Adaptive Malicious URLs Detection: Learning in the Presence of Concept Drifts

1st Guolin Tan
Institute of Information Engineering, CAS
University of Chinese Academy of Sciences
Beijing, China
tanguolin@iie.ac.cn

2nd Peng Zhang
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
pengzhang@iie.ac.cn

3rd Qingyun Liu
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
liuqingyun@iie.ac.cn

4th Xinran Liu
National Computer Network
Emergency Response and Coordination Center
Beijing, China
lxr@isc.org.cn

5th Chunge Zhu
National Computer Network
Emergency Response and Coordination Center
Beijing, China
zcg@cert.org.cn

***Abstract*—More people use the Internet to shop, access information, etc. But on the other hand, hackers implant malwares (e.g. Trojans, Worms, etc.) in the web pages to steal user information and acquire money illegally, which poses a great risk to the security of cyberspace and the privacy of users. Therefore, it is of great importance to detect malicious URLs in the field of cyberspace security. Different from most of previous methods, in this paper, we propose a method for online malicious URLs detection based on adaptive learning. By collecting the network traffic from backbone networks, we train machine learning models to detect the malicious URLs. But there is a serious problem in dynamically changing environments where the statistical properties of target variable change over time, which is known as concept drift. To address this problem, we apply a nonparametric test to correctly detect concept drifts in adaptive learning. Extensive experiments with different types of concept drifts are performed to demonstrate the feasibility of our proposed method on both artificial and real datasets. Our empirical study shows that this approach has good performance in detecting malicious URLs and concept drifts.**

***Index Terms*—malicious, traffic, adaptive learning, URL, concept drift**

I. Introduction

Malicious URLs are associated with web pages that are embedded malicious codes or contain illegal contents (e.g. phishing sites, spam, etc.). In fact, it is noted that close to one-third of all websites are potentially malicious in nature [1]. Hackers can implant malicious codes in web pages to steal user information or install malware, which poses a great risk to the security of cyberspace and the privacy of users. Therefore, it is of great importance to detect malicious URLs, which can also provide a good Internet environment for the vast number of Internet users.

The traditional method of malicious URLs identification is based on blacklist collected by user reports or manual judgments. When a new URL input comes, it queries the blacklist and checks whether the URL is in the blacklist. If so, then the URL is reported as malicious. This method is simple and efficient. However, there is an ever-growing number of malicious URLs that are not in the blacklist. For example, cyber-criminals may use a domain generation algorithm (DGA) to evade blacklists by generating new malicious URLs. Therefore, it is almost impossible to maintain an exhaustive blacklist of malicious URLs [1]. That is to say, this method cannot predict new malicious URLs.

In order to overcome these shortcomings of the method based on blacklist, researchers proposed methods based on machine learning to identify malicious URLs. The identification of malicious URLs is regarded as a binary classification task for two-class prediction: malicious and benign. The training process of machine learning is to find a best mapping from the d-dimensional feature vector space to output variable. Compared with the blacklist-based approach, this method has a good generalization ability to identify unknown malicious URLs.

Different from most of previous methods, our work focuses on finding malicious URLs buried within many benign URLs in massive network traffic. In this paper, we evaluate the effectiveness of our approach on real datasets. Experimental results show that malicious URLs can be correctly detected. On the other hand, benign URLs may change into malicious URLs. For example malicious codes are embedded in previous benign web pages by attackers. And malicious URLs may also change into benign URLs. Both of these changes mean that the data distribution has changed. This phenomenon of data distribution change in dynamically changing and non-stationary environments is called concept drift [2]. When concept drifts occur, the trained classification model cannot accurately represent the new data distribution, meaning the model cannot correctly predict the true class of URLs. In order to solve this problem, we apply a nonparametric test to correctly detect concept drifts with very low delay.

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

1) We introduce novel features with geographic information to identify malicious URLs. And these features can be obtained and processed quickly, which can be applied to real-time malicious URLs detection.
2) Based on our proposed approach, we have developed an online malicious URLs detection system, which can detect malicious URLs on backbone networks. We have verified the feasibility of our approach in this system.
3) In order to solve the problem of concept drift, we apply statistical test algorithm in our malicious URLs detection system, which can be applied to monitor the concept drift in non-stationary environments with lower delay.

The rest of the paper is organized as follows. In section 2, we review the related work. Section 3 describes the framework of our system and corresponding implementation details. We evaluate our approach over both artificial and real datasets in section 4. Finally we conclude with an overall discussion.

II. Related Work

In recent years, malicious URLs identification based on machine learning has been extensively studied. A malicious C&C domain detection method has been proposed in [3] by using supervised machine learning. However, in that paper, the classifier model needs to obtain host-based features from other servers, such as WHOIS information, DNS information, which to some extent will affect the real-time performance. [4] employs visible attributes collected from social networks to classify malicious short URLs on Twitter. In that method, features are extracted from Twitter using Twitter streaming APIs and labeled using VirusTotal and PhishTank. [5] proposed a method to detect malicious web pages using web contents as features (e.g. native JavaScript function, HTML document content etc.). It is obvious that this method is “heavy-weight”, because entire web contents are required to be obtained which is time-consuming. And it may also arise security problems, because the malicious code has been executed.

On the other hand, A serious problem with learning in many real-world domains is that the concept of interest may change constantly, e.g. there will be new malicious threats in the field of cyberspace security. To prevent deterioration in prediction accuracy because of concept drift, both active and passive solutions can be adopted. In passive solutions the classifier is retrained periodically and continuously which consumes a great deal of extra and useless computational cost (e.g. [6]). On the contrary, active solutions usually use statistical tests to detect concept drifts, and trigger retraining models only when concept drifts are detected (e.g. [7], [8]).

Although many machine learning methods and different features have been applied to the malicious web page identification with some success, there are several problems with current approaches:

1) What features to choose that can be processed quickly as input of classification model? Because the amount of web pages in reality is very large and continues to sharply increase. Efficient processing is a very important issue.

TABLE I
Partial destination port numbers of malicious and benign URLs

Malicious Port Numbers		Benign Port Numbers		
1988	8081	1010	1218	2000
2814	8181	1024	1234	2002
3128	8880	1188	1235	2010
3129	8888	1218	1303	2012
3690	9000	1234	1888	2016
4040	10078	1235	1935	2017
6666	11111	1303	1936	2086
8080	31288	1888	1937	2100

2) Based on the features selected above, it is still a difficult problem to choose which machine learning algorithm that can classify web pages accurately.
3) In non-stationary environments, how to efficiently detect concept drifts, so that the classification model can adjust to the change of distribution, and continue to run with high accuracy for a long time.

III. Application

In this section, we will introduce the framework of our system and corresponding implementation details of malicious URLs detection.

A. Features

Unlike previous work of others, our malicious URLs detection system is running on the backbone network. So the network traffic is collected on the campus backbone, and then features of accessing web pages are extracted from the traffic, such as source IP, destination IP, etc. We select these features for the following reasons.

1) *IP blocks*: Currently the IP addresses are assigned to different territories in corresponding blocks. That means IP addresses in the same territory usually have the same IP block. So the skewed distribution of malicious web pages over geographical location will be reflected on the IP address block. It is very reasonable to choose these features when designing features. We split the IP address into four segments according to their four bytes.
2) *Destination port*: In order to escape being identified, hackers may use different port numbers of malicious websites. The actual statistical results also verify this view. Table I shows the destination port numbers of different types of web pages we collected.
3) *Domain*: In many cases, the URLs of web pages collected from backbone network traffic may contain a lot of extraneous parameters, however, they actually point to the same malicious web page and their domain name is the same. So extracting the domain name from each URL as classification feature not only compresses the storage space but also saves classification time.

B. System Framework

In order to verify the feasibility of our approach, we have developed an online malicious URLs detection system, which detects malicious URLs on the backbone network. Our malicious URLs detection system consists of four modules (Feature Extraction, Training, Prediction and Concept Drift Detection). Figure 1 shows the framework of our system. And we will introduce these four modules in more detail in the following sections.

1) *Feature Extraction*: In order to train the classification model and predict malicious URLs, we need to extract features from the original traffic. We have developed the *Feature Extraction* module which can analyze network traffic and extract features that we need. After that, we label extracted features with 1 for malicious, and -1 for benign in combination with various blacklists. Table II shows the number of distinct values of features that we have extracted. These features will be used as input to the *Training* module and the *Prediction* module.
2) *Training*: Once the module of *Concept Drift Detection* outputs an alarm, which means concept drifts are detected, it will trigger retraining model using newly collected training set. However, in practice, there is a problem of class imbalance of the training set, that the benign web pages significantly outnumber the malicious web pages. This problem can seriously affect the performance of classifiers. We solve this problem by under sampling [9], so that the number of benign and malicious web pages are approximately equal. In addition, in this module, we evaluate the performance of different algorithms to select the optimal model.
3) *Prediction*: The *Prediction* module is deployed on the campus backbone network. After obtaining the optimal classification model from the *Training* module, the *Prediction* module use the classification model to identify whether a URL is malicious or not.
4) *Concept Drift Detection*: To enable our malicious URLs detection system to adapt to concept drifts and run continuously with high accuracy, we apply a nonparametric test to correctly detect concept drifts. In our system, we detect concept drifts only from partial feedback which is randomly sampled and manually labeled. The specific method for detecting concept drift is introduced in the next section.

C. Concept Drift Detection

Concept drift primarily refers to an online supervised learning scenario when the relation between the input data and the target variable changes over time [10]. Concept drift occurs commonly in the actual operating environment because dynamic environments often change unexpectedly. For example, the host IP of malicious URLs may changes, or hackers modify the URLs of malicious web pages to avoid detection. Figure 2 illustrates the change of concepts in terms of the features of the domain name and host IP. Different color circles represent different concepts (malicious or benign). In this case of non-stationary environments, learning a changing concept is infeasible. To address this problem, a learning system needs to be able to automatically detect concept drifts, and retrain the model with "fresh" data.

TABLE II
THE NUMBER OF DISTINCT VALUES OF FEATURES

Feature	Malicious	Benign	Total
SourceIP_1stByte	194	194	198
SourceIP_2ndByte	256	256	256
SourceIP_3rdByte	256	256	256
SourceIP_4thByte	256	256	256
DestinIP_1stByte	155	189	190
DestinIP_2ndByte	245	256	256
DestinIP_3rdByte	256	256	256
DestinIP_4thByte	254	256	256
DestinPort	24	354	365
DomainName	5601	16125	21600

Based on statistical test theory, [11] proposes a form of the *cumulative sum* (CUSUM) algorithm, which uses log-likelihood ratio to detect sequential procedures. If the indicator value is beyond the predefined confidence interval, a concept drift is suspected. However, this method requires that the probability density function of each procedure and the values of the parameter before and after the concept drift are supposed to be known, which is a quite unrealistic assumption for practical applications.

In order to solve these problems, we apply statistical test algorithm based on Wilcoxon Rank-Sum Test (WRST for short) [12]. The Wilcoxon rank sum test is a nonparametric test that can be used to determine whether two independent samples were selected from populations having the same distribution. In the WRST method, we can detect the concept drift by only maintaining two windows of observations. Let $w_1 = \{t_1, t_2, \cdots, t_n\}$ represents an observation window from time point 1 to n, the same, $w_2 = \{t_{n+1}, t_{n+2}, \cdots, t_{n+m}\}$ represents the second observation window from time point $n+1$ to $n+m$. Both t_i in w_1 and w_2 represent the observed values of the samples, $i = 1, 2, \cdots, n+m$. Here we use the accuracy as the observed value. Then sort and assign numeric ranks to all the observations (put the observations from both windows to one set), beginning with 1 for the smallest value. Now, add up the ranks for the observations which came from window w_1. Let r_1 equal to the rank sum of t_i when $t_i \in w_1$, correspondingly, r_2 equal to the rank sum of t_i when $t_i \in w_2$.

$$r_1 = \sum_{i=1}^{n+m} i * I(t_i \in w_1) \tag{1}$$

$$r_2 = \sum_{i=1}^{n+m} i * I(t_i \in w_2) \tag{2}$$

If the observations from w_1 and w_2 belong to the same distribution, then r_1 and r_2 should be approximately equal,

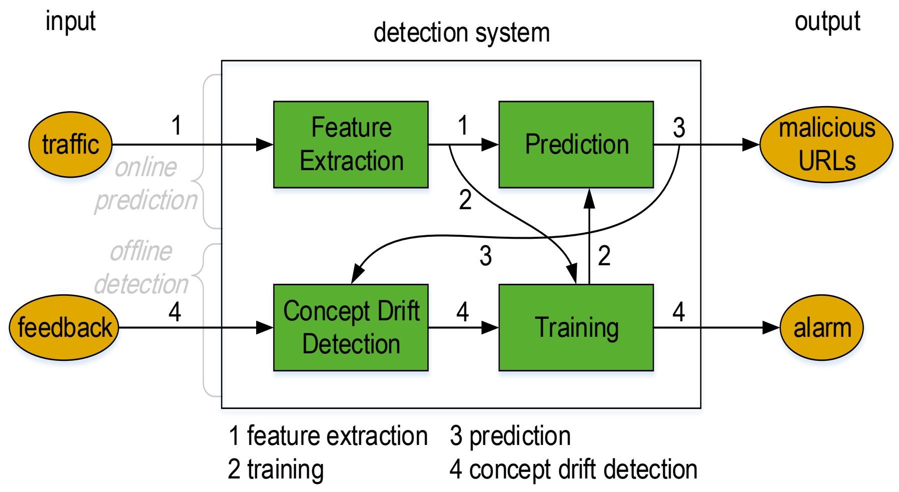

Fig. 1. The framework of our real-time malicious URLs detection system.

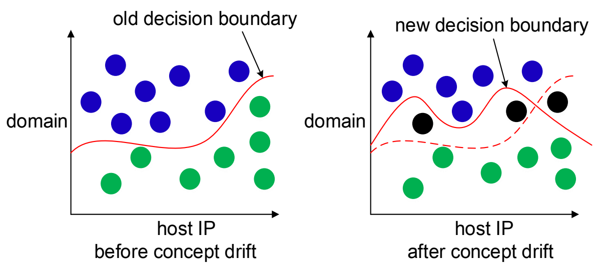

Fig. 2. Concepts drift on domain name and host IP. Different color circles represent different concepts of web pages that are malicious or benign. The black circles are pages that are misclassified after the concept drift.

otherwise a distribution change occurs. Moreover for large samples, r_1 is approximately Gaussian distributed. In that case, the mean μ and standard deviation σ of r_1 are given by [12] as follows:

$$\mu = \mathbb{E}(r_1) = \frac{n(n+m+1)}{2} \tag{3}$$

$$\sigma = \sqrt{D(r_1)} = \sqrt{\frac{nm(n+m+1)}{12}} \tag{4}$$

So we can use $Z = \frac{r_1 - \mu}{\sigma}$ as the test statistic and detect the concept drift by two-sided test. The specific steps to detect the concept drift are depicted in Algorithm 1.

Algorithm 1 RankSumDriftDetect

```
Input: the detection windows $w_1$, $w_2$;
       confidence interval $\alpha$;
Output: $drift$;
 1: Initialize $rankSum_1 = 0$ , $drift = -1$;
 2: n = length($w_1$) , m = length($w_2$)
 3: compute $\mu$ using formula (3)
 4: compute $\sigma$ using formula (4)
 5: $wTotal = w_1 \cup w_2$
 6: $wRanked = sort(wTotal)$
 7: for $i = 0 \to n + m$ do
 8:     if $wRanked(i) \in w_1$ then
 9:         $rankSum_1 = rankSum_1 + i$
10:     end if
11: end for
12: $z = \frac{rankSum_1 - \mu}{\sigma}$
13: if $|z| > \alpha$ then
14:     $drift = 1$
15: end if
16: return $drift$
```

IV. Evaluation

A. Data Set

In order to verify the feasibility of our malicious URLs detection system, we collect real HTTP request traffic traversing the Points of Presence (PoPs) in our campus network with a duration of 44 days. For each HTTP request, we extract the features we introduced earlier (see Section 3.1). Then we labelled 117210 malicious URLs using various blacklists, such as Phishtank [13] and Antivirus [14], etc. However the number of benign URLs is far more than the number of malicious URLs. Aiming to solve the problem of class imbalance, we use the "random under sampling" method [9] to remove majority (benign) samples from the original data set, which makes the number of malicious URLs and benign URLs approximately equal.

Secondly, in order to evaluate the performance of our concept drift detection algorithm, we compared the WRST algorithm and the CUSUM algorithm over the artificial dataset. This artificial dataset is constituted of independent and identically distributed samples following a Gaussian distribution with mean value 0 and variance 1 before the concept drift. After the concept drift, it takes the mean value 1 instead.

B. Evaluation Metrics

It is now well known that error rate is not an appropriate evaluation criterion when there are class imbalance or unequal costs [15]. For example, in the binary classification tasks, the majority class account for 99% of the total. If the model classifies all samples into the majority class, the accuracy of the model can achieve 99%. However, what we care about (the minority class) has not been identified. In this case, we use precision, recall and F_1-score et al. to measure the performance of malicious URLs identification.

To measure the performance of concept drift detection algorithms, one of the most popular criteria *average run length* (ARL) was proposed in [16], which is defined as the expected number of samples before an concept drift is detected:

$$ARL = \mathbb{E}(n_d) \tag{5}$$

where n_d is the detection time of the drift detection algorithm. In this paper, we will use the ARL and accuracy to evaluate the performance of concept drift detection.

C. Experimental Settings

First, in order to validate the effectiveness of our proposed method, we evaluate the performance of the model in detection malicious URLs upon 6 machine learning algorithms. For each machine learning algorithm, we perform a tenfold cross validation. The whole cross-validation process is repeated for ten times, and the final values from this method are the averages of these ten cross-validation runs. All of these machine learning algorithms are implemented in spark machine learning library MLlib [17]. We summarize the parameters of these machine learning algorithms in detail as follows:

1) Decision Tree (abbreviated as DT): it uses Gini impurity as metric criterion to split candidate training subset. The maximum depth of decision tree is set to 5 and 10.
2) Gradient-boosted tree (abbreviated as GBT): is a popular classification and regression method using ensembles of decision trees. GBT iteratively trains decision trees in order to minimize a loss function. The maximum depth of decision tree is set to 5 and 10, and both the numbers of iterations are set to 100.
3) Linear Support Vector Machine (abbreviated as LSVM): is trained with an L2 regularization. The number of iteration is set 100 and 1000.
4) Logistic regression (abbreviated as LR): is widely used to predict binary classification tasks. It is a linear method and the standard feature scaling and L2 regularization are used by default.
5) Naive Bayes (abbreviated as NB): is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions between the features. The parameter of Laplace smoothing is set to 1.
6) Random Forests (abbreviated as RF): are ensembles of decision trees described above. The maximum depth of decision tree is set to 5 and 10.

Next, in order to evaluate the performance of the concept drift detection algorithm, we simulate two types of concept drift scenarios [7] [18] on the real data set. Both in the simulated two types of concept drift scenarios, the test set is randomly split into 20 batches of equal size containing 1172 samples each.

In the first scenario (sudden concept drift), the concept drift suddenly occurs in batch 11, where all malicious URLs become benign, and the original benign URLs become malicious.

In the second scenario (incremental concept drift), the incremental concept drift slowly occurs in the batch 11 to the batch 20. In each of these batches, some of the malicious URLs become benign, and some of the original benign URLs become malicious. The speed of the drift increases from 0 to 1 with step length 0.2.

D. Results and Analysis

The first experiment is to evaluate the effectiveness of different classification algorithms. Table III shows the experimental results, from which several observations can be drawn. First of all, according to the experimental results, we found that both DT, GBT and RF algorithms significantly outperform the other algorithms. Therefore, we divide these algorithms into two groups. The first group contains "good" performance algorithms that all evaluation metrics are greater than 85% except for the false positive rate. The second group contains "bad" performance algorithms that are slightly better just randomly classification. Note that both the LSVM and LR are linear models. We argue that the linear model is not suitable for malicious URLs identification. Second, in terms of both F_1-score and precision, GBT has the best performance than all other compared models. Both F_1-score, accuracy and recall of GBT are greater than 96%. We thus believe the proposed method is practically attractive and suitable for the application of malicious URLs identification.

In order to evaluate the real-time processing efficiency of our approach, we have implemented and deployed it based on Apache Spark at each Point of Presence (PoP) in our campus network. Our approach can process nearly 3 billion of HTTP requests per day. That is to say, it can process 34 thousand records per second on average. This is much higher than the performance of the method of [19] that processes an average of 6.3 thousand records per second.

Finally we conducted a set of experiments to evaluate the performance of the method of concept drift detection. The method was tested over both artificial and real datasets. The results are presented in Fig. 3. It can be clearly seen that the detection delay of CUSUM is much higher than that of WRST. In addition, as shown in Table IV, the accuracy of the WRST is 1, which is significantly better than the 56.32% achieved by the CUSUM on artificial dataset. Note that all these parameters of the CUSUM algorithm are supposed known, which is a quite unrealistic assumption for practical applications [11], that is why we did CUSUM experiments only on artificial data.

CONCLUSION AND FUTURE WORK

In this paper, we propose a real-time and adaptive malicious URLs detection method. Experimental results show that our

TABLE III
EVALUATION OF THE EFFECTIVENESS OF DIFFERENT CLASSIFICATION ALGORITHMS.

Algorithm	Measures					
	FPR(%)	Accuracy(%)	Precision(%)	Recall(%)	F_1(%)	Parameters
DT	12.307±0.038	91.131±0.035	88.405±0.147	94.581±0.036	91.382±0.078	maxDepth=5
	10.374±0.050	93.821±0.034	90.371±0.120	98.014±0.022	94.033±0.063	maxDepth=10
GBT	8.034±0.034	92.507±0.015	91.981±0.115	93.072±0.031	92.518±0.052	maxDepth=5
	3.530±0.021	**96.923±0.016**	**96.458±0.058**	**97.382±0.024**	**96.917±0.033**	**maxDepth=10**
RF	12.632±0.279	91.569±0.112	88.357±0.271	95.766±0.121	91.876±0.123	maxDepth=5
	10.078±0.070	94.133±0.045	90.645±0.124	98.342±0.015	94.332±0.067	maxDepth=10
LSVM	68.513±4.398	50.892±0.338	39.457±1.816	69.962±4.555	47.303±3.054	Iterations=100
	51.262±4.443	51.849±0.463	39.836±1.723	54.552±4.622	39.746±3.174	Iterations=1000
NB	85.355±0.227	55.523±0.310	53.060±0.367	96.529±0.106	68.390±0.281	lamda=1
	85.260±0.220	55.565±0.293	53.089±0.350	96.530±0.103	68.417±0.263	lamda=100
LR	41.904±0.105	63.616±0.109	62.255±0.117	69.128±0.143	65.511±0.127	

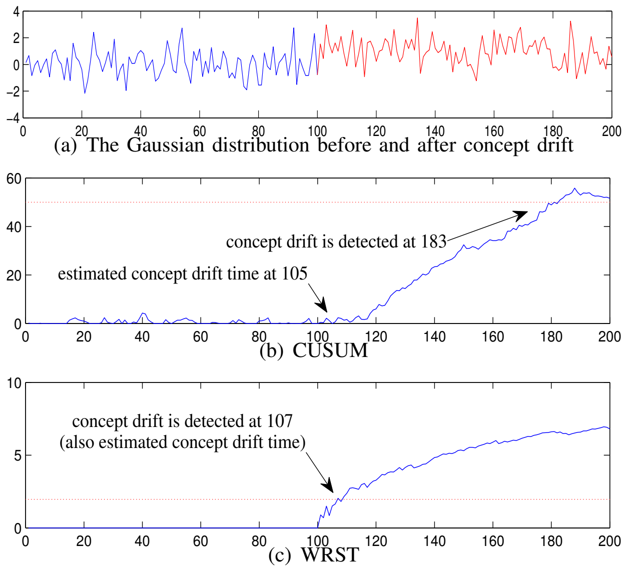

Fig. 3. (a) The Gaussian distribution with the mean value 0 before and 1 after the concept drift. (b) The concept drift is detected with the method of CUSUM at n_d = 183, leading to a detection delay of $n_d - n_c$ = 83. (c) The concept drift is detected with the method of WRST at n_d = 107, leading to a detection delay of $n_d - n_c$ = 7.

TABLE IV
THE PERFORMANCE OF DIFFERENT ALGORITHMS ON BOTH ARTIFICIAL AND REAL DATASETS.

Dataset	Method	ARL	Delay	Accuracy(%)	Parameters
artificial	CUSUM	185.65	85.65	56.32	h=50
	WRST	102.14	2.14	100	z=1.96
real	WRST	102.95	2.95	100	z=1.96

method can effectively detect malicious URLs by achieving 96% F_1-score and accuracy. We have deployed it at each Point of Presence (PoP) in our campus network, which can process 34 thousand records per second on average. On the other hand, the network environment is non-stationary, malicious and benign URLs may transform with each other. In order to adaptively and efficiently detect concept drifts, we apply nonparametric test algorithm based on rank-sum test that can detect the concept drifts with a very low delay.

We plan to extend our work to improve the efficiency. This is especially needed for online deployment on real-time ISP networks. We also plan to 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.

REFERENCES

[1] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious url detection using machine learning: A survey," *arXiv preprint arXiv:1701.07179*, 2017.
[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.
[3] M. Kuyama, Y. Kakizaki, and R. Sasaki, "Method for detecting a malicious domain by using whois and dns features," in *The Third International Conference on Digital Security and Forensics (DigitalSec2016)*, 2016, p. 74.
[4] R. K. Nepali and Y. Wang, "You look suspicious!!: Leveraging visible attributes to classify malicious short urls on twitter," in *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. IEEE, 2016, pp. 2648–2655.
[5] Y. Wang, W.-d. Cai, and P.-c. Wei, "A deep learning approach for detecting malicious javascript code," *Security and Communication Networks*, vol. 9, no. 11, pp. 1520–1534, 2016.
[6] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[7] Y. Sakamoto, K.-I. Fukui, J. Gama, D. Nicklas, K. Moriyama, and M. Numao, "Concept drift detection with clustering via statistical change detection methods," in *Knowledge and Systems Engineering (KSE), 2015 Seventh International Conference on*. IEEE, 2015, pp. 37–42.

[8] A. Meroño-Peñuela, C. Guéret, R. Hoekstra, S. Schlobach *et al.*, "Detecting and reporting extensional concept drift in statistical linked data," in *1st International Workshop on Semantic Statistics (SemStats 2013), ISWC. CEUR*, 2013.

[9] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[10] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.

[11] P. Granjon, "The cusum algorithm - a small review," *Saiga*, 2013.

[12] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[13] OpenDNS, "Phishtank," http://www.phishtank.com/.

[14] Antivirus, "Network security threat information sharing platform," https://share.anva.org.cn/index.

[15] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[16] A. R. Kamat, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[17] "Machine learning library (mllib) guide," http://spark.apache.org/docs/2.1.0/ml-guide.html.

[18] I. Zliobaite, "Learning under concept drift: an overview," *CoRR*, vol. abs/1010.4784, 2010. [Online]. Available: http://arxiv.org/abs/1010.4784

[19] L. Watkins, S. Beck, J. Zook, A. Buczak, J. Chavis, W. H. Robinson, J. A. Morales, and S. Mishra, "Using semi-supervised machine learning to address the big data problem in dns networks," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*. IEEE, 2017, pp. 1–6.