Fast and Accurate Typosquatting Domains Evaluation with Siamese Networks

Jing Ya[1,2], Tingwen Liu[1,2], Quangang Li[1,2*], Pin Lv[1,2], Jinqiao Shi[1,2], Li Guo[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
Email: {yajing, liutingwen, liquangang, lvpin, shijinqiao, guoli}@iie.ac.cn

***Abstract*— Typosquatting is a form of cybersquatting, which is a practice of registering typosquatting domain names that closely resemble legitimate and popular ones. It has become a serious speculation, as a large number of typosquatting domains are used to seek illegal interests or illegal purposes. Thus, many researches have been made on typosquatting in recent years. Some of them actively generate possible typo-variants of legitimate domains, and measure typosquatting phenomena in distribution, monetization and cost *etc*. Others detect typosquatting domains observed in passive DNS traffic through string similarity calculation by edit distance and time correlation. In this paper, we follow the latter work and propose a novel approach (named TypoEval) capable of evaluating typosquatting domains fast and accurately. Concretely, we exploit siamese neural networks to learn an embedding per domain and evaluate typosquatting domains by calculating the distance between vectors in Euclidean space. We validate our TypoEval approach on a real world data set. Experimental results show that TypoEval can improve the shortcomings of edit distance which is used by most of the previous work, and it is efficient and effective in evaluating typosquatting domains.**

I. Introduction

Typosquatting, a form of cybersquatting, is a practice of registering typosquatting domains that closely resemble legitimate and popular ones (*e.g.*, `google.com` vs `gooogel.com`). Cybercriminals register such domains in hopes of selling them back to companies and trademark owners for a substantial profit [1], parking them and making pay-per-click revenues from users' misspellings and typing errors [2], redirecting typosquatting traffic to competitors [3], deploying phishing sites for intercepting passwords or installing drive-by malicious software [4]. Recently, typosquatting domains also have been used in APT (Advanced Persistent Threat) attacks, such as "`qooqle.co`"[1] and "`gooogel.org`"[2].

Typosquatting has become a speculative and serious phenomenon for both Internet users and trademark owners. Many companies, including Verizon, Lufthansa, and Lego, have garnered reputations for aggressively chasing down typosquatting domains [4]. Visiting typosquatting domains may result in automatically downloading various computer viruses and other malicious softwares to the computer.

To date, many works have been done on typosquatting domains. Some of them actively generate possible typo-variants of legitimate domains, and measure typosquatting phenomena in distribution, monetization and cost [5], [6], [7], [8]. They use the most common typo-variant operations: addition, deletion, substitution of one character; transposition of neighboring characters; and the "." deletion operation specific to "www." domain names. Others detect typosquatting domains observed in passive DNS traffic through lexical similarity based on edit distance and time correlation [9], [10], [11]. In this paper, we follow the latter work, and propose a novel approach to evaluate typosquatting domains through domain similarity.

According to the definition and intent of typosquatting, a domain is doubtful and likely to be a typosquatting domain if it looks very similar with one popular website but not registered by the brand owner. Most of the previous detection approaches are based on the edit distance between domains. Edit distance is a way of quantifying how dissimilar two strings are by counting the minimum number of operations required to transform one string into the other. So, two domains with smaller edit distance are likely to have higher similarity.

However, such methods do not fully utilize the context information inside domain names and have some shortcomings. First, they can not capture all the typo-variants. For example, the edit distances of ("qooqel.com" vs "google.com") and ("abcgle.com" vs "google.com") are both 3. But the former one is likely to be a typosquatting domain, which tries to confuse users with similar characters. Second, they usually give lots of false positive results, especially for short domains. For example, the edit distance between "jd.com" and "qq.com" is 2, but obviously "qq.com" is not a typosquatting domain of "jd.com". Some work are proposed to improve the accuracy by crawling auxiliary data such as web pages and Whois information. However, crawling such data usually requires high bandwidth and many computing resources. What is more, for a given domain, searching of the maximum similarity using edit distance over large-scale legitimate domains also consumes much time.

To address the above problems, we propose a novel approach named TypoEval, which can evaluate typosquatting domains fast and accurately. Specifically, we exploit siamese neural networks to learn an embedding per domain and evaluate typosquatting domains by calculating the distance between vectors in Euclidean space. We use Recurrent Neural Networks (RNNs) to fully utilize the context information inside the

*Quangang Li is the corresponding author of this paper.
[1]https://www.threatminer.org/domain.php?q=qooqle.co
[2]https://www.threatminer.org/domain.php?q=gooogel.org

978-1-5386-7185-6/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: Institute of Information Engineering CAS. Downloaded on August 19,2022 at 08:00:47 UTC from IEEE Xplore. Restrictions apply.

character sequence of a domain. The siamese networks are trained such that the squared L2 distances in the embedding space directly correspond to domain similarity. A domain has small distance with its typosquatting domains and has large distance with unrelated domains. In that way, we can reduce the problem of typosquatting evaluation to a problem of vector distance calculation. We further speed up the evaluation procedure through batch processing by using GPU.

The main contributions of this paper can be briefly summarized as follows:

1) We propose a novel approach named TypoEval, which can evaluate typosquatting domains fast and accurately.
2) We reduce the problem of typosquatting domains evaluation to a vector distance calculation problem, and speed up the procedure through batch processing by using GPU.
3) Experimental results show that TypoEval can improve the shortcomings of prior edit distance based methods, and is efficient and effective to evaluate typosquatting domains.

The rest of the paper is organized as follows. We give a detailed description of our approach in Section II. The implementation details and experimental results are present in Section III. And, we summarize the related work in Section IV. Finally, we conclude the paper's work in Section V.

II. TypoEval Method

This section gives a detailed description of our TypoEval approach. First, we express the problem of typosquatting domains evaluation and formulate it symbolically. Then, we describe some technique details of our approach.

A. Problem Statement

Before we try to solve one problem, we should first precisely formulate what the problem actually is. Here in this part, we introduce and express the problem of typosquatting domains evaluation symbolically.

According to the definition and intent of typosquatting, a typosquatting domain is targeting to a legitimate one. Then, the problem of typosquatting evaluation can be defined as follows: input an unknown domain s, for the pre-given set T of targeted domains, we want to get the most similar domain $t(t \in T)$ with s. That means we want to get the minimum distance ($\min\{D(s,t)|t \in T\}$), where $D(s,t)$ is the distance between domains: s and t. Because two domains with smaller distance will have higher similarity. We can then use the minimum distance to evaluate whether the input domain s is a typosquatting domain of the corresponding targeted domain t by setting up a threshold β empirically. If the minimum distance is smaller than β, the domain s is likely to be a typosquatting domain.

So, the key problem is how to calculate the distance between domains. On one hand, the distance should directly correspond to domain similarity. On the other hand, the calculation procedure should be as fast as possible. In this paper, we reduce it to a problem of vector distance calculation in Euclidean space. We first learn an embedding per domain with siamese recurrent networks, and fully utilize the context information

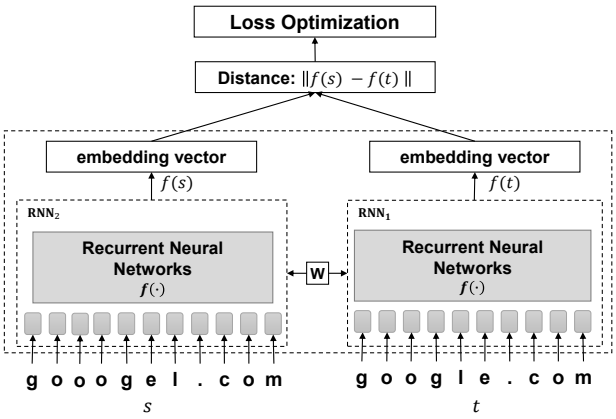

Fig. 1. Architecture of Our Siamese Recurrent Networks

inside the character sequences of domains.. Then, we use Euclidean distances between vectors to evaluate typosquatting domains. And, this procedure can be speeded up through batch processing by using GPU.

B. Networks Architecture

The architecture of our siamese recurrent networks is outlined in Fig. 1. The input is a pair of domains (s, t). In the pair, s is the input unknown domain which is given as a domain to be evaluated, and t is one of the targeted domains ($t \in T$) which are given as legitimate and popular ones.

The networks read in the character sequences of each domain in a given pair. There are two recurrent networks RNN_1 and RNN_2 which each processes one of the domains. The networks learn a mapping from the space of variable length sequences of d_{in}-dimensional vectors into $R^{d_{rep}}$. We solely focus on the siamese architectures which are tied with the same weights such that RNN_1 = RNN_2 [12]. Recurrent Neural Networks (RNNs) can fully utilize the context information inside sequences. Specifically, we use Long Short Term Memory networks (LSTM) in this paper. The details are described in Section II-D. W is the shared parameter vector that is subject to learning. $f(s)$ and $f(t)$ are the two vectors learned in the low-dimensional space that are generated by embedding s and t. Then our system can be viewed as a scalar 'Distance' that measures the dissimilarity between s and t. In that way, we can evaluate typosquatting domains through the distance between two domains with the following formula: $Distance(f(s), f(t)) = \| f(s) - f(t) \|$.

We use Euclidean distance in this paper. Empirically, the results should be fairly stable across different types of distance functions, but we find that Euclidean distance slightly performs better than other reasonable alternatives.

Euclidean distance [13] is the ordinary straight-line distance between two points $x_1 = f(s)$ and $x_2 = f(t)$ in Euclidean space, and is the length of the line segment connecting them, which is defined as the following formula:

$$d = \sqrt{\sum_{i=1}^{N}(x_{1i} - x_{2i})^2} \tag{1}$$

Authorized licensed use limited to: Institute of Information EngineeringCAS. Downloaded on August 19,2022 at 08:00:47 UTC from IEEE Xplore. Restrictions apply.

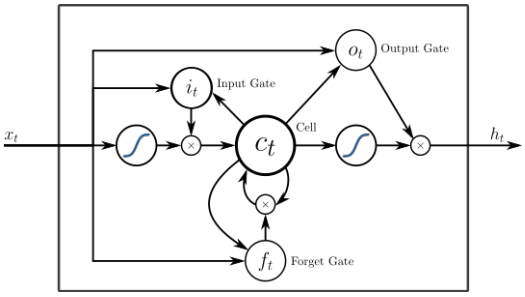

Fig. 2. Architecture of a LSTM unit

where $N = d_{rep}$.

C. Loss Function

The siamese networks take a pair of domains as input, and train the embedding for each domain. The distance between two domains is minimized if s is a typosquatting domain of the targeted domain t and is greater than some margin value if they are not related at all. We will minimize a contrastive loss function used in [14]. The loss function depends on the input and the parameters indirectly through the distance between two input domains.

The learned embedding is represented by $f(x) \in R^{d_{rep}}$. It embeds a domain name x into a d_{rep}-dimensional Euclidean space. This loss is motivated in [15] in the context of nearest-neighbor classification. For a given targeted domain x_i^t, we want to ensure that it is closer to all the typosquatting domains x_i^p which are targeted to it (positive samples) than it is to any other unrelated domains x_i^n (negative samples). This can be formulated as:

$$\|f(x_i^t) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^t) - f(x_i^n)\|_2^2 \tag{2}$$

For a batch with the size of M, the loss that is being minimized as L:

$$\sum_i^M [\|f(x_i^t) - f(x_i^p)\|_2^2 - \|f(x_i^t) - f(x_i^n)\|_2^2 + \alpha]_+ \tag{3}$$

D. Recurrent Neural Networks

In natural language process, RNNs have been used to capture meaningful temporal relationships among elements in a sequence. The key benefit of RNNs is that they incorporate contextual information in their mapping from input to output.

LSTM networks are variants of RNNs. They were introduced by Hochreiter and Schmidhuber (1997), and were refined and popularized by many people in the following work. They work tremendously well on a large variety of problems, and are now widely used. LSTM networks are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTM networks sequentially update a hidden-state representation, but these steps also rely on a memory cell containing four components (which are real-valued vectors): a memory state c_t, an output gate o_t that determines how the memory state affects other units, as well as an input and an forget gate i_t and f_t which control what gets stored in and omitted from memory based on each new input and the current state. A peephole LSTM unit is shown in Fig. 2. The formulas below are the updates performed at each unit t in an LSTM parameterized by weight matrices $W_f, W_i, W_o, W_c, U_f, U_i, U_o, U_c$, and bias-vectors b_f, b_i, b_o, b_c [16].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{6}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{7}$$
$$h_t = o_t \odot \sigma_h(c_t) \tag{8}$$

More technical details and thorough exposition of the LSTM networks and some variants of the model are provided by [16], [17], [18].

In our approach, we use LSTM networks to read in character sequence representing each input domain and employ its final hidden state as a vector representation for each domain. Subsequently, the similarity between these representations is used as a predictor of domain similarity to evaluate typosquatting domains.

E. Typosquatting Evaluation

After we get the embedding of each domain, we can evaluate typosquatting domains through Euclidean distance between vectors. Specifically, we first get the embedding of the input domain $f(s) : (x_1, x_2, \ldots, x_{d_{rep}})$ with the trained model. Now the problem is to calculate the minimum distance $\min D(f(s), f(t))$, where $t \in T$. The embeddings of domains in T can be learned beforehand off line: $(f(t_1), f(t_2), \ldots, f(t_n))$.

Typosquatting domains evaluation is defined as the following operations.

$$D_{min} = \min(D_1, D_2, \ldots, D_i, \ldots, D_n) \tag{9}$$

where D_{min} is used to evaluate typosquatting domains.

And, $(D_1, D_2, \ldots, D_i, \ldots, D_n) =$

$$\begin{pmatrix} x_1 & x_2 & \ldots & x_{d_{rep}} \end{pmatrix} \otimes \begin{pmatrix} x_{11} & x_{21} & \ldots & x_{n1} \\ x_{12} & x_{22} & \ldots & x_{n2} \\ & \ldots & & \\ & \ldots & & \\ x_{1d_{rep}} & x_{2d_{rep}} & \ldots & x_{nd_{rep}} \end{pmatrix} \tag{10}$$

The matrix operator $\otimes$ is defined as similar as the matrix multiplication operation. In this paper, it is defied as:

$$D_i = \frac{\sqrt{\sum_{j=1}^{d_{rep}} (x_j - x_{ij})^2}}{\sqrt{\sum_{j=1}^{d_{rep}} (x_j)^2} + \sqrt{\sum_{j=1}^{d_{rep}} (x_{ij})^2}} \tag{11}$$

Authorized licensed use limited to: Institute of Information EngineeringCAS. Downloaded on August 19,2022 at 08:00:47 UTC from IEEE Xplore. Restrictions apply.

where the numerator part is Euclidean distance [13], and the denominator part is used to normalize the Euclidean distance.

Apparently, $\sum_{j=1}^{d}\sqrt{(x_j)^2}$ and $\sum_{j=1}^{d}\sqrt{(x_{ij})^2}$ are the distances of the two points to the origin. From the triangle theorem, we know that:

$$\sqrt{\sum_{j=1}^{d_{rep}}(x_j - x_{ij})^2} < \sqrt{\sum_{j=1}^{d_{rep}}(x_j)^2} + \sqrt{\sum_{j=1}^{d_{rep}}(x_{ij})^2} \qquad (12)$$

So, D_i must be a value between 0 and 1. Then, the similarity between each pair of domains can be calculated as $1 - D_{min}$.

We can set a threshold β to evaluate whether the domain is a typosquatting domain of the targeted one. For example, we set 0.5 in this paper.

III. Experimental Results

This section presents our implemental details and experimental results. We first give a brief description of our experimental setup, including experimental data sets and validation metrics. Then we validate the effectiveness and efficiency of our method. Finally, we give some further discussions.

A. Experimental Setup

1) DataSet: To validate the effectiveness and efficiency of our approach, we first construct a data set from the real world with an open source tool "typofinder"[3]. The tool is released by NCC Group on the Github to discover typosquatting domains. We collect the top one million domain names provided by Alexa website[4]. And, we choose the top 500 domains as seeds to discover their typosquatting domains in real world with "typofinder". For each domain name, "typofinder" generates a candidate list of typosquatting domains based on the common typosquatting construction models. And, it actively obtains the relevant information of each candidate domain name to determine whether it is a real typosquatting domain. So, the accuracy rate is quite high.

In order to validate the effectiveness of our work on short domain names clearly, we limit the length of each domain is no longer than 20. For the top 500 domain names, we totally discover 26,273 typosquatting domains with "typofinder", namely 26,273 positive pairs. Then, we construct negative pairs from the top 10000 domain names from Alexa. As the top 10000 domain names are all with large amount of access traffic in the Internet, we can regard that any domain among them can not be a typosquatting domain of the others. We limit the edit distance between domains in each pair is not larger than 3. Totally, we construct 29,998 negative pairs.

Our experimental dataset includes 56,271 pairs of domains from the real world, with 26,273 positive samples and 29998 negative samples. To train and test our model, we randomly separate the two data sets into two parts respectively (70% for training, 30% for testing).

[3]https://github.com/nccgroup/typofinder

[4]https://www.alexa.com/topsites

TABLE I
Hyper Parameters

Parameter	Value	Parameter	Value
Input dimension	300	Out dimension	64
Learning rate	0.01	Dropout rate	0.2
Hidden dimension	128	Batch size	128

TABLE II
Results Comparisons

Method	Precision	Recall	F_1
Edit Distance ($\alpha = 1$)	0.9851	0.9122	0.9472
Edit Distance ($\alpha = 2$)	0.8170	0.9439	0.8758
TypoEval ($\beta = 0.5$)	0.9719	0.9811	0.9765

2) Validation Metrics: We validate the effectiveness of our work with three metrics: precision rate, recall rate, and F_1 measure. When the method correctly predicts a typosquatting domain, it is a true positive (TP). When the method correctly predicts a normal domain, it is a true negative (TN). When the method predicts a normal domain as a typosquatting domain, it is a false positive (FP). When the method predicts a typosquatting domain as a normal domain, it is a false negative (FN). The precision rate is defined as the proportion of predicted typosquatting domains that are really typosquatting, which is $precision = \frac{TP}{TP+FP}$. The recall rate is the proportion of real typosquatting domains that are correctly predicted, which is $recall = \frac{TP}{TP+FN}$. The F_1 measure is the harmonic mean of the precision rate and the recall rate, and defined as $F_1 = \frac{2*precision*recall}{preciison+recall}$.

3) Hyper Parameters for Training: There are several hyper parameters in our deep neural networks. In this paper, we set them empirically. The details of hyper parameters used in our experiments are shown in Table I.

We use TensorFlow[5] to implement our model, which is a popular python framework for implementing deep neural networks. And, we run the programs on a single NVIDIA Tesla P100-PCIE-12GB GPU.

B. Effectiveness Validation

Given a pair of domains, our goal is to predict whether the former one is a typosquatting domain of the latter one (which is one of the targeted domains). We validate the effectiveness of our TypoEval method compared with prior edit distance calculation methods. Specifically, we set the threshold of edit distance calculation method α to be 1 and 2 respectively, and the threshold of our TypoEval method β is set as 0.5. If the similarity of the two domains is larger than 0.5, then we consider the given domain is a typosquatting domain of the target one. And, the results of the two methods on typosquatting domains evaluation are shown in Table II.

From the Table II, we can observe that TypoEval performs better than the edit distance based calculation methods on F_1 measures with a 2.93% upgrade and a 10.07% upgrade

[5]https://www.tensorflow.org

Authorized licensed use limited to: Institute of Information EngineeringCAS. Downloaded on August 19,2022 at 08:00:47 UTC from IEEE Xplore. Restrictions apply.

TABLE III
SOME EXAMPLES OF RESULTS

Domain pairs	Edit distance	TypoEval distance	Label
(go.com, jd.com)	2	0.975196	normal
(hp.com, qq.com)	2	0.783284	normal
(lg.com, qq.com)	2	0.995456	normal
(bt.com, qq.com)	2	0.996082	normal
(jt.com, jd.com)	1	0.29464	typosquatting
(sc.com, qq.com)	1	0.243301	typosquatting
(qq.cm, qq.com)	1	0.198599	typosquatting
(qu.com, qq.com)	2	0.116243	typosquatting
(ww.com, qq.com)	2	0.255587	typosquatting
(g00gles.com.tw, google.com.tw)	3	0.220611	typosquatting
(g00gleing.com, google.com)	5	0.284614	typosquatting

respectively for $\alpha = 1$ and $\alpha = 2$. We can also observe that when the threshold α is set to be 2, the precision of edit distance calculation is not up to 1.0, but obviously lower. The reason is that there are many pairs of domains like ('jd.com', 'qq.com') in reality, in which the edit distance of the two domains is small, but they are not typosquatting domains at all apparently.

Table III shows some examples of the distance calculated by the two methods. From the table, we can observe that edit distance calculation method may have many false positive results especially for short domains. While, TypoEval can address this problem owing to considering the full context information inside the character sequences of domains. It can also learn some intrinsic information in character sequences. For example, the edit distance between domains in pair (google.cn, g00gles.cn) is 3. But the distance calculated by TypoEval is very small, which means it is very likely to be a typosquatting domain. As character '0' is similar with 'o', it may trick users into a trap. Edit distance can not observe this phenomenon which can be learned by deep neural networks with character sequences as inputs.

C. Efficiency Validation

To validate the efficiency of our method, we compare the time used by edit distance calculation based method and our TypoEval method. More concretely, we implement the two methods on data sets of different sizes, and compare time they use respectively. The input data of the two methods are both pairs of domains.

Table IV shows the speedup of our TypoEval method on different sizes of input pairs. From the table, we can observe that the time of edit distance calculation increases linearly with the size of data. Our method is orders of magnitude faster than the linear searching method with matrix operations. The reason is that deep learning is based on matrix operations and can be run in batch on a GPU card, rather than linearly. Furthermore, we can improve the throughput of our method by using more GPU cards or parallel processing technology, such as multicore and multithreaded processing.

TABLE IV
SPEEDUP OF TYPOEVAL METHOD

# of pairs	Time of EditDistance	Time of TypoEval	Speedup
20000	1542.27ms	330.87ms	4.66
40000	3127.78ms	431.01ms	7.26
80000	5650.20ms	584.50ms	9.67
100000	6562.88ms	671.89ms	9.77
200000	14055.34ms	1093.51ms	12.85
300000	21132.21ms	1453.48ms	14.54

D. Further Discussions

The approach proposed in this paper evaluates typosquatting domains with the distance between domains. Specially, we reduce it to a problem of vector distance calculation. We use siamese recurrent neural networks to fully utilize the context information inside character sequences of domains, and learn an embedding per domain directly corresponds to domain similarity. In that way, we can improve the shortcomings of prior edit distance based methods. However, it may also have some false positive results. For example, two legitimate domains "jd.com" and "td.com" are coincidentally similar. To reduce the false positive rate, more features should be taken into account, as done in prior work of detecting malicious domains from DNS traffic.

Despite this, experimental results show that our method may improve the detection of typosquatting domains over the state of the art. This method can remove many false positive results of edit distance based methods, and return more suspicious domains. And, further analysis can be used only on the results of our method.

IV. RELATED WORK

Recently, more and more researchers pay attentions on typosquatting domains. They attempt to identify typosquatting domains and measure typosquatting phenomenon to discover the preferred monetization strategies of typosquatters. Identifying typosquatting domains is a necessary step to measure typosquatting phenomenon. Most prior work on typosquatting involve the two parts.

1) Typosquatting Identification: Most of existing approaches on typosquatting domains identification are active [5], [6], [7], [8]. They usually work with generative models. They first select a targeted domain set from passive data sources (such as the Alexa top list of one million popular websites), and then construct a list of all strings which are lexically similar to the selected targeted domains, following some known and predefined typosquatting models. For these constructed strings, they collect registration data, DNS data and web data with many self-developed crawlers. Based on the collected data, they make an active analysis to identify typosquatting domains which might have been registered by typosquatters. Although these active approaches can identify typosquatting domains, they cannot capture all typosquatting domains because of limited targeted domains and distances, and cannot capture typosquatting domains which are not yet registered.

Authorized licensed use limited to: Institute of Information EngineeringCAS. Downloaded on August 19,2022 at 08:00:47 UTC from IEEE Xplore. Restrictions apply.

To address these limitations, Mohammad *et al.* [9] proposed a passive approach of identifying typosquatting domains based on conditional probability model. The approach passively analyzes DNS traffic records, and assumes that typosquatting domains attract visitors via direct type-in, have a high bounce rate (proportion of visitors who leave the site soon after arriving), and are very often followed by a visit to a more popular site with a similar name. The approach, although exactly finding typosquatting domains without needing to worry about coincidentally similar domains, can not prevent users from being harmed immediately, as it needs to collect traffics over time to build conditional probability model.

Our work is also a passive approach that does not actively inspect the contents of typosquatting domains, but it responses to typosquatting domains more timely.

2) Typosquatting Measuring: Moore and Edelman [3] generated approximately 938,000 domains estimated as typosquatting domains of the Alexa top 3,264 domains. By manually checking 2,195 samples, the authors find that domains within Damerau-Levenshtein distance 1 and fat-finger distance 1 to 2 of popular websites are very likely to be typosquatting domains. After crawling typosquatting pages, the authors find two main uses for traffics diverted to typosquatting domains: placing pay-per-click advertising and redirecting to other (often competing) domains. Moreover, large advertising networks willingly cooperate with typosquatters by showing advertising on typosquatting domains.

Szurdi *et al.* [7] investigated typosquatting registrations targeting `.com` domains that are in the "long tail" of the popularity distribution. Their experimental results show that, although the typosquatting saturation decreases for less popular authoritative domains, it is still at 40% near the Alexa one million rank and an estimated 20% of all .com domains appear to be typosquatting domains. The authors find that typosquatting domains of popular authoritative domains appear to change hands much more frequently than other domains. Mohammad *et al.* [9] proposed a method to quantify the harm experienced by humans caused by typosquatting domains.

V. Conclusions and Future Work

In this paper, we propose a novel approach named TypoEval, which is capable of evaluating typosquatting domains fast and accurately. Specifically, we exploit siamese neural networks to learn an embedding per domain and evaluate typosquatting domains by calculating the distance between vectors in Euclidean space. We validate our TypoEval approach on a real world data set. Experimental results show that our approach can improve the shortcomings of edit distance which is used by most of the previous work , and it is efficient and effective in evaluating typosquatting domains.

The siamese networks are trained such that the squared L2 distances in the embedding space directly correspond to domain similarity. We use recurrent networks to fully utilize the context information inside domains. A domain has small distance with its typosquatting domains and has large distance with unrelated domains. In that way, we can reduce the problem of typosquatting evaluation to a problem of vector distance calculation with deep learning. We can speed up the vectors operations by using GPU, and shorten the time of typosquatting evaluation.

Evaluating typosquatting domains is a necessary step to measure typosquatting phenomenon, and can protect Internet users from cyber crimes. The work in this paper evaluates typosquatting domains with the similarity of domain names. In the future, we would like to include both the analysis of the contents of web pages, as well as the analysis of features extracted at domain registration time and DNS features especially to improve the detection accurancy of our method.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant No.2016YFB0801003.

References

[1] S. Jeffrey, U. Shambhu, and M. Aziz, "The Landscape of Domain Name Typosquatting: Techniques and Countermeasures," in *Proc. IEEE International Conference on Availability, Reliability and Security*, 2016, pp. 347–358.

[2] V. T, J. W, and N. N, "Parking Sensors: Analyzing and Detecting Parked Domains," in *Proc. ISOC NDSS*, 2015.

[3] M. T and E. B, "Measuring the Perpetrators and Funders of Typosquatting," in *Financial Cryptography and Data Security*, 2010, pp. 175–191.

[4] "Typosquatting - Wikipedia," https://en.wikipedia.org/wiki/Typosquatting.

[5] W. Yi-Min, B. Doug, W. Jeffrey, V. Chad, and D. Brad, "Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting," in *Proc. 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006, pp. 31–36.

[6] C. Guanchen, F. J. Matthew, R. M. Pavan, K. S. Naveen, Y. Xin, and P. Vamsi, "Combating Typo-squatting for Safer Browsing," in *Proc. IEEE WAINA*, 2009, pp. 31–36.

[7] S. Janos, K. Balazs, C. Gabor, S. Jonathan, F. Mark, and K. Chris, "The Long "Taile" of Typosquatting Domain Names," in *Proc. USENIX Security*, 2014, pp. 191–206.

[8] A. Pieter, J. Wouter, F. Piessens, and N. Nick, "Seven Months' Worth of Mistakes: A Longitudinal Study of Typosquatting Abuse," in *Proc. ISOC NDSS*, 2015.

[9] K. Mohammad, Taha, H. Xiang, Z. Li, and K. Chris, "Every Second Counts: Quantifying the Negative Externalities of Cybercrime via Typosquatting," in *Proc. IEEE S&P*, 2015, pp. 135–150.

[10] L. Tingwen, Z. Yang, S. Jinqiao, Y. Jing, L. Quangang, and G. Li, "Towards Quantifying Visual Similarity of Domain Names for Combating Typosquatting Abuse," in *Proc. IEEE MILCOM*, 2016, pp. 770–775.

[11] P. Paolo, A. Davide, B. Battista, C. Igino, P. Luca, G. Giorgio, and R. Fabio, "Deepsquatting: Learning-based Typosquatting Detection at Deeper Domain Levels," in *Proc. Italian Association for Artificial Intelligence (AIIA)*, 2017, pp. 347–358.

[12] M. Jonas and T. Aditya, "Siamese Recurrent Architectures for Learning Sentence Similarity," in *Proc. AAAI*, 2016, pp. 2786–2792.

[13] "Euclidean distance - Wikipedia," https://en.wikipedia.org/wiki/Euclidean_distance.

[14] H. Raia, C. Sumit, and L. Yann, "Dimensionality Reduction by Learning an Invariant Mapping," in *Proc. CVPR*, 2006.

[15] W. Kilian, Q, B. John, and K. S. Lawrence, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," in *Proc. NIPS*, pp. 1473–1480.

[16] "Long short-term memory - Wikipedia," https://en.wikipedia.org/wiki/Long_short-term_memory.

[17] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," in *Studies in Computational Intelligence, Springer*, 2012.

[18] G. Klaus, S. Rupesh, K, K. Jan, S. Bas, R, and S. Jrgen, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 628–637, 2017.

Authorized licensed use limited to: Institute of Information EngineeringCAS. Downloaded on August 19,2022 at 08:00:47 UTC from IEEE Xplore. Restrictions apply.