DomainObserver: A Lightweight Solution for Detecting Malicious Domains Based on Dynamic Time Warping

Guolin Tan[1,2], Peng Zhang[2(✉)], Qingyun Liu[2], Xinran Liu[3], and Chunge Zhu[3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
tanguolin@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
pengzhang@iie.ac.cn
[3] National Computer Network Emergency Response and Coordination Center, Beijing, China

Abstract. People use the Internet to shop, access information and enjoy entertainment by browsing web sites. At the same time, cyber-criminals operate malicious domains to spread illegal content, which poses a great risk to the security of cyberspace. Therefore, it is of great importance to detect malicious domains in the field of cyberspace security. Typically, there are broad research focusing on detecting malicious domains either by blacklist or learning the features. However, the former is infeasible due to its unpredictability of unknown malicious domains, and the later requires complex feature engineering. Different from most of previous methods, in this paper, we propose a novel lightweight solution named DomainObserver to detect malicious domains. Our technique of DomainObserver is based on dynamic time warping that is used to better align the time series. To the best of our knowledge, it is a new trial to apply passive traffic measurements and time series data mining to malicious domain detection. Extensive experiments on real datasets are performed to demonstrate the effectiveness of our proposed method.

Keywords: Malicious domain · Detection · Passive traffic
Time series · Dynamic time warping

1 Introduction

The domain plays an important role in the operation of the Internet by providing convenience to identify Internet resources, such as computers, networks, and services. At the same time, cyber-criminals operate malicious domains to spread illegal information (phishing, malware, fraud and adult content, etc.). It is noted that close to one-third of all websites are potentially malicious in nature [16]. To make things worse, new malicious domains emerge on the Internet in endlessly.

Y. Shi et al. (Eds.): ICCS 2018, LNCS 10860, pp. 208–220, 2018.
https://doi.org/10.1007/978-3-319-93698-7_16

To provide a safe Internet environment for the vast number of Internet users, there are extensive research focusing on detecting malicious domains. Although a variety of methods have been applied to the problem of malicious domain detection with some successes, it is still very challenging for the reason that the patterns of malicious domains can be evolved by the adversaries to avoid being detected. Therefore, it is of great importance to discover the inherent patterns of malicious domains from the perspective of passive traffic measurements.

The passive traffic measurement is the process of measuring the amount and type of traffic on a particular network, which can help network operators better understand the properties of traffic. By using the passive measurement techniques, we can observe and reveal the inherent access patterns of domains. Moreover, this passive observation has the advantage that the cyber-criminals have no means to hide or change the access patterns of domains.

In this paper, we propose a novel lightweight solution, i.e., DomainObserver, to detect malicious domains. The highlight of our solution is that DomainObserver is based on dynamic time warping that is used to better align the time series. As far as we know, it is a new trial to apply passive traffic measurements and time series data mining to detect malicious domains.

To that end, the primary contributions of this paper are as follows:

1. First, we carefully observe and study the access patterns between Internet users and domains based on passive traffic measurements. This observation has the advantage that the cyber-criminals have no means to hide or change the patterns that we find in passive traffic measurements.
2. Based on the above analysis, we propose a novel lightweight method, i.e., DomainObserver, to detect malicious domains, which does not require constant crowd updates or complex feature engineering. To the best of our knowledge, no prior work in the literature applies this method to detect malicious domains.
3. Third, we evaluate our solution on real datasets collected from backbone networks. Experiment results show that DomainObserver is effective by achieving 83.14% accuracy and 87.11% F_1 score in the detection of malicious domains.

The rest of the paper is organized as follows. In the next section, we review the related work. In Sect. 3, we describes the time series datasets collected by using passive traffic measurements. Section 4 introduces the classification model based on time series data mining. We evaluate our approach on real world datasets in Sect. 5. Finally we conclude our work in Sect. 6.

2 Related Work

In the past few decades, the problem of malicious domain detection has been extensively studied. Much of these techniques can be broadly divided into two categories: (1) based on crowd-generated blacklists [2,11,20], (2) based on machine learning [5,10,13,17,19].

Generally, the typical methods based on blacklist are simple and efficient. However, the number of malicious domains drastically increased over time, which

requires constant updates generated by crowd. Moreover, it is almost impossible to maintain an exhaustive blacklist of malicious domains [16]. That is to say, these methods cannot predict newly registered malicious domains. In order to overcome these shortcomings, researchers proposed methods based on machine learning to detect malicious domains, which requires complex feature engineering to obtain discriminative features.

In [10], the authors proposes a method for identifying the C&C domains by using supervised machine learning and the feature points obtained from WHOIS and the DNS. To ensure the effectiveness of a URL blacklist, the authors of [17] proposes a framework called automatic blacklist generator (AutoBLG) that automatically expands the blacklist using passive DNS database and web crawler. [13] employs visible attributes collected from social networks to classify malicious short URLs on Twitter. The work of [19] presents a new deep learning framework (SdA) for detection of malicious JavaScript codes. In this method, 480 features are extracted from the JavaScript code. In a nutshell, all of these machine learning methods either need to extract complex features or require preconditions and specific data input.

The benefit from using our solution (DomainObserver) as opposed to existing methods is the fact that, our solution is not only more comprehensive in detecting a variety of malicious domains, but also more lightweight that does not require specific data input or complex feature engineering. The comparison is as shown in Table 1

Table 1. Qualitative comparison of existing work and our solutions.

	Types of malicious	Feature engineering	Data collection
AutoBLG	Unlimited	Yes	Active and passive
SdA	JavaScript	Yes	active
DomainObserver	Unlimited	No	Passive

3 Passive Traffic Measurements

To discover the different access patterns between benign and malicious domains, we tracked and studied the passive network traffic of thousands of the most popular domains. In this section, we first introduce the domain dataset that we collect from multiple reliable sources. Then we describe the time series data collected from backbone networks based on passive traffic measurements. Finally, we demonstrate how they can be used to detect interesting, anomalous domain access patterns.

3.1 Domain Dataset

As a starting point for our times series data collection we first construct domain dataset from multiple reliable sources. We used the Alexa top 20,000 global

sites [1] as the whitelist. For malicious domains we extensively investigated many blacklists of various malicious domains, such as malwaredomains.com [11], Antivirus [2] and Phishtank [14], etc. It is worth mentioning that we are conservative when constructing the domain dataset, since the ground truth has a great impact on the performance of the classifier. In order to determine whether the domain we collected is really malicious, for each domain, we validate it multiple times using different third-party platforms, such as 360 Fraud Reporting [15], Baidu Website Security Center [3]. Table 2 shows the verification results using [3]. Finally, we identified 1402 malicious domains and 1813 benign domains as the seed dataset.

Table 2. Verification results of Alexa top 20,000 domains using Baidu Website Security Center only.

Type	# of domains	Results
1	84	Fraud
2	5	Malware
3	196	Adult
4	23	Gambling
5	9509	Benign
6	10183	Unknown

3.2 Time Series Data

A time series is a series of data points indexed in time order. In the last decade, data mining of time series has attracted significant interest, due to the fact that time series data are present in a wide range of real-life fields [9]. In this paper, we combined passive traffic measurements and time series data mining to detect malicious domains. We categorize the collected time series data as three types: *access*, *users* and *entropy*, which will be explained later. According to our experimental evaluation, these data with high discriminative ability are sufficient to achieve a highly accurate classifier for malicious domain detection.

Passive traffic measurements play a crucial role in understanding the complex temporal properties of traffic [7]. In our passive traffic measurements, we count and aggregate the traffic traversing each Point of Presence (PoP) in the backbone networks to obtain time series data. We formalize the notion employed in time series data collection as follows:

Definition 1: *Traffic Measurement.* A traffic measurement M is a tuple (*timestamp, sip, domain*), representing a web request to the *domain*. Specifically, *timestamp* is the current time when a user start to access a website (represented by a URL), *sip* is the source IP address of the user, and *domain* is the second level domain name extracted from the URL.

Definition 2: *Time Window.* A time window W is a time bin (e.g., 1 h). When collecting time series data, we aggregate the traffic measurements by each time window.

Definition 3: *Time Series.* A time series $T = t_1, t_2, \ldots, t_n$ is a time-ordered set of n real-valued variables, where n is length of time series T. In our application, t_i is the aggregated number of traffic measurements in each time window.

Definition 4: *Time Series Dataset.* A time series dataset $D = T_1, T_2, \ldots, T_m$ is a collection of m such time series.

Definition 5: *Time Series Distance.* The distance d between two time series T and S is a function *dist(T,S)* that measures the similarity between T and S. Since time series distance plays a crucial role in time series data mining, we will introduce it in detail in the next section.

In order to collect time series datasets, we monitor web requests of each domain in the domain dataset (Sect. 3.1) on the backbone networks. Using passive traffic measurements, finally, we collect three types of time series data.

Access. The time series $Access = t_1, t_2, \ldots, t_n$, where t_i is the number of traffic measurements in a time window W.

User. The time series $User = t_1, t_2, \ldots, t_n$, where t_i is the number of distinct *sip* in a time window W.

Entropy. The time series $Entropy = t_1, t_2, \ldots, t_n$, where t_i is entropy of *sip* in a time window that is defined as follows.

$$E(sip) = -\sum_{i=1}^{I} \frac{n_i}{N} \log(\frac{n_i}{N}) \tag{1}$$

I is the number of distinct *sip*, n_i is the number of a *sip* and N is the total number of all *sip*.

It is important to note that in our passive traffic measurements, not all domains in domain dataset are observed, simply because some of the domains have never been accessed by users during passive measurement. Table 3 shows some basic information of our collected time series datasets over 94 h.

Table 3. Time series datasets.

Type	# of domains in dataset	# of observed domains	Start time	End time
Malicious	1402	1296	2017/11/22 11:45	2017/11/26 10:20
Benign	1813	373	2017/11/22 11:45	2017/11/26 10:20

For different types of websites, we find that the access patterns appearing in time series are different. For example, for pornographic websites, users are

more likely to access these websites at night. Figure 1 shows the different domain access patterns. We can see that the number of users accessing malicious domains usually reaches a significant peak around midnight, while there is no such phenomenon of benign domains. These potential domain access patterns are very easy to find when using passive traffic measurements. In the next section, we will introduce how we detect malicious domains using time series data mining.

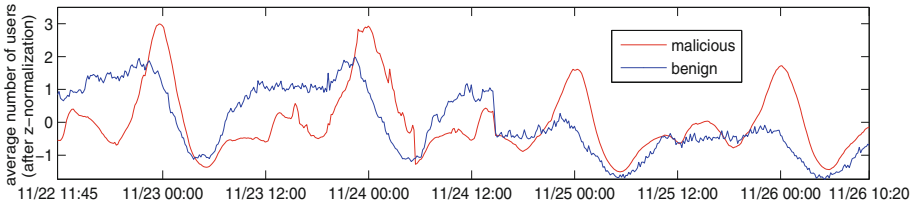


Fig. 1. The different access patterns of malicious and benign domains.

4 Time Series Classification

As mentioned in the previous section, time series distance plays a crucial role in time series data mining. Before we introduce how to detect malicious domains, we will first describe the time series distance used in our classification algorithm.

4.1 Dynamic Time Warping

There are plenty of distance measures used for evaluating similarity of time series, such as Euclidean distance (ED) [8], Dynamic Time Warping (DTW) [4]. Although it is simple and efficient, Euclidean distance is very sensitive to even slight misalignments. Inspired by the need to handle time warping in similarity computation, Dynamic Time Warping was proposed in order to allow a time series to be "stretched" or "compressed" to provide a better match with another time series [18]. That is to say, by warping a little to match the nearest neighbor, DTW can better measure the similarity of time series. The dynamic programming formulation of DTW is defined as follows.

$$D(i,j) = d(i,j) + min[D(i-1,j), D(i-1,j-1), D(i,j-1)] \tag{2}$$

That is, the DTW distance is the sum of the distance between current points and the minimum of the DTW distances of the neighboring points. In our work, we use DTW as the measure of distance because we found that there may be an advance or delay of the patterns. Figure 2 illustrates this temporal shifting of malicious domain access patterns.

In order to improve the computation efficiency, it is important to restrict the space of possible warping paths. Therefore, there is a variant of the DTW algorithm by adding a temporal constraint ω on the warping window size of

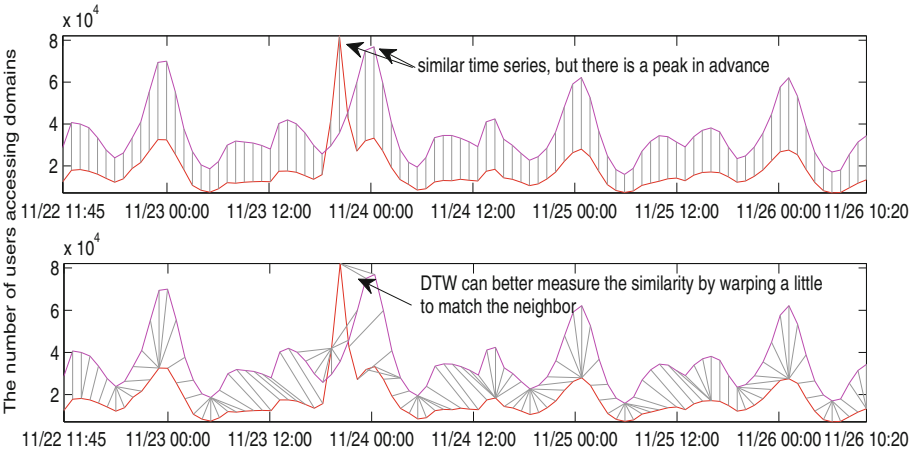


Fig. 2. A larger similarity distance measured by the Euclidean distance (top). Compared with ED, DTW can better measure the similarity by warping a little to match the nearest neighbor (bottom). Similarity is proportional to the sum of the pairwise distances (indicated by gray lines).

DTW, which is called constrained DTW. The details of the constrained DTW that is used in our paper are described in Algorithm 1 based on the dynamic programming approach.

Given two time series t and s, and time warping window ω, we initialize a two-dimensional DTW matrix whose value represents the DTW distance of the corresponding points within time series t and s (lines 2–10). Then we traverse the DTW matrix to calculate the DTW distances (lines 11–19). According to the time warping window ω, we determine the range of the DTW matrix that needs to be traversed (lines 12–13). The DTW distance is calculated according to Eq. 2 (lines 15–17). In order to allow meaningful comparisons between DTW distances of different DTW path lengths, length normalization must be used. The subroutine DTWPathLen (shown in Algorithm 2) returns the length of the DTW path (line 20). We calculate the final DTW distance using length normalization (line 21).

As we show in Algorithm 1, the function DTWPathLen is called to calculate the length-normalized DTW (NDTW for short) distance. For concreteness, we briefly discuss the DTWPathLen function in Algorithm 2 below.

Given the DTW matrix, the DTW path can be found by tracing backward in the matrix by choosing the previous points with the lowest DTW distance [4]. In line 5, we calculate the position of the neighboring points with the minimum DTW distance. From lines 7 to 14, we move the DTW path from the current position to the neighboring position with the minimum DTW distance until the start position. And the length of DTW path increases by one for each move in line 15.

Algorithm 1. Dynamic Time Warping

Input: time series t, s; warping window ω;
Output: DTW distance $dist$;

function DTWDistance(t, s, ω)
  $m \leftarrow size(t)$
  $n \leftarrow size(s)$
  $DTW \leftarrow zeros(m+1, n+1)$
  for $i = 1 \rightarrow m+1$ **do**
    for $j = 1 \rightarrow n+1$ **do**
      $DTW(i,j) \leftarrow Inf$
    end for
  end for
  $DTW(1,1) \leftarrow 0$
  for $i = 2 \rightarrow m+1$ **do**
    $lowerBound \leftarrow max(2, i-\omega)$
    $upperBound \leftarrow min(n+1, i+\omega)$
    for $j = lowerBound \rightarrow upperBound$ **do**
      $currentDist \leftarrow (t(i-1) - s(j-1))^2$
      $minDist \leftarrow min(DTW(i-1,j), DTW(i,j-1), DTW(i-1,j-1))$
      $DTW(i,j) \leftarrow currentDist + minDist$
    end for
  end for
  $pathLen \leftarrow$ DTWPathLen(DTW);
  $dist \leftarrow sqrt(DTW(m+1, n+1)/pathLen)$;
  return $dist$
end function

Algorithm 2. DTW Path Length

Input: DTW distance matrix DTW;
Output: DTW path length $pathLen$;

function DTWPathLen(DTW)
  $i \leftarrow row(DTW)$
  $j \leftarrow col(DTW)$
  $pathLen \leftarrow 0$
  while $i \neq 2$ **or** $j \neq 2$ **do**
    $[\sim, minIndex] \leftarrow min(DTW(i-1,j), DTW(i,j-1), DTW(i-1,j-1))$
    if $minIndex \equiv 1$ **then**
      $i \leftarrow i-1$
    else if $minIndex \equiv 2$ **then**
      $j \leftarrow j-1$
    else
      $i \leftarrow i-1$
      $j \leftarrow j-1$
    end if
    $pathLen \leftarrow pathLen + 1$
  end while
  return $pathLen$
end function

4.2 KNN Classification

We are now in a position to explain how to detect malicious domains. We consider the most common classification algorithms K-Nearest Neighbor (KNN) [6] in the time series mining community. In this method, for a new test instance, the nearest k neighbors are derived from the training dataset using similarity distance, and then it defines the class of the instance according to the class of the majority of its k nearest neighbors.

5 Evaluation

In order to examine the feasibility of our solution for detecting malicious domains, a series of experiments are carried out using the datasets introduced in Sect. 3.2. In our default experimental evaluation, we use the time series data of 70% domains as the training data while the remaining 30% as the testing data. Unless otherwise stated, we use KNN as the underlying classifying algorithm, and the final values are the averages of ten random runs.

5.1 Evaluation Metrics

In order to evaluate our method comprehensively, we utilize the well-known precision (P), recall (R), accuracy (A) and F_1 score to evaluate the performance of malicious domain detection. These evaluation metrics are functions of the confusion matrix as shown in Table 4.

Table 4. The confusion matrix of binary classification tasks.

	Predicted positive	Predicted negative
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

$$P = \frac{TP}{TP + FP} \tag{3}$$

$$R = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = \frac{2 * P * R}{P + R} \tag{5}$$

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

5.2 Results and Analysis

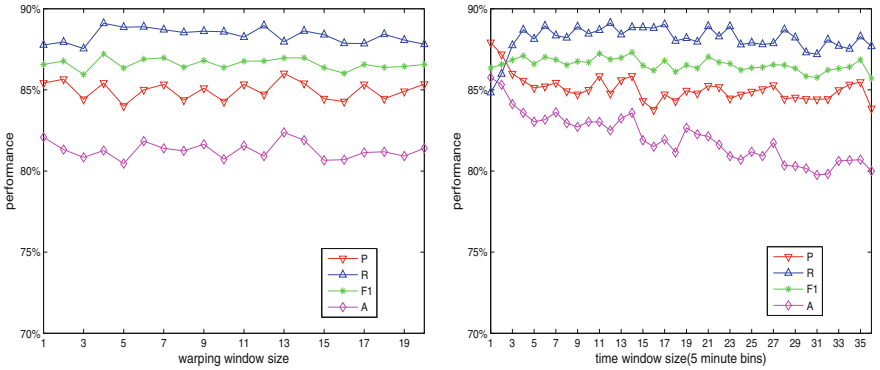
Varying Distance Measures: The first experiment is to evaluate the performance of different distance measures on three time series datasets. Table 5 shows the experimental results, from which several observations can be drawn. First of all, experiments on real datasets demonstrate that our solution based on dynamic time warping is effective in detecting malicious domains with 86.43% F_1 score and 81.58% accuracy. We thus believe the proposed method is suitable for malicious domain detection. Secondly, among these distance measures, the NDTW is better than the other two. This is because it is not sensitive to noise and misalignments in time, and able to handle local time shifting, i.e., similar segments that are out of phase (Fig. 2). As for the time performance, the ED distance is more efficient, because this distance measure is simpler than DTW and NDTW, which sacrifices the accuracy. Finally, we observe that there is no significant difference between datasets *Access*, *User*, and *Entropy*. For the sake of simplicity, in the following experiments, we use *User* as the default dataset and NDTW as the default distance measure.

Table 5. Performance comparison between different distance measures

Dataset	Distance	P(%)	R(%)	A(%)	F_1(%)	Avg time (s)
Access	ED	82.84	87.97	73.78	85.31	0.0001
	DTW	83.20	88.97	79.80	85.98	0.3428
	NDTW	83.77	89.41	**80.40**	**86.49**	0.6584
User	ED	87.20	85.02	79.30	86.09	0.0001
	DTW	84.86	87.92	80.32	86.35	0.5335
	NDTW	85.55	87.35	**81.58**	**86.43**	0.6512
Entropy	ED	87.70	82.48	79.00	84.89	0.0001
	DTW	88.11	84.89	80.26	85.45	0.5493
	NDTW	85.74	85.90	**80.72**	**85.70**	0.6722

Varying Warping Windows: The most important parameter of DTW is the warping window ω (see Algorithm 1), which enforces a temporal constraint on the warping range and has great influence on the experiment results. We will test it in the following experiments. The results of malicious domain detection are presented in Fig. 3(a). It can be clearly seen that the performance of malicious domain detection does not increase as the warping window ω increases. And when $\omega = 4$, it achieves the best F_1 score of 87.21%. This is because too wide warping window may introduce pathological matching between two time series and distort the true similarity [18]. Therefore, we set $\omega = 4$ in the following experiments.

Varying Time Windows: Another important parameter is the time window of time series (see Definition 2). As shown in Fig. 3(b), the smaller size of time window has a higher accuracy. This may be because the finer grained time series can better reflect the domain access patterns. While the F_1 score is not sensitive to the size of time window, which varies only in a narrow range from 85.71% to 87.32%.



(a) Varying Warping Windows (b) Varying Time Windows

Fig. 3. (a) Impact of warping window ω to the performance of malicious domain detection. (b) Impact of time window w to the performance of malicious domain detection.

Varying K-Nearest Neighbor: Finally, we conduct a set of experiments to evaluate the performance of our solution with different numbers of nearest neighbors. We perform experiments on two time window sizes, 14 and 24 respectively. Experimental results show that the KNN classifier gives the best F_1 score and accuracy for the value K = 10 over both two time window sizes, which we report in Table 6.

5.3 Discussion

In a sense, the approach taken here may appear surprising. Most malicious domain name detection methods are very complicated and heavy-weight, such as [5,10,12,13,17,19,20]. These methods require complex feature engineering and specific data input and even constant update of models. So we propose a lightweight solution that requires very few preconditions can be easily deployed, although it will slightly deteriorate the performance at an acceptable level.

Table 6. Performance comparison between different numbers of nearest neighbors.

K	Time window size $w = 24$					Time window size $w = 14$				
	P(%)	R(%)	F_1(%)	A(%)	Time(s)	P(%)	R(%)	F_1(%)	A(%)	Time (s)
1	84.90	83.95	84.41	75.70	319.15	85.68	86.79	86.22	80.12	542.03
2	87.88	79.57	83.50	71.90	312.72	87.53	81.65	84.47	76.26	571.93
3	84.37	86.62	85.46	79.02	315.55	85.52	87.37	86.43	82.18	576.50
4	86.35	84.20	85.25	78.02	313.30	86.34	85.58	85.95	80.44	584.49
5	83.70	88.29	85.92	80.24	312.91	85.07	88.96	86.97	82.64	585.85
6	85.50	87.12	86.28	79.92	310.79	86.32	86.76	86.53	82.76	584.55
7	84.40	89.45	86.84	81.14	310.01	83.93	90.14	86.91	82.12	582.32
8	85.15	87.71	86.39	80.48	309.86	84.59	88.35	86.41	81.50	584.57
9	83.57	89.66	86.49	80.80	310.32	84.01	89.57	86.69	82.20	575.36
10	85.43	89.33	**87.33**	**81.58**	310.28	85.12	89.23	**87.11**	**83.14**	572.11
11	83.41	90.16	86.65	81.10	323.93	83.30	91.13	87.03	82.06	574.61
12	83.89	89.53	86.61	80.56	312.64	84.14	89.78	86.86	82.26	572.83
13	83.37	90.85	86.94	80.64	322.02	82.80	90.79	86.60	81.54	569.33
14	84.25	90.12	87.08	81.16	337.83	83.83	89.94	86.76	82.24	571.26
15	82.54	90.27	86.22	80.68	333.79	82.89	91.43	86.93	81.44	572.30

6 Conclusion

In this paper, we have described a novel lightweight solution named DomainObserver for malicious domain detection, which does not require constant crowd updates or complex feature engineering. By using time series data mining, we apply DomainObserver to three different time series data collected in the actual network. Extensive experiments show that our method can effectively detect malicious domains by achieving 83.14% accuracy and 87.11% F_1 score.

Future work includes: (1) investigate how to improve the efficiency. This is especially needed for online deployment on real-time ISP networks. (2) study the misclassified samples to further improve the detection performance.

Acknowledgment. The author gratefully acknowledges support from National Key R&D Program 2016 (Grant No. 2016YFB0801300), National Natural Science Foundation of China (No. 61402464), and Youth Innovation Promotion Association CAS. And we also want to thank the anonymous reviewers for the valuable comments.

References

1. Alexa: Alexa top 1m. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip. Accessed 7 Nov 2017
2. Antivirus: Network Security Threat Information Sharing Platform. https://share.anva.org.cn/en/index

3. Baidu: Baidu Website Security Detection Platform. http://bsb.baidu.com/
4. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD Workshop, Seattle, WA, vol. 10, pp. 359–370 (1994)
5. Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: Exposure: a passive DNS analysis service to detect and report malicious domains. ACM Trans. Inf. Syst. Secur. (TISSEC) **16**(4), 14 (2014)
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
7. Duffield, N.: Sampling for passive internet measurement: a review. Stat. Sci. 472–498 (2004)
8. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD 1994. Citeseer (1994)
9. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 392–401. ACM (2014)
10. Kuyama, M., Kakizaki, Y., Sasaki, R.: Method for detecting a malicious domain by using WHOIS and DNS features. In: Third International Conference on Digital Security and Forensics (DigitalSec2016), p. 74 (2016)
11. Malware: Malware Domain Block List. http://www.malwaredomains.com/
12. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 1–18. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_1
13. Nepali, R.K., Wang, Y.: You look suspicious!!: leveraging visible attributes to classify malicious short URLs on Twitter. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), pp. 2648–2655. IEEE (2016)
14. OpenDNS: Phishtank. http://www.phishtank.com/
15. Qihu 360: 360 Fraud Reporting Center. https://110.360.cn/
16. Sahoo, D., Liu, C., Hoi, S.C.: Malicious URL detection using machine learning: a survey. arXiv preprint arXiv:1701.07179 (2017)
17. Sun, B., Akiyama, M., Yagi, T., Hatada, M., Mori, T.: Autoblg: automatic URL blacklist generator using search space expansion and filters. In: 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 625–631. IEEE (2015)
18. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. Data Min. Knowl. Discov. **26**, 1–35 (2013)
19. Wang, Y.: Cai, W.d., Wei, P.c.: A deep learning approach for detecting malicious Javascript code. Secur. Commun. Netw. **9**(11), 1520–1534 (2016)
20. Zhang, J., Porras, P.A., Ullrich, J.: Highly predictive blacklisting. In: USENIX Security Symposium, pp. 107–122 (2008)