.86	94.66
7	46	4926	0	4	99.92	100.00	92.00	95.83
8	67	4906	0	3	99.94	100.00	95.71	97.81
9	51	4918	0	7	99.86	100.00	87.93	93.58

Experiment on one week data In order to analyze the effectiveness of deMSF with new servers, we further leverage one model trained above to predict the results with the next six days. One significant problem here is that there are many new servers that we don't know the semantic vectors of them. Retraining the vectors will change the value of vectors we used in the trained model and make the model not effective anymore. Thus we make an incremental training of server vectors with new flocks while keeping the original server vectors be constant. Then we use the newly trained vectors to make the classification. In order to measure the result of

classification, we only use labeled flocks to execute the experiment. The summary of data is presented in Table 5 and the result is showed in Table 6.

It can be seen that deMSF has excellent results. It has a high precision that all detected flocks are actually malicious flocks. It has an acceptable recall that only a few malicious flocks are not detected. This could be caused by the new threat that has weak connections with the known threat we trained thus deMSF cannot detect it. We show some examples of malicious flocks in Table 7.

Table 5. The summary of six days data

Date	Flocks(label)	Flocks(mal)	Flocks(clean)
2018-12-21	47,763	672	47,091
2018-12-22	30,156	655	29,501
2018-12-23	27,704	618	27,086
2018-12-24	48,552	665	47,887
2018-12-25	49,245	638	48,607
2018-12-26	48,955	667	48,288
All	252,375	3,915	248,460

Table 6. The result of six days data

Date	TP	TN	FP	FN	Accuracy	Precision	Recall	F1_socre
12-21	610	47091	0	62	99.87	100.00	90.77	95.16
12-22	606	29501	0	49	99.84	100.00	92.52	96.11
12-23	551	27086	0	67	99.76	100.00	89.16	94.27
12-24	605	47887	0	60	99.88	100.00	90.98	95.28
12-25	572	48607	0	66	99.87	100.00	89.66	94.55
12-26	610	48288	0	57	99.88	100.00	91.45	95.54
All	3,554	248,460	0	361	99.86	100.00	90.77	95.16

Table 7. Examples of malicious flocks

Type	Servers
Mirai	e.mariokartayy.com cnc.arm7plz.xyz cnc.junoland.xyz

Conficker	vqhxyffk.ws linepve.cc zzqvketg.info kpsvqpozld.net nzdkujuj.ws usjhbqrctb.cn ...
NrsMiner, CoinMiner	lebec.attendecr.com tar.kziu0tpofwf.club swt.njaavfxcgk3.club phelan.chereher.com p3.qsd2xjpzfky.site yuma.dification.com ...
Install Core	q96b7b7.strangled.net q968787.ignorelist.com q96b7b7.homenet.org q968787.mooo.com ...

4.5 Discussion

Overhead

The most expensive part of deMSF is to calculate the client similarity of servers. Since we should calculate similarity among different servers and there may be a large number of servers in data. Fortunately, there are some techniques like sparse matrix multiplication can significantly reduce the complexity of calculation.

Limitation

- Single malicious servers. deMSF focuses on multiple servers involved in malicious activities or evasion techniques instead of a single server. Thus, deMSF cannot detect malicious campaigns with only a single server cause there are
- no connections we can extract from these campaigns. However, malicious campaign with a single server is very rare.
- Noise. deMSF based on the query sequences of a client. It is inevitable that there are queries triggered by background activities mixed in the true continuous queries. Although we leverage the client similarity to decrease the noise, this phenomenon can not be eliminated. But it should be noted that noise is a small probability event. With the data increase, its impact is negligible. – New threat. Since deMSF leverage the inter-connections of servers according to client queries, deMSF can hardly detect completely new threats that don't have connections with before
- servers. To overcome this may need other properties and data sources. It can be a topic for our future work.

Evasion

- Attackers can make internal associations between benign servers and malicious servers by mixing benign queries in malicious activities. Thus deMSF may divide malicious servers within a campaign into different flocks and delete flocks cause they only contain one server. However, we can filter popular benign servers which are impossible involved in malicious campaigns by add whitelist in preprocessing step.
- Another approach attackers can use is to let different compromised clients communicate with different servers to reduce the client similarity of malicious servers. However, this may be costly for attackers, as the more bots they have, the more servers they need to register.
- One more method attackers can use is increasing the time interval between two queries. While some attacks require continuous queries such as malicious redirections and DGA. In addition, researchers can adjust the time window threshold to catch them.

Universality

- deMSF is designed to monitor the traffic from the edge of a network and it only requires basic three

fields: client, timestamp and server, thus it can be deployed at most enterprise or ISP networks.

- deMSF is an automatic threat discovery system. It leverages a basic hypothesis that servers occur in the same contexts tend to have similar meanings. Then it learns semantic vectors of servers to get the internal association between them and further classifies malicious flocks from normal activities. It should be noted that deMSF does not need any defined feature rules or knowledge.
- deMSF don't need researchers to manually adjust parameters to get the proper value. By training a sufficiently good model, deMSF can discover malware behaviors and exclude known non-malicious behaviors. While the parameters can be stable and effective for a long time.

5 Related work

5.1 Studies focus on Individual Servers.

Many approaches concentrate on individual malicious servers to mitigate malicious avitivites.

Some works analyze web content to recognize malicious webs. Liao et al. [16] develop a semantic-based technique, which leverages Natural Language Processing (NLP) to identify the bad terms most irrelevant to an sTLD's semantics and detects webpages with malicious promotional injections. Delta [5] is a system identifing malicious web sites according to the changes of sites. It extracts change-related features between two versions of the same website and identifies an infection using signatures generated from such modifications. Saxe et al. [24] propose a deep learning approach to detecting malevolent web pages operated on a language-agnostic stream of tokens extracted directly from static HTML files with a simple regular expression.

Some works construct reputation system for a single server to recognize malicious servers. Notos [3] is a dynamic reputation system for domains. It uses passive DNS query data to construct the network and zone features of domains and compute accurate reputation scores. EXPOSURE [4] employs large-scale, passive DNS analysis techniques to detect malicious domains. It extracts 15 features from DNS traffic to characterize different properties of domains and the ways they are queried. PREDATOR [9] uses only time-of-registration features to establish domain reputation to predict malicious domains when they are registered.

Some concentrate on the technique adversaries use to avoid detection. Yadav et al. [32] develop a methodology to detect domain fluxes in DNS traffic by looking for patterns inherent to domain names that are generated algorithmically, in contrast to those generated by humans. Phoenix [25] is a mechanism using a combination of string and IP-based features to tell DGA and non-DGA domains.It can find groups of DGA domains that are representative of the respective botnets. It can associate previously unknown DGA-generated domains to these groups, and produce novel knowledge about the evolving behavior of each tracked botnet. WoodBridge et al. [30] leverages long short-term memory (LSTM) networks to predict malicious domains and their respective families. Luo [18] leverages the query time lags of non-existent domains (NXDomain) to mitigate DGA-based malware without the lexical property.

5.2 Studies focus on relations of servers.

There are many studies focus on malicious redirections. VisHunter [34] investigates the visibility of servers and finds that certain malicious servers tend to be invisible to normal users. It identifies malicious redirections from visible servers to invisible servers at the entryway of malicious web infrastructures. Akiyama et al. [2] develope a honeypot-based monitoring system across four years and analyze the ecosystem of malicious URL redirections. Stringhni et al. [27] aggregate the different redirection chains that lead to a specific web page and analyze the characteristics of the resulting redirection graph. Then they detect malicious web pages by looking at the redirection chains that lead to them. Mekky et al. [19] develop a methodology to identify malicious chains of HTTP redirections. They passively collected traffic and extract statistical features which capture inherent characteristics from malicious redirection cases. They further apply a supervised decision tree classifier to identify malicious chains.

Some works leverage many other relations of malicious servers. Zhang et al. [35] utilize an unsupervised framework to infer malware associated server herds by systematically mining the relationships among all servers from multiple dimensions: client similarity, IP address set similarity, whois similarity, URI file similarity. Li et al. [15] perform a study on the topological relations among hosts and find that dedicated malicious hosts are well connected to other malicious hosts and do not receive traffic from legitimate sites. They develope a graphbased approach that relies on a small set of known malicious hosts as seeds and results in an expansion rate of over 12 times in detection. Lee et al. [14] construct a domain travel graph based on the sequential correlation of DNS, cluster domains using the graph structure and determine malicious clusters by referring to public blacklists. Sun et al. [28]model the DNS scene as a Heterogeneous Information Network (HIN) consist of clients, domains, IP addresses and their diverse relationships. They leverage a transductive classification method to detect malicious domains with only a small fraction of labeled samples. Liu et al. [17] analyze a new attack infrastructures named shadowed domain. They propose a system to detect these domains from two dimensions: the deviation from legitimate domains under the same apex and the correlation among shadowed domains under a different apex.

5.3 Studies using embedding in security

Xu et al. [31] propose a neural network-based model to generate vectors based on the control flow graph of each binary function. Then the cross-platform binary code similarity detection problem can be done efficiently by measuring the distance between vectors. Popov [23] proposes a method applying word2vec technique for extracting vector embeddings of machine code instructions. And further build a convolutional neural network-based classifier using extracted vectors to detect malware. Ding et al. [6] develope a representation learning model named Asm2Vec to construct feature vectors for assembly code. It takes assembly code as input and does not require any prior knowledge such as the correct mapping between assembly functions. It can find and incorporate rich semantic relationships among tokens appearing in assembly code. Shen et al. [26] calculate the vector of an attack step by considering the entire attack sequence as a sentence, and each step as a word. They develop attack2vec to understand the emergence, the evolution, and the characteristics of attack steps in relation to the wider context in which they are exploited.

6 Conclusion

In this paper, we focus on the servers that are involved in the same malicious campaign. We learn the features of vectors leveraging the querying relationships among different servers and propose a novel approach to detect malicious activities using a neural network based on server semantic vectors. deMSF first mines server flocks from both temporal and spatial dimensions. Further it generates server semantic vectors with the techniques developed in the area of natural language processing, which can effectively model the internal connection among servers. Finally it recognizes malicious flocks by a deep neural network based on all server vectors of a flock. The feasibility of deMSF is demonstrated with one week logs acquired from real-world, and the results show that deMSF achieves a high precision of 99% with 90% recall.

References

[1] Abuse: Urlhaus. https://urlhaus.abuse.ch

[2] Akiyama, M., Yagi, T., Yada, T., Mori, T., Kadobayashi, Y.: Analyzing the ecosystem of malicious url redirection through longitudinal observation from honeypots. Computers & Security **69**, 155–173 (2017)

[3] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for dns. In: USENIX security symposium. pp. 273–290 (2010)

[4] Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis. In: Ndss. pp. 1–17 (2011)

[5] Borgolte, K., Kruegel, C., Vigna, G.: Delta: automatic identification of unknown web-based infection campaigns. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 109–120 (2013)

[6] Ding, S.H., Fung, B.C., Charland, P.: Asm2vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 472–489. IEEE (2019)

[7] DNS-BH: Malware domain blocklist. http://www.malwaredomains.com

[8] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

[9] Hao, S., Kantchelian, A., Miller, B., Paxson, V., Feamster, N.: Predator: proactive recognition and elimination of domain abuse at time-of-registration. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1568–1579. ACM (2016)

[10] IANA: Top level domain. http://www.iana.org/domains/root/db/

[11] Kim, B.I., Im, C.T., Jung, H.C.: Suspicious malicious web site detection with strength analysis of a javascript obfuscation. International Journal of Advanced Science and Technology **26**, 19–32 (2011)

[12] Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751 (2014)

[13] Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**(2) (2012)

[14] Lee, J., Lee, H.: Gmad: Graph-based malware activity detection by dns traffic analysis. Computer Communications **49**, 33–47 (2014)

[15] Li, Z., Alrwais, S., Xie, Y., Yu, F., Wang, X.: Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In: 2013 IEEE Symposium on Security and Privacy. pp. 112–126. IEEE (2013)

[16] Liao, X., Yuan, K., Wang, X., Pei, Z., Yang, H., Chen, J., Duan, H., Du, K., Alowaisheq, E., Alrwais, S., et al.: Seeking nonsense, looking for trouble: Efficient promotional-infection detection through semantic inconsistency search. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 707–723. IEEE (2016)

[17] Liu, D., Li, Z., Du, K., Wang, H., Liu, B., Duan, H.: Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 537–552 (2017)

[18] Luo, X., Wang, L., Xu, Z., An, W.: Lagprober: Detecting dga-based malware by using query time lag of non-existent domains. In: International Conference on Information and Communications Security. Springer (2018)

[19] Mekky, H., Torres, R., Zhang, Z.L., Saha, S., Nucci, A.: Detecting malicious http redirections using trees of user browsing activity. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications. pp. 1159–1167. IEEE (2014)

[20] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[21] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[22] Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 conference of the north american chapter of the

association for computational linguistics: Human language technologies. pp. 746–751 (2013)
[23] Popov, I.: Malware detection using machine learning based on word2vec embeddings of machine code instructions. In: 2017 Siberian Symposium on Data Science and Engineering (SSDSE). pp. 1–4. IEEE (2017)
[24] Saxe, J., Harang, R., Wild, C., Sanders, H.: A deep learning approach to fast, format-agnostic detection of malicious web content. In: 2018 IEEE Security and Privacy Workshops (SPW). pp. 8–14. IEEE (2018)
[25] Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Phoenix: Dga-based botnet tracking and intelligence. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 192–211. Springer (2014)
[26] Shen, Y., Stringhini, G.: Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In: 28thUSENIX Security Symposium. pp. 905–921 (2019)
[27] Stringhini, G., Kruegel, C., Vigna, G.: Shady paths: Leveraging surfing crowds to detect malicious web pages. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 133–144. ACM (2013)
[28] Sun, X., Tong, M., Yang, J., Xinran, L., Heng, L.: Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In: 22nd International Symposium on Research in Attacks,
[29] Intrusions and Defenses ({RAID} 2019). pp. 399–412 (2019)
[30] ThreatBook: https://x.threatbook.cn
[31] Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. arXiv preprint arXiv:1611.00791 (2016)
[32] Xu, X., Liu, C., Feng, Q., Yin, H., Song, L., Song, D.: Neural network-based graph embedding for cross-platform binary code similarity detection (2017)
[33] Yadav, S., Reddy, A.K.K., Reddy, A., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. pp. 48–61. ACM (2010)
[34] Yin, L., Luo, X., Zhu, C., Wang, L., Xu, Z., Lu, H.: Connspoiler: Disrupting c&c communication of iot-based botnet through fast detection of anomalous domain queries. IEEE Transactions on Industrial Informatics (2019)
[35] Zhang, J., Hu, X., Jang, J., Wang, T., Gu, G., Stoecklin, M.: Hunting for invisibility: Characterizing and detecting malicious web infrastructures through server visibility analysis. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. pp. 1–9. IEEE (2016)
[36] Zhang, J., Saha, S., Gu, G., Lee, S.J., Mellia, M.: Systematic mining of associated server herds for malware campaign discovery. In: 2015 IEEE 35th International Conference on Distributed Computing Systems. pp. 630–641. IEEE (2015)