2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications

MalFilter: A Lightweight Real-time Malicious URL Filtering System in Large-scale Networks

1st Guolin Tan
Institute of Information Engineering, CAS
University of Chinese Academy of Sciences
Beijing, China
guolintan@qq.com

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

6th Li Guo
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
guoli@iie.ac.cn

***Abstract*—With the rapid development of communication technologies, the broadband-speed has been greatly improved, resulting in a sharp increase in global Internet traffic. For this reason, the traditional malicious URL detection technologies based on web content analysis have encountered a large performance bottleneck. Therefore, it is of great importance to filter highly suspicious malicious URLs in advance, before feeding these URLs to malicious URL detection systems. This paper explores how to filter highly suspicious malicious URLs in large-scale real-time networks, so as to reduce the processing pressure of back-end malicious URL detection system based on content analysis. By applying lightweight features, our system can filter malicious URLs in real time while achieving a high recall. Extensive experiments are performed to demonstrate the feasibility of our proposed method on real datasets. The practical deployment of our system on backbone networks shows that our approach has good performance in filtering malicious URLs.**

***Index Terms*—malicious, URL, filtering, traffic**

I. Introduction

With the rapid development of communication technologies, the broadband-speed has been greatly improved, resulting in a sharp increase in global Internet traffic. According to the latest report of Cisco, IP traffic will grow at a Compound Annual Growth Rate (CAGR) of 24 percent from 2016 to 2021. And global IP traffic will increase nearly threefold over the next 5 years [1].

On the other hand, cybercriminals constantly change and generate new malicious URLs which brings many new challenges to malicious URL detection systems. For example, malicious URL blacklists are getting bigger and bigger, and more and more data need to be processed in a timely manner. Therefore, it is necessary to mine and filter out highly suspicious malicious URLs from the massive data to reduce the load on existing malicious URL detection systems.

Due to the reasons mentioned above, the traditional malicious URL detection technologies based on web content analysis have encountered a large performance bottleneck. For example, the method [2] needs to download the entire webpage using a monitored Firefox web browser (Selenium). And finally it builds a high-dimensional vector containing 212 features from the data sources in the webpage, which is very time-consuming and not suitable for large-scale real-time processing.

To mitigate these issues, this paper explores how to filter highly suspicious malicious URLs in advance, before feeding these URLs to malicious URL detection systems. By adding this pre-filtering operation, we can greatly reduce the processing pressure of back-end malicious URL detection system based on content analysis that is heavy-weight. Different from most of previous methods, our work focuses on filtering malicious URLs buried within massive benign URLs from the perspective of network traffic. And after in-depth study of the network traffic, we elaborately design lightweight features that are suitable for online real-time malicious URL filtering.

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

(1) To the best of our knowledge we are the first to apply the guilt-by-association principle in designing of features to filter malicious URLs. Moreover, all the features are collected from the network traffic and are lightweight.
(2) In order to validate our method, we developed an online malicious URL filtering system. By using only lightweight features, our system can filter malicious URLs in real time while achieving a high recall.
(3) Extensive experiments are performed to demonstrate the feasibility of our proposed method on real datasets. The practical deployment of our system on backbone networks shows that our approach has good performance in filtering malicious URLs.

The rest of the paper is organized as follows. In Section 2, we review the related work. We introduce the methodology of designing and selecting features in Section 3. Section

978-1-7281-1141-4/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCloud.2018.00089

4 describes the framework of our malicious URL filtering system and corresponding implementation details. Then we evaluate our approach over real datasets in Section 5. Finally we summarize the paper and discuss future work in Section 6.

II. Related Work

In the last decades, the detection of malicious URLs has attracted a large number of researchers. Among these numerous research methods, the most attractive methods are based on machine learning [3] [2] [4] [5] and graph inference [6] [7] [8] [9] [10].

Machine learning. The application of machine learning in the field of cyberspace security has achieved great success. However, most of these methods need to extract heavy and complicated features to detect malicious URLs. The machine learning features come from different sources spanning across web content, search engine results, WHOIS information and DNS traffic. The authors of [3] trained hidden Markov models from inclusion sequences on web pages to detect malicious third-party content through a set of modifications to the Chromium browser. Samuel et al. [2] use 212 features extracted from multiple sources to build Gradient Boosting [11] model to detect phishing sites. Another work of Samuel et al. leverages search engine query data from Google and Yahoo to compute intra-URL relatedness as machine learning features [4]. However, this method takes an average processing time of 0.77 seconds per URL, and most of the time is spent on the requests to search engines. Based on passive DNS traffic analysis, Exposure applies 15 unique features to detect malicious domains [5]. Similarly, our system uses passive HTTP traffic to filter out malicious URLs, but our features are more lightweight and more suitable for real-time computing.

Graph inference. In addition to machine learning, there are emerging methods for detecting malicious URLs based on graph inference. The principle of graph inference is that malicious URLs are more likely to be accessed by the same or similar users. The paper [9] calls this principle as guilt-by-association, which is widely used in collaborative filtering in the field of recommendation systems. Compared with machine learning based methods, this type of methods do not require complicated feature engineering and use less ground truth. By exploring the coexistence of domain cache-footprints, DNSRadar [6] uses link analysis techniques to infer maliciousness likelihood of unknown domains. The authors of [7] construct a host-domain graph from HTTP proxy logs, and then use belief propagation to detect malicious domains. GANG [9] detects fraudulent users in directed online social networks using graph inference algorithm. In fact, although the accuracy of these methods is relatively high, graph inference is very time-consuming when there are tens of millions of users and domain names, so it is more suitable for off-line detection.

Malicious URL filtering. From a large-scale network traffic standpoint, malicious URL detection methods based on deep content analysis are usually very time-consuming. For example, EXPOSURE [5] detects abrupt changes as a feature from time series that require at least 8 hours to build, which is unacceptable in network traffic applications that require real-time processing. As a complement to deep content analysis methods, [12] and [13] quickly filter out most of the benign URLs before they are processed. The authors of [14] apply semi-supervised machine learning to filter out non-suspicious network traffic. Although these methods have achieved some good results, there are still unsatisfactory shortcomings. For example, [12] needs to download the entire web page to extract features. And it may also arise security problems, because the malicious code has been executed in the browser.

On the other hand, another thorny problem, a.k.a. class imbalance, is that the proportion of malicious URLs in actual network traffic is very small [9], [15]. This is why the performance of fraudulent users detection in [9] and [15] is not highly effective. Clearly this problem will make it so difficult to detect malicious URLs in the wild, just like looking a needle in a haystack. However, this problem is often overlooked in most existing work [4], [10], [12], [16]. In their experiments, the ratio of malicious URLs to benign URLs is always approximately equal by well constructing datasets. To mitigate this problem, we need to quickly filter malicious URLs from large volumes of network traffic.

III. METHODOLOGY

In order to quickly filter malicious URLs from the perspective of network traffic, we extensively study the characteristics of HTTP traffic to predict highly suspicious malicious URLs. In this paper, to the best of our knowledge we are the first to apply the guilt-by-association principle in designing of machine learning features. Specifically, we elaborately select lightweight machine learning features, taking into account both the servers hosting URLs and their access users together. Because malicious URLs are more likely to be accessed by the same or similar users. These features considered from the user's point of view are effective, and is not taken into account by existing work when designing features. By doing so, we can filter out the overwhelming HTTP traffic of benign URLs while achieving a high recall rate. Next we will introduce the features in detail and the reasons for doing so.

A. Server-based Features

(1) Server IP. The server IP is the IP address of the web page URL we extracted from network traffic. When mapping the server IP to geographic location, we found that the ratio of malicious server IPs in different regions is skewed. Fig. 1 shows this kind of geographically skewed distribution of malicious server IPs. Unlike the previous work [5], [12], [17], we did not map the server IP to a geographic location when extracting the features. Instead, we divided the server IP into four parts by byte as our features, because IP addresses are actually assigned to different regions according to the IP address block. The benefit is that it saves time compared to mapping the entire IP into the corresponding city.

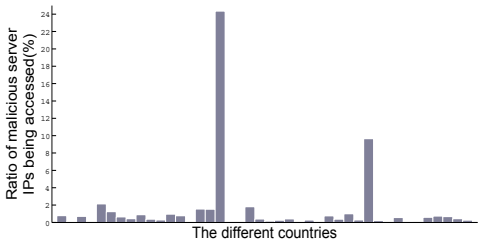

Fig. 1: Geographically skewed distribution of server IPs. To avoid conflicts of interest, We remove the names of these countries.

(2) Server port. Different network applications usually use specific server ports. To evade discovery, cybercriminals may intentionally set the server port to a specific port number. However, it is self-defeating that a particular server port will disclose the identity information of malicious services. Therefore the server port is also a discriminative feature.

B. User-based Features

(1) User IP. Based on the fact that malicious URLs are more likely to be accessed by the same or similar users, we propose using user IP as machine learning feature. To the best of our knowledge, it is the first time this feature is used to filter malicious URLs. By combining bidirectional server IP and user IP characteristics in machine learning, we can obtain the accuracy of the graph inference based methods, but also avoid the shortcoming of time consuming. The same operation as server IP, we divided the user IP into four parts by byte as our features. Fig. 2 demonstrates the heat map of user IPs accessing malicious URLs in geographical location. For the sake of simplicity, we only show the IP addresses from China.

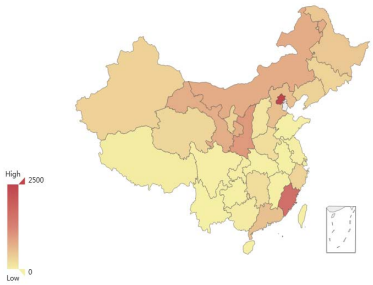

Fig. 2: The heat map of user IPs accessing malicious URLs.

(2) Access time. We observed the amount of malicious URL accessed in different hours of a day. We found that the amount of visited malicious URL typically peak in the afternoon or in the evening. Fig.3 is the result of our observation. We record the hour of a URL being requested as a feature.

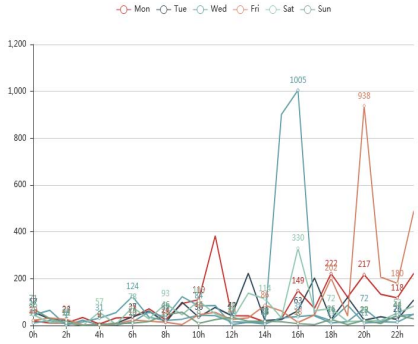

Fig. 3: The amount of malicious URLs accessed during different time intervals.

C. URL-based Features

In order to better understand the characteristics of URL, we analyze the components of a URL. A typical URL usually contains the following five parts.

URL = [protocol://]domain[:port][/path][?parameters]

A non-empty protocol component followed by a colon (:) and two slashes (//). The most popular protocols of website are HTTP and HTTPS. In this paper we focus on the HTTP traffic and will discuss HTTPS related issues later. The domain is a fully qualified domain name of a network host on which the port and path are accessible. Consider an example URL: http://avbus55.com:8998/index_movies.php?op=view&class=2 Its components include the following:

- protocol=http
- domain=avbus55.com
- port=8998
- path=index_movies.php
- parameters=op=view&class=2

(1) URL length. Cybercriminals can use the DGA algorithm to generate massive malicious domain names. The URL length is a lightweight feature compared to other deep content analysis methods that require obtain and analyze the entire page content. And this feature has been proven to be effective in other work [10], [13].

(2) Domain length. We found that the domain length of a malicious URL is usually larger than a benign URL. A phisher has full control over the subdomains portion and can set it to any value [2].

(3) Is IP. It is very common for malicious websites to use IP as domain name directly, which can avoid the trouble of registering domain name and being censored. This feature holds a boolean value to specify whether the domain is IP.

(4) Path length and depth. Websites of different services have differences in their website URLs. We take the length and depth of the path in the URL as features. The length is the number of characters in the path, and the depth is the number of slash in the path.

(5) Parameter length and number. Some websites generate parameters of a specific length and number in the URL, which is also a distinguishable feature, such as http://59.120.24.51:5011/vod/c18ebcf68b1aad4830296f82607b9c73.mp4?starttime=0&check=0bfef5cdfe1151fe6f2654efe8beca3d&timestamp=1523940678654. The 32-byte value of the parameter "check" in the above URL may be the value after using the MD5 hash.

D. Referer-based Features

In addition to extracting features from the accessing URL, we also extracted the referer field from the traffic. The referer can identify the address of the webpage that linked to the resource being requested. For the referer URL we extract the same features as the accessing URL, so there are also 7 referer-based features.

In the end, we summarize all the 24 features that are grouped into four categories in Table I.

TABLE I: Summary of extracted features.

Feature Set	#	Feature Name	Description
Server-based Features	1	SvrIP1	First byte of server IP
	2	SvrIP2	Second byte of server IP
	3	SvrIP3	Third byte of server IP
	4	SvrIP4	Fourth byte of server IP
	5	SvrPort	The server port
User-based Features	1	SrcIP1	First byte of source IP
	2	SrcIP2	Second byte of source IP
	3	SrcIP3	Third byte of source IP
	4	SrcIP4	Fourth byte of source IP
	5	AccTime	User accessing time
URL-based Features	1	URLLen	The length of URL
	2	DomLen	Length of domain
	3	IsDomIP	Whether domain is IP
	4	PathLen	Length of path in URL
	5	PathDep	Depth of path in URL
	6	ParaLen	Length of parameters
	7	ParaNum	Number of parameters
Ref-based Features	1-7	Similar to URL	Similar to URL-based features

IV. FILTERING SYSTEM

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

Based on the proposed methodology, we implemented our online malicious URL filtering system with three modules (PacketParser, Training and Filtering). Fig. 4 presents the framework of our system. And we introduce these three modules in more detail in the following sections.

1) *PacketParser.* As mentioned earlier, this paper focuses on filtering malicious URLs from the perspective of network traffic. And all of our features are obtained from the header fields of network packets. To achieve this goal, we developed a full stack packet parsing module—*PacketParser*. We deploy this module at Points of Presence (PoPs) of ISP networks. By passively parsing the header fields of network packets, we obtains the server IP, user IP, etc. as features. After that, we label extracted features with 1 for malicious, and 0 for benign using existing malicious URL blacklist. In particular, considering privacy issues, we have anonymized all these features.
2) *Training.* The problem of malicious URL filtering is a binary classification task. Let $\vec{X} = \{x_1, x_2, \cdots, x_n\}$ represents a feature vector of a training sample, and $Y = \{1, 0\}$ is the corresponding label where $y_i = 1$ represents malicious and $y_i = 0$ represents benign. To maximize the distinction between malicious and benign URLs, we evaluated the effectiveness of several machine learning models on our experimental datasets. According to our experimental results, we find that linear classifiers are not suitable for malicious URL detection, such as Naive Bayes, Logistic regression (reported in Section V-D). Among these classifiers, the AdaBoost classifier not only has a good performance in filtering malicious URLs, but also has good time performance. So we choose the AdaBoost classifier as our *Filtering* module.
3) *Filtering.* We implemented our malicious URL filtering module on Apache Spark, a fast cluster computing system, and deployed it on a cluster of 26 nodes. For each URL that is accessed, we use the trained model to calculate the malicious and benign probabilities of the URL, and then filter highly suspicious malicious URLs to feed deep content detection.

V. EVALUATION

In this section, the two questions we are most concerned about are: 1) Can our method effectively filter malicious URLs while achieving high recall rate? 2) What is the time performance of our malicious URL filtering system? To answer this two questions, we have done extensive experiments on real datasets to illustrate the effectiveness of our approach.

A. Datasets

We deployed our *PacketParser* module at Points of Presence (PoPs) of ISP networks. By passively collecting and parsing network packets, we extracted all the features (see Table I). Then we labeled the collected features with 1 for malicious, and 0 for benign using URL blacklist. Table II gives information about the experimental datasets.

B. Evaluation Metrics

In our malicious URL filtering application, the two most concerned metrics are filtering rate (F) and recall rate (R). And we hope to achieve low filtering rate but also high recall rate. The definition of filtering and recall rate are as follows.

$$F = \frac{\#\ of\ classified\ malicious\ URLs}{total\ number\ of\ URLs\ to\ classify} \quad (1)$$

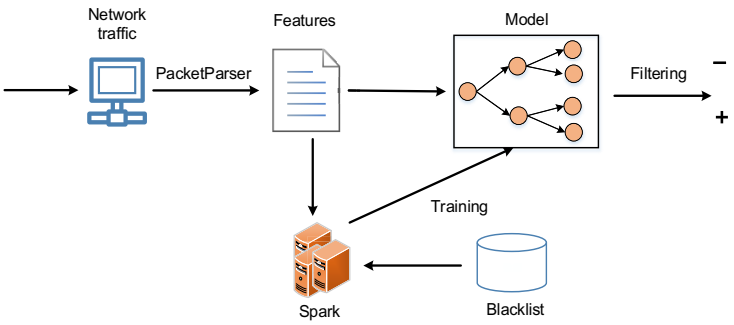

Fig. 4: Framework of real-time malicious URL filtering system.

TABLE II: The experimental datasets are extracted from 6 most popular operator networks of China.

Datasets	Malicious URLs	Benign URLs	Total URLs	Malicious Proportion
CSTNET	157	23638	23795	0.66%
CERNET	1206	264149	265355	0.45%
CNNext2	668	371424	372092	0.18%
MOBILE	5180	1349729	1354909	0.38%
UNICOM	17770	6272434	6290204	0.28%
TELECOM	18177	5680386	5698563	0.32%

$$R = \frac{\#\ of\ correctly\ classified\ malicious\ URLs}{total\ number\ of\ truly\ malicious\ URLs} \quad (2)$$

To facilitate the simultaneous comparison of these two metrics, we introduced a new metric *Density* defined as the ratio of recall to filter rate.

$$Density = \frac{recall\ rate\ R}{filtering\ rate\ F} \quad (3)$$

The *Density* value means a multiple of the proportion of malicious URLs being increased after filtering. A higher *Density* indicates a better filtering effectiveness.

C. Experimental Settings

This malicious URL filtering system is implemented on the Spark cluster of 26 nodes where each node is an Intel(R) Xeon(R) CPU E5-2640 v3@ 2.60GHz machine with 16 cores and 125GB memory. And all experiments of time performance were conducted on the Spark cluster.

D. Results and Analysis

The first experiment was to evaluate the filtering effectiveness of different classifiers, including AdaBoost classifier (ADB), Decision Tree (DT), Gradient-Boosted Trees (GBDT), Perceptrons (PC), and so on. In order to accurately evaluate the filtering effectiveness of different classifiers, we conduct experiments on the dataset CNNext2 with the lowest proportion (0.18%) of malicious URLs and the most difficult task to classify. The *Density* values of different classifiers are shown in Fig.5. Among the these classifiers, The performance of ADB is much better than that of other algorithms, and the *Density* value exceed 250. Therefore, we choose ADB algorithm that has the best filtering effectiveness as our filter.

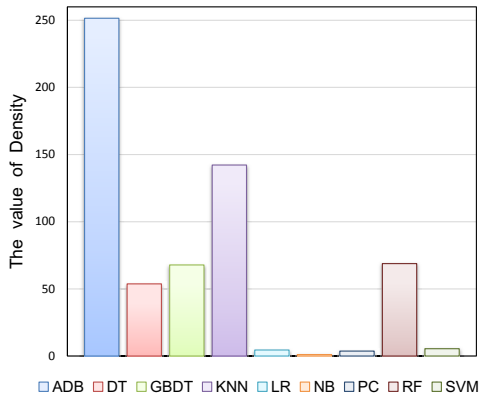

Fig. 5: Comparison of *Density* between different classifiers on the CNNext2 dataset.

The second experiment is to evaluate the effectiveness of our malicious URL filtering system on different datasets. Fig. 6 shows how recall of the proposed method changes with filtering rate by varying the filtering rate from 0 to 0.6. We can see that our malicious URL filtering system can reduce at least more than half of the load while achieving a high recall rate.

In Table III, we further list the recall rate when the filtering rate reaches 50%, and the filtering rate when the recall rate reaches 90%. From this we can know that when the filtering rate is approximately 50%, the average recall rate can reach 94.53% and the best can reach 98.37% on the CERNET. When the recall rate is approximately 90%, the filtering rate averages 28.99%, and the best filtering rate can reach 13.64% on the CSTNET.

Next, to demonstrate the importance of different features, we evaluate the impact of these features on the classification effect based on univariate statistical tests. Then we score and

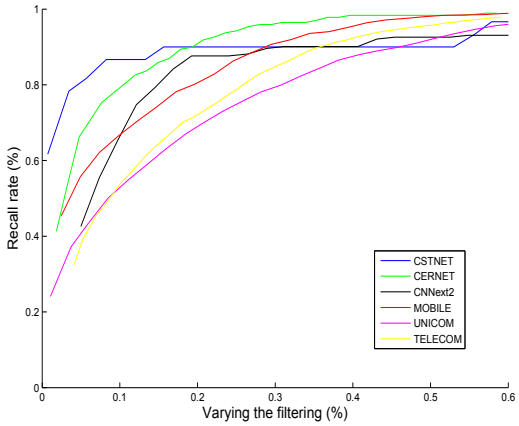

Fig. 6: Recall and filtering rate on different ISP networks.

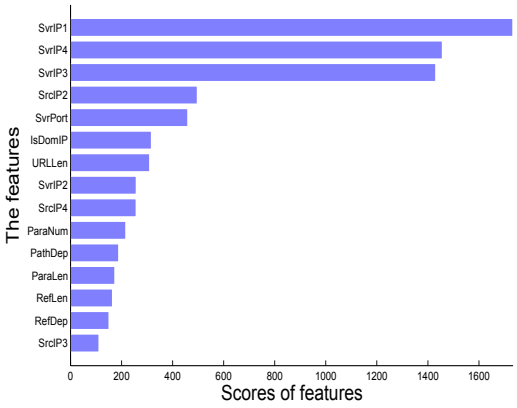

Fig. 7: The importance of different features.

TABLE III: Details of filtering and recall rates.

Dataset	Filtering	Recall
MOBILE	51.67%	98.34%
	29.53%	90.71%
TELECOM	50.31%	95.76%
	35.68%	90.07%
CERNET	50.04%	**98.37%**
	19.57%	90.49%
UNICOM	50.34%	92.15%
	46.10%	90.01%
CNNext2	50.01%	92.57%
	29.40%	90.10%
CSTNET	50.02%	90.00%
	13.64%	90.00%
Average	50.40%	**94.53%**
	28.99%	90.23%

rank all features. We list the top 15 features of the highest score in Fig. 7. It is worth noting that among the top 15 features with the highest scores, the features extracted from the user IP account for three, namely the fourth SrcIP2, the ninth SrcIP4, and the fifteenth SrcIP3. This also validated that it is effective to filter malicious URLs from the perspective of the user IP accessing the URLs.

Finally we conducted a set of experiments to evaluate the time performance of our malicious URL filtering system. The experimental results are presented in Table IV. It can be seen that the time performance of our method outperforms other work in both feature collection and URL classification. This is because our method is based on more lightweight features and combines bidirectional server IP and user IP characteristics. It is worth mentioning that the time performance is the average of each machine, although our experiment was done on the Spark cluster.

TABLE IV: Comparison of the time performance between our MalFilter and previous work.

Work	Features collection	Classification	Features
[18]	0.150s/page	0.034s/page	Web content
[19]	3.560s/URL	0.020s/URL	URL lexical, Host-based
[12]	3.060s/page	0.237s/page	Web content, URL lexical, Host-based
MalFilter	0.057s/URL	0.018s/URL	Server-based, User-based, URL lexical, Referer-based

E. Discussion

There is an important issue we need to consider is that our features are based on the technology of deep packet detection. So can our approach work when it comes to encrypted traffic, such as HTTPS? Encouragingly, one of the great advantages of our approach is that the features we need are very easy to obtain. For example, the server-based features and user-based features can be obtained from TCP layer traffic, without being affected by the HTTPS protocol at all. On the other hand, most servers in the long tail do not support HTTPS [20], especially malicious URLs. Moreover, as an alternative, we can also get most of the features from passive DNS traffic. To sum up, our approach is still promising in filtering malicious URLs.

VI. Conclusion and Future Work

In this paper, we propose a lightweight real-time malicious URLs filtering method for reducing the load of deep content analysis system. By introducing novel and discriminating features, our system can effectively reduce the load to an

average of 28.99%, the best 13.64%, while achieving a recall rate of approximately 90%. And when the load is reduced to approximately 50%, our method can achieve an average recall rate of 94.53%, the best 98.37%. Benefited from lightweight features, our system is able to process each URL within 0.075 seconds including feature collection and classification, which is pretty suitable for online real-time malicious URLs filtering. In the future, we intend to expand our work to explore the specific effects of these features. And we will further improve the filtering rate while achieving a high recall rate.

ACKNOWLEDGMENT

The research work is supported by National Key R&D Program 2016 (Grant No.2016YFB0801300, No.2016YFB0801205), National Natural Science Foundation of China (No.61602474, No.61602467, No.61702552). P.Zhang is the corresponding author.

REFERENCES

[1] Cisco, "The zettabyte era: Trends and analysis," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[2] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know your phish: Novel techniques for detecting phishing sites and their targets," in *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*. IEEE, 2016, pp. 323–333.

[3] S. Arshad, A. Kharraz, and W. Robertson, "Include me out: In-browser detection of malicious third-party content inclusions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 441–459.

[4] S. Marchal, J. François, R. State, and T. Engel, "Phishstorm: Detecting phishing with streaming analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.

[5] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, p. 14, 2014.

[6] X. Ma, J. Zhang, J. Tao, J. Li, J. Tian, and X. Guan, "Dnsradar: Outsourcing malicious domain detection based on distributed cache-footprints," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1906–1921, 2014.

[7] P. K. Manadhata, S. Yadav, P. Rao, and W. Horne, "Detecting malicious domains via graph inference," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 1–18.

[8] J. Zhang, P. A. Porras, and J. Ullrich, "Highly predictive blacklisting." in *USENIX Security Symposium*, 2008, pp. 107–122.

[9] B. Wang, N. Z. Gong, and H. Fu, "Gang: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 465–474.

[10] I. Alabdulmohsin, Y. Han, Y. Shen, and X. Zhang, "Content-agnostic malware detection in heterogeneous malicious distribution graph," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 2395–2400.

[11] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[12] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 197–206.

[13] M.-S. Lin, C.-Y. Chiu, Y.-J. Lee, and H.-K. Pao, "Malicious url filteringa big data application," in *big data, 2013 IEEE international conference on*. IEEE, 2013, pp. 589–596.

[14] 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.

[15] S. Wang, C. Liu, X. Gao, H. Qu, and W. Xu, "Session-based fraud detection in online e-commerce transactions using recurrent neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017, pp. 241–252.

[16] S. Kim, J. Kim, and B. H. Kang, "Malicious url protection based on attackers' habitual behavioral analysis," *Computers & Security*, 2018.

[17] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive dns traces," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 311–320.

[18] C. Seifert, I. Welch, and P. Komisarczuk, "Identification of malicious web pages with static heuristics," in *Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian*. IEEE, 2008, pp. 91–96.

[19] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1245–1254.

[20] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring HTTPS adoption on the web," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1323–1338. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt