Malicious URL Filtering – A Big Data Application

Min-Sheng Lin, Chien-Yi Chiu, Yuh-Jye Lee and Hsing-Kuo Pao
Dept. of Computer Science and Information Engineering
National Taiwan Univ. of Science and Technology
Taipei, 10607 Taiwan
M9915023, D9815013, yuh-jye and pao@mail.ntust.edu.tw

***Abstract*—Malicious URLs have become a channel for Internet criminal activities such as drive-by-download, spamming and phishing. Applications for the detection of malicious URLs are accurate but slow (because they need to download the content or query some Internet host information). In this paper we present a novel lightweight filter based only on the URL string itself to use before existing processing methods. We run experiments on a large dataset and demonstrate a 75% reduction in workload size while retaining at least 90% of malicious URLs.**

Existing methods do not scale well with the hundreds of millions of URLs encountered every day as the problem is a heavily-imbalanced, large-scale binary classification problem. Our proposed method is able to handle nearly two million URLs in less than five minutes. We generate two filtering models by using lexical features and descriptive features, and then combine the filtering results. The on-line learning algorithms are applied here not only for dealing with large-scale data sets but also for fitting the very short lifetime characteristics of malicious URLs. Our filter can significantly reduce the volume of URL queries on which further analysis needs to be performed, saving both computing time and bandwidth used for content retrieval.

***Index Terms*—Data Mining, Machine learning, Information Security, Information Filtering**

I. Introduction

Malicious URLs have become a common channel to facilitate Internet criminal activities such as drive-by-download, spamming and phishing. Many attackers try to use these web sites for spreading malicious programs or stealing identities. Kaspersky Lab [14] reports that browser-based attacks in 2012 increased from 946,393,693 to 1,595,587,670 and 87.36% of these used malicious URLs. The Anti-Phishing Working Group (APWG) [4] also reports that phishing attacks using malicious URLs increased from 93,462 to 123,486 in the second half of 2012. Of the millions of URLs used each day less than 0.01% are malicious and furthermore they are short-lived in order to avoid blacklist blocking. There is an urgent need to develop a mechanism to detect malicious URLs from the high volume, high velocity and high variety data. Many information security companies and organizations offer malicious URL detection including Google's safe browsing [9] and Trend Micro's web reputation services [16].

The blacklist is a general solution for protecting users from the malicious web sites, examples include PhishTank [18], SORBS [2] and URIBL [3]. These services provide a list of malicious web sites reported by volunteers or collected by web crawlers and verified by some reliable back-end system. The content-based analysis service BLADE [1] is also a well known solution for detecting malicious web sites. They download the web page content and analyze it for malicious content. To download the content for analyzing is time consuming and consumes bandwidth. Therefore content-based analysis services will be integrated with a blacklist or a cache mechanism to optimize the performance and avoid re-analyzing the same URL. However, content-based analysis methods are not a practical solution for the large volume of URLs and the speed at which new URLs can be created. We propose a filtering mechanism to be used before the content-based analysis to remove the bulk of benign URLs, reducing the volume of URLs on which content-based analysis needs to be performed. This job is just like finding a needle in a burning haystack because of the following challenges:

- Large scale: the service provider averagely receives several million URLs every hour
- Extremely imbalanced data set: the malicious URLs are only around 0.01% of the total received URLs
- Sparse data set: the lexical features give us a very sparse data set

To meet the goal of reducing the great amount of URL queries, the filtration system should be able to work in real-time and achieve a high detection rate with a tolerable false alarm rate. The system should also be able to update the model using the feedback of the content-based analysis system. In this paper, we introduce a framework to filter the non-suspicious URLs using only features extracted from the URL string. We set the filtration as a binary classification problem. The proposed framework combines lexical information and static characteristics of URL strings together for classification. We evaluate our framework using a real-world data set, which is a large-scale and extremely imbalanced data set. Our framework can filter out 75% of URLs, which are regarded as benign. The remaining 25% suspicious URLs cover around 90% of the malicious ones. Our framework does not use a network query or web page contents. All the features are directly extracted from the URL string, so our URL filtering system can work efficiently. More specifically, our system updates the model with one million of URLs in less than five minutes. The prediction process of our system is also efficient; one million URLs can be classified in less than two minutes.

II. Related Work

Garera et al. [8] designed 18 hand-selected features to classify phishing web sites, such as page rank information and

978-1-4799-1293-3/13/$31.00 ©2013 IEEE

the period of accessible time. The 18 hand-selected features are very similar to our static characteristic features, but these features still need to query information from the network. Our work extracts the static characteristic features only from the URL string and gives a well representation to distinguish benign and suspicious URLs.

The work by Ma et al. [13] is the most popular research on malicious web site in recent years. They utilize the blacklist, host-based information and lexical information of URLs to build different feature sets. In order to handle large-scale data sets, they use an online learning algorithm and compare the performance with a batch learning algorithm. The results of tests on a data set spanning 100 days are promising. We study their lexical features and adapt them to our own. Le et al. [11] present a modified version of the method proposed by Ma et al. by using another online learning algorithm. They merge the study from Garera et al. to improve the detection performance. Thomas et al. [20] design a framework to extract lexical, host-based and page content information as features and implement their idea on a cloud computing platform. This framework is tested on a data set of spam messages gathered from Twitter and e-mail. The result shows the framework achieves high performance at little cost. Pao et al. [19] proposed using a Kolmogorov complexity-based measure to detect malicious URLs. They adopt a compression method to approximate Kolmogorov complexity, and used the approximation as a significant feature for detection. Unlike their goal of detecting malicious URLs, our work focuses on filtering the large amount of URLs to pick up the most suspicious ones. We use the decision value of the online model to quantify if a URL is sufficiently suspicious to justify downloading the linked content for analysis.

Blum et al. [5] have proposed a similar idea. They try to avoid network queries to reduce processing time, and also choose a confidence-weighted algorithm for lexical information features to detect phishing sites. Different from them, we focus on not only the phishing pages but also all the attacks over the URLs. We also evaluate our framework on a large-scale and extremely imbalanced data set. The problem in the work by Whittaker et al. [21] is similar to ours in that they too use an imbalanced and large-scale data set. They use the information of host-based, networking, lexical and page content to classify web sites and automatically generate their blacklist. The framework which we proposed does not need to query the information from the network so our system can handle the user query in real time. Several projects have also explored different ways to protect users from malicious URLs. Li et al. [12] proposed MadTracer which can automatically generate detection rules to detect malicious advertising activities. Invernizzi and Comparetti [10] presented EVILSEED to search malicious web pages more efficiently from an initial seed of known, malicious web page.

III. Feature Extraction

In order to successfully separate benign URLs from malicious ones, we design two feature sets to describe URLs.

TABLE I
Components of URL

Component	Example
URL	aneisig.es/vx/hstart.php?id=664&logon=141
Domain Name	aneisig.es
Path	vx/hstart.php
Sub-Directory	vx
Filename	hstart
File Extension	php
Argument	id=664&logon=141

TABLE II
The Delimiters of URL Component

Component	Delimiters
Domain Name	dash and slash
Path	dash, dot, underscore and back slash
Argument	ampersand and equal sign

These two feature sets represent the lexical information and some other characteristics of URLs. The first feature set which includes *lexical features* describes the lexical information of URLs. The second feature set which includes *descriptive features* describes some statistical characteristics of URLs. These two feature sets represent two different natures of the URLs. At the first sight, we can imagine that the lexical features are effective for malicious URL detection by a blacklist rule. However, most of malicious web sites have a short lifetime and change their addresses frequently to avoid being blocked by the blacklist. Therefore, even though the blacklist could be effective for a short period of time we need to study some other features that can also be effective in a long run. In our opinion, the lexical features describe the dynamic nature of URLs by the "words" used in the URLs; while the descriptive features describe the static nature of URLs since the characteristics between benign and malicious URLs are rarely changed drastically. For example, phishing sites often use similar letter or symbols (such as the lower case of 'L' and the digit '1') to confuse victims, therefore the sites may have some particular statistics about the number or the consecutive relationship of alphabets and digits. Some other malicious sites tend to have randomly chosen characters or characters that are less recognized as words of known meanings. In summary, we believe that the lexical features are more effective than descriptive features, but they can only work in a short period of time; on the other hand, the descriptive features are less-effective than the lexical features, but they may work for a longer time. In this paper, we combine these two complementary feature sets which serve as two different views to filter malicious URLs. Before introducing the feature sets, we give a definition of URL components. We split the URL into several components as shown in table I. The lexical features and descriptive features are extracted from these components. The details are described in following subsections.

A. Lexical Features

The lexical features use the bag-of-words model to describe URLs. We use three components for extracting lexical features: domain name, path and argument. Each component is split by specific delimiters. The details of delimiters are shown in table II. Additionally, we use a three character length sliding window on the domain name for generating the fixed length tokens. This method can identify a malicious website which subtly modifies its domain name.

Each word/feature generated from URLs is stored in a dictionary with a specific index, only the same word can get the same index. This dictionary could be dynamically updated with data streaming, but in practice the dictionary would consume a large amount of memory for the words generated from millions of URLs. For this reason, we use following methods to remove less useful words and to reduce memory usage.

1) Remove zero-weight words: Removing zero-weight words in the learned model is an easy and intuitive method. The learned model of lexical features are a set of weights of the extracted words (see section IV). If the weight of the word is zero, it means this word doesn't provide any information on giving a decision. In our setting, these zero-weight words will be removed from the dictionary after an updating procedure is finished. If the removed word appears again in next updating procedure, it will be treated as another new word. It means the reappeared word will get a new index which is different from removed one.

2) Remove argument value words: The argument value in a URL is usually a serial number, random string or user input. Except for the user input which may contain some information, the words generated from the serial number and random strings are meaningless. These words will consume much memory but provide no help for classifying URLs. For this reason, we decide to remove all words from argument value. For example, in table I, the argument part is "id=664&logon=141". We remove the argument value "664" and "141" and are leave the argument names with "id" and "logon".

3) Replace IP address with an AS number: Many URLs use an IP address for their domain name. To reduce the dictionary size and effectively manage these IP addresses, we use the mapping table from MaxMind [15] for mapping the IP address to an Autonomous System (AS) number. Each AS numbers uniquely identifies a network on the Internet, therefore, the IP will be changed to an AS number which represents the network which contains the IP address.

4) Replace the digits in words with regular expression: Many strings differ only in numerical values, so we replace all numerical values by a regular expression. For example, if two words are abc123 and abc789, they will be rewritten by the same regular expression "abc[0-9]+". So if some words contain different digit characters, they will be seen as the same word.

5) Keep the words generated in the last 24 hours only: In addition to the index, we also give every word a time stamp. In each training process, if a word exists in the dictionary and occurs again, the time stamp will be updated. If any word has not been updated for 24 hours, the words will be removed because we think the words are expired.

B. Descriptive Features

In descriptive features, we split the path component into sub-directory, filename and file extension to obtain more detailed information. To accurately extract the information on domain name, we remove the "www" (and www with any number), country code top-level domain (ccTLD) and generic top-level domain (gTLD). Removing these tokens can help us to focus on the remaining part chosen by the domain name owner.

The descriptive features represent the static characteristics of the URL. These static characteristics are different from lexical features, which can be easily changed by slightly modifying the characters of URLs. We studied the URLs of phishing and malware sites to design these descriptive features. These features can help us to distinguish malicious and benign URLs. Table III shows the descriptive features, extracted components and dimension.

1) Length: From our observation, the malicious URLs commonly need to add some keywords in its components. This action causes URLs to become noticeably lengthier. For detecting this situation, we record the length of each component. All these length values will be stored by the logarithm base 10 to avoid the normalization process being unduly affected by the large length value.

2) Length ratio: In addition to length, we also want to see the length ratio of different components. It can help to find the abnormal component. The length ratio feature uses the length of URL, domain name, path and argument component. We check all of the combinations of these components and compute the length division as follows:

- Domain Name divided by URL
- Path divided by URL
- Argument divided by URL
- Path divided by Domain Name
- Argument divided by Domain Name
- Argument divided by Path

Each pair of combination on the list contributes a descriptive feature in our feature set.

3) Letter-digit-letter and Digit-letter-digit: For detecting a phishing website masquerading as another website, we record the number of occurrences of the following two patterns: a letter between two digits (ex. award2o12) and a digit between two letters (ex. examp1e). They can help detect whether the URL is trying to deceive the user or not.

4) Delimiter count and Longest word length: We previously used delimiters to split the components in lexical features, now we create a descriptive feature by counting the number of delimiters in each component. The length of the longest word after splitting is also recorded.

5) Letter, Digit and Symbol count: Here, we categorize all characters as letter, digit and symbol and tally the frequency

TABLE III
THE USED COMPONENT AND DIMENSION OF DESCRIPTIVE FEATURE

	URL	Domain Name	Path	Sub-directory	Filename	File Extension	Argument	Dim.
Length	√	√		√	√	√	√	6
Length ratio	√	√	√				√	6
Letter-digit-letter	√	√		√	√	√	√	6
Digit-letter-digit	√	√		√	√	√	√	6
Delimiter count		√	√				√	8
Longest word length		√	√				√	3
Letter count	√	√		√	√	√	√	6
Digit count	√	√		√	√	√	√	6
Symbol count	√	√		√	√	√	√	6
Alphabet entropy	√	√		√	√	√	√	6
Number rate	√	√		√	√	√	√	6
Executable file or not						√		1
IP as domain name		√						1
Default port number	√							1
Character continuity rate		√						1
Total								69

of each of the three types in all URL components except path. Each type provides a 6-dimensional vector as a further feature.

6) Alphabet entropy and Number rate: Some URLs of malicious websites are randomly generated. For recognizing this static characteristic, we use the alphabet entropy and the rate of number to help us distinguish the randomly generated URLs. Set $P(x_i)$ is the probability of letter x_i occurs in the URL component, the alphabet entropy H can be computed by

$$H = -\sum_{i=0}^{25} P(x_i) \log P(x_i)$$
$$\text{where } (x_0 = a, x_1 = b, ..., x_{25} = z)$$

The number rate records the proportion of the digits in each URL component. This is computed by tallying the number of occurrences of digits and dividing by the string length of URL component. For example, the number rate of "Example123" $= 3/10 = 0.3$.

7) Executable file or not: The executable file has a high probability of being a malware [17]. We use a binary feature to represent whether the file extension is "*.exe" or not. Notice that we only look at the file extension of the URL rather than download the real file.

8) Using IP as domain name and Default port number: Most benign URLs do not use IP address as its domain name, and the port number of website is usually 80. Based on these ideas, we check if the domain name is an IP address and if the port number is the default HTTP port or not.

9) Character continuity rate: Generally, the owners of legitimate websites prefer to use a simple and memorable domain name, even if they need to pay more money. But malicious website owners frequently change the domain name so they are unlikely to pay more money for buying a better domain name. Based on this idea, we design the character continuity rate of domain name.

First we categorize the character to letter, digit and symbol. The domain name will be split by the connection of different character category, and the sum of the longest token length of each character type will divide by the domain length. For example, the continue rate of "abc567-gt" $= (3 + 3 + 1)/9 = 0.77$.

IV. ONLINE LEARNING ALGORITHM

To meet our goal on dealing with large-scale data efficiently, we choose to use online learning algorithms for building our filters. An online learning algorithm works in rounds. For each round i, the algorithm receives an instance $x_i \in \mathbb{R}^d$ to make the prediction with its model w_i. And then it receives the true label $y_i \in \{-1, +1\}$ and suffers prediction loss $\ell(w_i; (x_i, y_i))$.

In our system, we choose the Passive-Aggressive (PA) algorithm and the Confidence Weighted (CW) algorithm as our learners to train our models. The PA algorithm tries to modify the model as little as possible to correctly classify the new incoming instance. This rule matches our idea of maintaining a stable model for describing the static characteristics. The CW algorithm updates the weights of the model by their confidence. The weights of rarely updated features will be updated more aggressively. For the lexical features, this algorithm can maintain the dynamic lexical information appropriately.

A. Passive-Aggressive Algorithm

The PA algorithm [6] uses the hinge loss function. For each instance x_i, PA will solve following optimization problem:

$$w_{i+1} = \arg\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w - w_i\|^2$$
$$\text{s.t. } \ell(w; (x_i, y_i)) = 0$$

For each round i, the model will be updated if the new instance x_i is misclassified. The updated model should correctly predict x_i and the difference between the old model and updated model should be minimized. If y_i is mislabeled, the PA algorithm will be updated substantially in the wrong direction. To deal with this phenomenon, it can add non-negative slack variable ξ to this optimization problem:

$$w_{i+1} = \arg\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w - w_i\|^2 + C\xi$$
$$\text{s.t. } \ell(w; (x_i, y_i)) \leq \xi \text{ and } \xi \geq 0$$

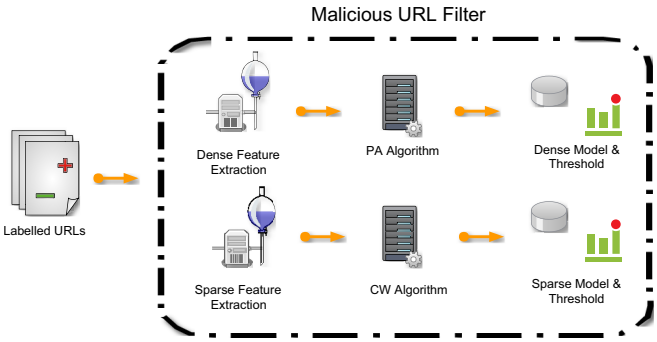


Fig. 1. Training/Updating Process

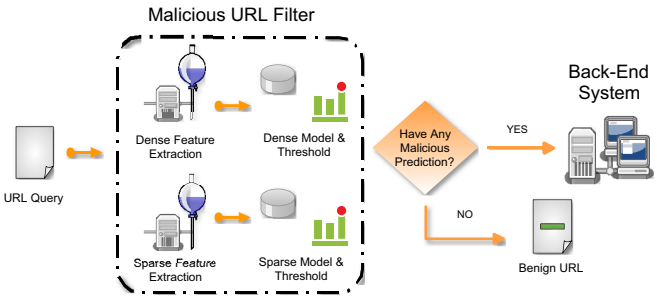


Fig. 2. Prediction Process

The above formulation is called PA-I. The positive parameter C controls the effect of the slack term on the objective function, a smaller C will give a larger soft margin. The closed-form solutions of PA-I are as follows:

$$w_{i+1} = w_i + \alpha_i y_i x_i$$
$$\alpha_i = \min\left\{C, \frac{\ell(w;(x_i,y_i))}{||x_i||^2}\right\}$$

For the above closed-form solution, the updating step size is limited by C, so even if noise exists the model will not be greatly altered in the wrong direction.

B. Confidence Weighted Algorithm

The CW algorithm [7] uses the confidence to decide the updating step size of the weights. The model $w \sim \mathcal{N}(\mu, \Sigma)$ which $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d\times d}$, and the divergence between old model and updated model will be minimized on the condition of correctly predicting x_i. The optimization problem is as follows:

$$(\mu_{i+1}, \Sigma_{i+1}) = \min D_{\mathrm{KL}}(\mathcal{N}(\mu,\Sigma)\|\mathcal{N}(\mu_i,\Sigma_i))$$
$$s.t. \quad Pr[y_i(w\cdot x_i)\geq 0] \geq \eta$$

Here D_{KL} is KL Divergence and η is the probability with which the model correctly predicts the training instance, so $\eta \in [0,1]$. And The closed-form is as follows:

$$\mu_{i+1} = \mu_i + \alpha_i y_i \Sigma_i x_i$$
$$\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha_i\phi \mathrm{diag}^2(x_i)$$

where the α_i is the learning rate which can be computed by the following equation:

$$\alpha_i = \max(0, \gamma_i)$$
$$\gamma_i = \frac{-(1+2\phi M_i) + \sqrt{(1+2\phi M_i)^2 - 8\phi(M_i - \phi V_i)}}{4\phi V_i}$$

where Φ is the cumulative distribution function of the Gaussian distribution, and $\phi = \Phi^{-1}(\eta)$. The $M_i = y_i(x_i, \mu_i)$ is the mean margin and the $V_i = x_i \Sigma x_i$ is margin variance before updating.

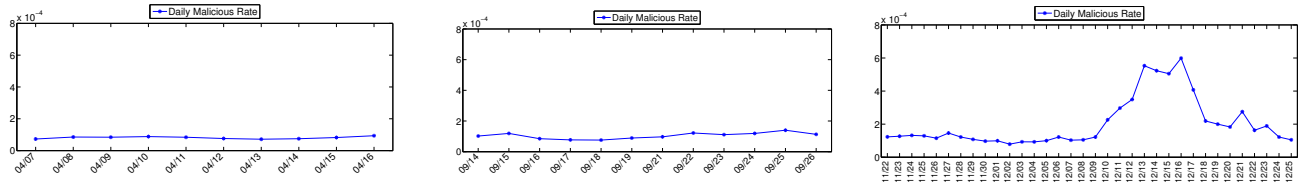
V. Our Proposed Framework

Our goal is to construct a front-end system, which can automatically filter out 75% of URLs queried and pass 75% or more malicious URLs to the back-end system. In other words, we want to reduce the loading of the back-end system down to 25% while ensuring that at most 25% of genuinely malicious URLs are dropped.

We use the CW algorithm as the learner of lexical features, because the nature of lexical information is similar to Natural Language Processing (NLP). For descriptive features, we select the PA-I algorithm, because PA-I can make sure that the model is modified within a limited range in each round. The PA-I is suitable for descriptive features, because the descriptive features represent the static characteristics of the URL and so its weights should not be changed greatly.

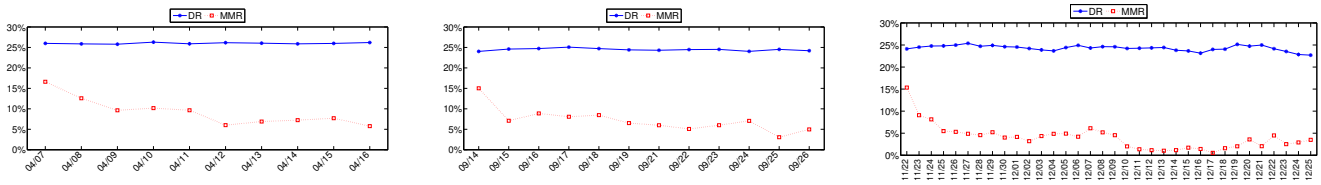
In the training/updating process (see Figure 1), our system receives the URL instances and its label, each instance will be processed to compute both a lexical feature vector and a descriptive feature vector. The descriptive feature vectors are normalized by min-max normalization using min/max values obtained from the first training set. After feature extraction, both descriptive feature and lexical feature vectors are sent to corresponding learning modules for updating the model. For dealing with the extremely imbalanced data, we use the over-sampling technique to make sure the characteristics of malicious URLs are learned by both filters. During the training/updating process, each benign instance will follow a randomly picked malicious instance to feed into the learners. That will let both learners receive balanced data to train/update the model.

For achieving our goal, we use the updated models to compute the suspicious score of each instance. The suspicious score is the decision value computed by a given model and a predicting instance. For binary classification, the classifier predicts an instance as normal if the score is less than zero. In our framework, we replace the zero by a threshold which is derived from training data. Here we define a configurable parameter τ to obtain the threshold. After training, we will evaluate the suspicious score of all training instances and use τ percentile to decide our threshold. Namely, that τ percent of the scores should be within this selected score. We set the τ as 85% in our framework for deriving the threshold of lexical and descriptive filter. Notice that, the thresholds of descriptive



(a) April (b) Sept. (c) Nov.-Dec.

Fig. 3. Daily Malicious Rate



(a) April (b) Sept. (c) Nov.-Dec.

Fig. 4. Prediction Performance

filter and lexical filter are independently obtained from this process and τ could be changed for different situation.

In the prediction process (see Figure 2), each URL is extracted to descriptive feature vector and lexical feature vector (the descriptive feature vector is normalized.) After feature extraction, the vectors are sent to the corresponding model to estimate their suspicious score. If any suspicious score is greater than the threshold, the URL will be predicted as a suspicious URL and it should be passed to the back-end system. In our framework both the lexical filter and descriptive filter can work independently. To combine their prediction results, if one of them predicts an URL as suspicious, the URL will be considered as suspicious and will be analyzed by back-end system.

When the back-end system receives the filtered suspicious URLs, the further results will be fed back to our system by a more accurate but time consuming technique (such as content-based analysis). These results will be used as the ground truth to update our model. In our idea, the filtered benign URLs should also be examined by another light-weight technique (such as rule-based detection or analysis with host-based features) to double checked results. These checks can also be used to update our model.

VI. Experiments

Our data set is very large and extremely imbalanced, one million URLs only contain approximately one hundred malicious instances. For this reason, the accuracy is not suitable for measuring our performance. Instead of accuracy, we define the download rate and the missing malicious rate to evaluate our prediction performance. We evaluate our framework using three data sets (include a monthly data set), and record the average processing time. For understanding the contribution of the descriptive and lexical filter, we also independently evaluate each filter on these data sets. Finally, we run an experiment to evaluate the performance of two filters without updating. The results help us to explore the differences and properties of two filters. In our evaluation, the parameter η of CW (lexical learner) is set to 0.85 and C of PA-I(descriptive learner) is set to 0.001.

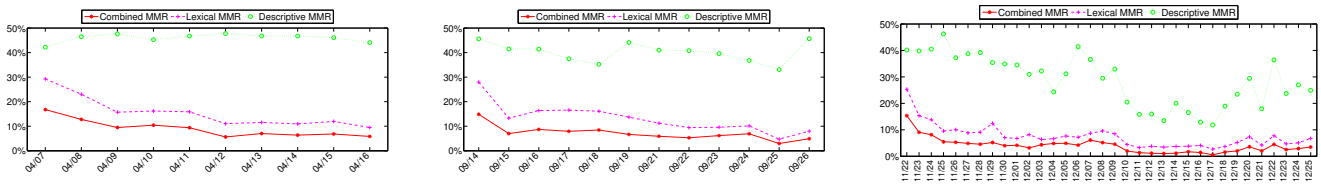
TABLE IV
The Information of Data Set

	Server 1	Server 2	
Data Set Name	Apr.	Sept.	Nov.-Dec.
Date Range	04/07-04/16	09/14-09/26	11/22-12/25
# of URL	261,426,377	284,429,411	848,867,427
# of Attacks	20,280	26,610	125,783
Malicious Rate	0.0077%	0.0093%	0.0148%

A. Data Set

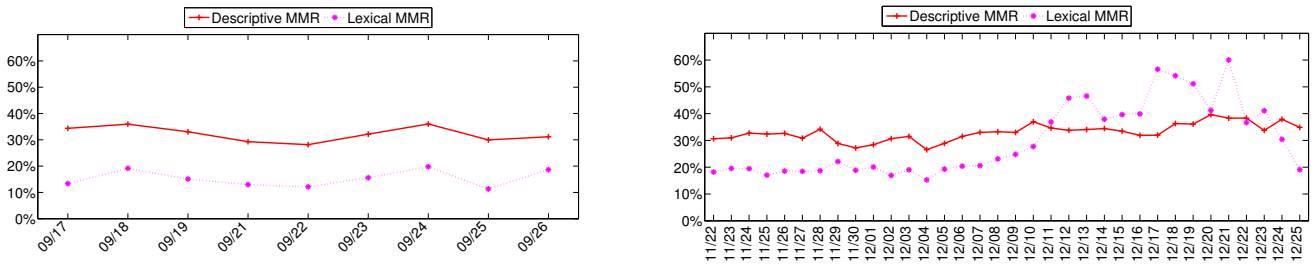
The data set is provided by a security company which provides web reputation services. They collected three data sets from two different servers, which are named Server 1 and Server 2. All the URLs in three data sets are unique and labeled as eigher benign or malicious. The first data set is collected from Server 1 and the period is from April 7, 2011 to April 16, 2011. This data set is called April (Apr.) data set. The second data set collected from Sever 2, and the period is from September 14, 2011 to September 26, 2011 without September 20. It is named after September (Sept.) data set. The third data set is the longest data set from November 22, 2011 to December 25, 2011. It is collected from Server 2 and called November and December (Nov.-Dec.) data set. The breakdowns of the data sets are shown in Table IV.

In our evaluation, the data set is segmented by hours. Each training and prediction process is dealt with one hour data and the performances are averaged by days. The daily malicious rate of each data set is shown in Figure 3. The data sets have on average one million instances per hour, but only contain a few malicious instances. In a general classification task, this situation will let the model classify all URLs as benign for



(a) April (b) Sept. (c) Nov.-Dec.

Fig. 5. The MMR result of Descriptive Filter, Lexical Filter and Combined Results



(a) Sept. (b) Nov.-Dec.

Fig. 6. The performance of static model

correctly predicting most of the instances. Additionally, the malicious rates are rapidly increasing in the Nov.-Dec. data set, the impact of this phenomenon will be shown in the evaluation subsection.

B. Performance Evaluation

1) Measure: We use two measures as our criteria to judge the performance of the malicious URL filter, the download rate (DR) and missing malicious rate (MMR). The download rate is the percentage of URLs classified as suspicious instances by our system. It is also the rate at which content must be downloaded for analysis by the back-end system. The missing malicious rate is the rate of misclassification of malicious instances as benign URLs. The two measures can help us to judge the prediction performance in the extremely imbalanced data set.

2) Evaluation: We will evaluate our implemented system in this subsection and focus on following key points

- Performance of our framework in large and extremely imbalanced data set
- Efficiency of our implemented system
- The differences between descriptive filter and lexical filter.

a) Prediction performance: We use URLs of each hour to train/update our implemented filter, then apply the filter to classify the next hour of data. The daily average results are shown in Figure 4. The download rates are steady at around 25% and missing malicious rates at 9% or lower after the first day. This result shows that our framework can filter out 75% URLs, and only a few malicious URLs will be lost. It is worth noting that our system can achieve better performance with a stable detection rate while the daily malicious rate rapidly increased in Nov.-Dec. data set.

TABLE V
THE TIME OF TRAINING PROCESS AND PREDICTION PROCESS

	Apr.	Sept.	Nov.-Dec.
Training Time (min.)	2.99	3.06	3.34
Prediction Time (min.)	1.08	1.15	1.28

b) Processing time: Table V shows the averaged time for dealing with the URL strings that generated in one hour on three datasets. In our approach, we do not need the host-based information and the contents of web sites. Thus, we can train our filtering models with more than one million URL strings in 3.5 minutes including feature extraction, lexical filtering model and descriptive filtering model. We can test more than on million URL strings in 1.5 minutes.

c) Performance of descriptive and lexical filters: The previous evaluations show our framework works well by combining the prediction results of descriptive filter and lexical filter. To evaluate the performance of each filter independently, the individual missing malicious rate is compared with combined results in Figure 5. As the figure shows, most of malicious instances are detected by lexical filter. Although the descriptive filter does not work well, it can be used as a complement to the lexical filter for improving performance.

For further understanding of the difference between the lexical filter and descriptive filter, we set up another experiment to evaluate the stability and expiration time of models. We use the data set from Sept. 14 to Sept. 19 as training data. The data set from Sept. 17 to Sept. 26 is used as short-term testing set and the Nov.-Dec. data set is used as long-term testing data. The model and threshold generated from the training data were not updated during testing. Figure 6 shows the MMR of lexical filter is lower than descriptive filter in the short-term

testing set. But in the Nov.-Dec. data set, the MMR of lexical filter becomes to increase and unstable. Even in middle of December, the lexical filter misses more malicious URLs than descriptive filter.

Based on these results, the lexical filter has unstable performance over time. The model using lexical features needs to continually update to retain its performance. In contrast with the lexical filter, the descriptive filter has long-term effectiveness but the detection ability is lower than the lexical filter.

VII. Conclusion

We focus on a very practical problem, efficiently and effectively detecting malicious URLs. Working with a world-famous information security company, the goal is to provide them a filtering system that is able to filter out 75% URLs and the remaining portion should contain 75% of all malicious URLs originally present. That is, both the download rate and missing malicious rate should be less than 25%. The most challenging part of this problem is how to find an extremely small portion of malicious URLs out of a huge volume of URLs being generated at high speed. The scale of the problem and limited processing time prohibit the solution provided by a conventional approach. We proposed a novel method that uses the URL strings only for the detection. In the proposed method, we combine the lexical information and static characteristics of URL string. Without the host-based information and content-based analysis, we are able to deal with two millions URL strings in five minutes. Our system misses less than 9% of malicious instances while only 25% of the data set are required detailed content-based analysis by the back-end system. This has surpassed the requirements from the security service provider. We also introduced an online learning technique in our framework so that the filtering models can be modified dynamically if there is any feedback from the back-end content analysis engine. Moreover, the online learning mechanism also fits with the very short lifetime characteristic of malicious URLs. Our proposed framework will significantly reduce the burden of doing content-based analysis and using the bandwidth for content retrieval and can be combined with other web security services smoothly.

Acknowledgment

We thank the anonymous reviewers for their valuable comments and feedback. This work was supported by National Science Council, National Taiwan University and Intel Corporation under the grants NSC101-2911-I-002-001 and NTU102R7501.

References

[1] Blade malware url analysis results. http://www.blade-defender.org/eval-lab/.

[2] Spam and open relay blocking system. http://www.sorbs.net/.

[3] Uribl.com realtime uri blacklist. http://www.uribl.com/.

[4] G. Aaron and R. Rasmussen. Global phishing survey: Trends and domain name use in 2h2012. http://docs.apwg.org/reports/APWG_GlobalPhishingSurvey_2H2012.pdf, 2013.

[5] A. Blum, B. Wardman, T. Solorio, and G. Warner. Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security*, AISec '10, pages 54–60, New York, NY, USA, 2010. ACM.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Dec. 2006.

[7] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 264–271, New York, NY, USA, 2008.

[8] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, WORM '07, pages 1–8, New York, NY, USA, 2007. ACM.

[9] Google. Google safe browsing. https://developers.google.com/safe-browsing/.

[10] L. Invernizzi and P. M. Comparetti. Evilseed: A guided approach to finding malicious web pages. In *IEEE Symposium on Security and Privacy*, pages 428–442. IEEE Computer Society, 2012.

[11] A. Le, A. Markopoulou, and M. Faloutsos. Phishdef: Url names say it all. In *INFOCOM*, pages 191–195. IEEE, 2011.

[12] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing your enemy: understanding and detecting malicious web advertising. In T. Yu, G. Danezis, and V. D. Gligor, editors, *ACM Conference on Computer and Communications Security*, pages 674–686. ACM, 2012.

[13] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 681–688, New York, NY, USA, 2009. ACM.

[14] D. Maslennikov and Y. Namestnikov. Kaspersky security bulletin. statistics 2012. http://www.securelist.com/en/analysis/204792255/Kaspersky_Security_Bulletin_2012_The_overall_statistics_for_2012, 2013.

[15] MaxMind. Geolite autonomous system number database. http://www.maxmind.com/app/asnum.

[16] T. Micro. Trend micro web reputation service. http://cloudsecurity-apac.trendmicro.com/solutions-and-services/spn-feature/web-reputation-service.aspx.

[17] S. Mustaca. Phishing, spam and malware statistics for february 2011. http://techblog.avira.com/2011/03/12/phishing-spam-and-malware-statistics-for-february-2011/en/, 2011.

[18] OpenDNS. Phishtank. http://www.phishtank.com/.

[19] H. K. Pao, Y. L. Chou, , and Y. J. Lee. Malicious url detection based on kolmogorov complexity estimation. In *The 2012 IEEE/WIC/ACM International Conference on Web Intelligence*, Dec. 2012.

[20] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and evaluation of a real-time url spam filtering service. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, pages 447–462, Washington, DC, USA, 2011. IEEE Computer Society.

[21] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In *NDSS*. The Internet Society, 2010.