Malicious URL Detection Based on Kolmogorov Complexity Estimation

Hsing-Kuo Pao, Yan-Lin Chou,Yuh-Jye Lee
Dept. of Computer Science & Information Engineering,
National Taiwan Univ. of Science & Technology
Taipei, Taiwan
Email: pao@mail.ntust.edu.tw, cyl99m@gmail.com, yuh-jye@mail.ntust.edu.tw

***Abstract*—Malicious URL detection has drawn a significant research attention in recent years. It is helpful if we can simply use the URL string to make precursory judgment about how dangerous a website is. By doing that, we can save efforts on the website content analysis and bandwidth for content retrieval. We propose a detection method that is based on an estimation of the conditional Kolmogorov complexity of URL strings. To overcome the incomputability of Kolmogorov complexity, we adopt a compression method for its approximation, called conditional Kolmogorov measure. As a single significant feature for detection, we can achieve a decent performance that can not be achieved by any other single feature that we know. Moreover, the proposed Kolmogorov measure can work together with other features for a successful detection. The experiment has been conducted using a private dataset from a commercial company which can collect more than one million unclassified URLs in a typical hour. On average, the proposed measure can process such hourly data in less than a few minutes.**

***Index Terms*—blacklist, compression, entropy, Kolmogorov complexity, malicious URL.**

I. Introduction

The Internet environment contains a large amount of malicious websites. Understanding content of the malicious sites can help us to classify the sites as one that is designed with bad intention and make appropriate defensive actions afterwards. However, the content analysis usually takes a lot of computation efforts not to mention the bandwidth load to retrieve the content before the analysis. In this work, we propose a website classification method that can automatically detect malicious sites from benign ones based on only the URL strings.

Normally the URLs are made by website designers. Each designer uses his/her own preference to select a domain name, build the file structure under the domain name, and give name to the file. The URL of the website may have a pattern that belongs to a particular designer, but that is not quite the case for malicious websites. Usually, malicious websites are generated by pre-defined rules or procedures and by that, tons of URLs are produced in a short period of time. Based on the different naming procedures for benign and malicious sites, we should be able to find the difference between the two groups. That is the goal of this work, to detect malicious websites by understanding the patterns of URLs.

TABLE I
Domain names and filenames with the top five frequencies. The notation "?" indicates a wildcard character. Some patterns are common in both groups.

Rank	Domain name		Filename	
	Malicious	Benign	Malicious	Benign
1	?.bjcandy.com	tracker.metrotorrents.info	bad?.exe	favicon.ico
2	keeplinkslife.com	122.208.189.10	index1.php	din.aspx
3	www.imdvd.net	web.fc2.com	favicon.ico	index.php
4	www.jsccia.org	sanguo.pk238.com	index.php	announce.php
5	first.vs	nht-2.extreme-dm.com	sapdf0.pdf	index.html

Tables I shows[1] some common keywords in malicious URLs and benign URLs.[2] A simple and one of the traditional detection methods is to investigate keywords existed in malicious group or benign group and make a judgment based on a so-called *blacklist/whitelist* method. It is not always effective. For instance, we do see that the malicious and benign groups share some common patterns frequently, such as we observe "index" with high frequencies in both of the malicious and benign groups (on the right-hand side of Table I).

In this work, we propose a Kolmogorov complexity-based method for malicious URL detection. Some welcome features of the proposed method include:

1) The proposed method can successfully separate malicious URLs from benign ones.
2) The proposed method needs no prior understanding about the URL structure, such as how to parse the URL into separated parts like domain name, path, etc.
3) The proposed method can work independently as a single measure; on the other hand, it can also work with other useful features to achieve an even better performance.
4) The proposed method has efficient computation, such as it can finish the computation for a million URLs in a few minutes.

Before we go on to introduce the proposed method, we discuss some previous works on malicious URL detection in Section II; after that, we give more details about the datasets that we use in this work in Section III. In Section IV, we

[1]Be careful of browsing some of the websites as they may give unexpected or unwanted result.
[2]The statistics is based on one of our datasets, further discussed in Section III.

detail the proposed method. The experiment result is shown in Section V and in Section VI, we conclude our presentation.

II. Previous Work

Malicious URL detection has received major attention in recent years. It is mainly due to the fact that the number websites grows rapidly and the Web browsing for information finding becomes popular in the last decade.

The malicious URL detection can roughly divided into two groups, depending on whether the content information is used in the detection. The detection based on content analysis is usually more reliable because more information can help to make a better decision. On the other hand, the detection without using the content information can save a significant bandwidth. It says that eight billions of URLs may need to be analyzed per day for a company that focuses on malicious URL detection [1].

One of the most traditional approaches for malicious URL detection is based on the blacklist method. The major drawback of the blacklist method is its weakness for generalization. Ma et al. [2], [3] started to design a URL classifier to identify malicious URLs by examining lexical features of the URLs and other features of the sites. In their work, a URL was partitioned into four parts which are "domain", "path", "filename" and "argument"; and then a classification method will select the lexical features from the groups of malicious and benign URLs. Some lexical features include the number of dot, or how the delimiters such as "/", "?", etc are used in the path. Some other non-lexical features were also included in their feature set, such as the WHOIS, TTL information. Following Ma et al., PhishDef proposed by Le et al. [4] discussed another set of features for URL classification. One of the differences between Ma's method and PhishDef is that PhishDef can deal with URL obfuscation that are frequently used by attackers; moreover, PhishDef emphasized its ability to deal with noise data. Using some heuristics on lexical features as well as an approximate matching algorithm, PhishNet [5] attempted to detect phishing sites. We want to emphasize that our proposed method needs no particular understanding of URL structure. That makes the method easy to implement and robust to the situation when the understanding of URL structure is not correct, such as the case of short URL (e.g., http://tinyurl.com/).

III. Data Description

In this work, we use two datasets for the evaluation of the proposed method. The first dataset is obtained from Trend Micro Incorporated[3], denoted by D_{TM} that collects daily URLs that are submitted by the company's clients around the world. We test our proposed detection method on such a real world data to demonstrate that the proposed method can indeed deal with real world problems with commercial level efficiency. On the other hand, to compare with other existing methods, we also collect some data from public domains. We have the malicious URLs that were collected from the publicly accessible PhishTank [6]; and the benign URLs that were collected from Yahoo [7]. This dataset is denoted by D_{P+Y}. The complete statistics about the datasets are shown in Table II.

[3]The dataset is belonged to Trend Micro.

A. Company Data Set

The Trend Micro company collects the URLs that are submitted by their clients around the world based on the clients' browsing experience. Once some clients do not have confidence on browsing some websites, they may choose to upload the URLs to the company's server. Typically, the company receives around one million such URLs per hour on average. It is much larger than any datasets that we can collect from public domains. Once the company receives the URLs, they test the URLs based on several automatic content analysis measures, sometimes also with a small amount of human efforts to make judgment about whether the URLs belong to malicious ones or benign ones. There is no reason to believe that such labeling procedure can produce 100%-correct labels; but it should give a much more accurate result than that can be obtained from non-content-based measures.

The dataset is a very unbalanced dataset which has the ratio of malicious URLs to benign URLs less than 1 to 10000 on average. In total, we selected URLs that were received in 10 days (from April 7 to April 16, 2011), and then we divided the datasets into different groups on a daily basis. Afterwards, we use the first-day data (April 7) to build the first model; then, we continually to use the previous model to test the data on the next day, from April 8 to April 16 for our evaluation.

B. Open Source Data Set

We also collect URLs from public domains that consist of malicious URLs from PhishTank [6] and Yahoo directory [7]. The website PhishTank is a site for Web security community where anyone can submit suspicious URLs to the site. After that, the website checks the URLs by at least two other members. If the website confirms that a submitted URL is malicious they will add the URL to the PhishTank database. The database is available in multiple formats and updated hourly. We collected our data during the period from June, 2010 to February, 2012. The dataset consists of 4000 verified malicious URLs (phishing sites, or the sites that contain known attacks, suspicious content, etc). We also collected an equal-size (4000) benign URLs from Yahoo directory [7].

IV. Detection Method

Given a finite-length string s, we can use Kolmogorov complexity to describe its complexity. In this work, we focus on the URL string which includes domain name, path, filename, and some arguments, etc as described before. Given a URL, we would like to judge whether the URL belongs to a malicious one or benign one, based on its Kolmogorov measure. Moreover, the proposed Kolmogorov measure can be combined with other features to form a powerful detector for malicious URLs.

TABLE II
DATA STATISTICS

Dataset	# of Benign URLs	# of Malicious URLs	Total	Malicious rate	Period
D_{TM}: Trend Micro	261,426,377	20,280	261,446,657	0.00776%	April 7 – April 16, 2011
D_{P+Y}: PhishTank + Yahoo	4,000	4,000	8,000	50%	June, 2010 – Feb. 2012

A. *Kolmogorov complexity*

The Kolmogorov complexity (or called Kolmogorov entropy, algorithmic entropy) [8] is one of the best measure to describe the complexity/entropy of finite objects. We can define the Kolmogorov complexity of a string s as the length of the smallest program (measured in bits) that can produce the string s, that is:

$$K(s) = \min_{p}\{|p| : U(p) = s\}\,, \tag{1}$$

where U is the universal Turing machine and $|p|$ measures the length of the program with program number p. Between different Turing machines their program lengths are up to a constant difference which is independent from the object that the program produces. Intuitively, we expect that wring a small program that produces a regular 2500-fold repetitive string "0001000100010001...0001" should be easier than writing a program that produces a string that encodes a series of coin-flipping trials.

We shall also define the conditional Kolmogorov complexity of a string t, given the information of another string s for free as follows:

$$K(t\,|\,s) = \min_{p}\{|p| : U(p,s) = t\}\,. \tag{2}$$

Here the universal Turing machine $U(p,s)$ simulates the program with program number p and takes the input s. That is, to produce string t, we do not need to spend effort (storage in the program) on describing s because s has a separate (input) storage to keep its content.

The Kolmogorov complexity and Shannon's entropy [9] are known to share some common properties. For two random variables X and Y, we have

$$H(X,Y) = H(Y,X)\,, \tag{3}$$
$$H(X,Y) = H(X) + H(Y\,|\,X)\,, \tag{4}$$

where $H(X)$ indicates the Shannon's entropy of X and $H(X,Y)$ and $H(Y\,|\,X)$ indicate the joint entropy and conditional entropy of X and Y respectively. On the other hand, given two strings s and t, we have

$$K(st) \sim K(ts)\,, \tag{5}$$
$$K(st) \sim K(s) + K(t\,|\,s)\,, \tag{6}$$

where st indicates the string concatenation of two strings s and t, the notation $\sim$ simply means that two values are equal, up to a constant that is independent from (any of) the discussed values for Eq. 5 and up to a logarithm term for Eq. 6. For the first formula, intuitively, we say that the programs to produce st and ts should have similar program lengths. The only thing that can make the difference between producing st and ts is how we can take advantage of the regularity in the concatenation point of st or ts. For Eq. 6, we say that producing st is very similar to the case where we write a program to produce s, and then write another program that can produce t, but given the information of s for free, and combine both programs together in the end.

B. *Kolmogorov Complexity Estimation by Compression Methods*

The Kolmogorov complexity is not computable in general (e.g., by the Gödel's incompleteness theorem or by [8]). We can use compression methods to acquire an approximation or an upper bound of Kolmogorov complexity. Given a finite URL string s and a compression method, we approximate the Kolmogorov complexity $K(s)$ of the string s by the length $g(s)$ of the compressed version of s.

We would also like to estimate the joint Kolmogorov complexity and conditional Kolmogorov complexity. To approximate the Kolmogorov complexity of st, we measure the length $g(st)$ of the compressed string of st. On the other hand, to approximate the conditional Kolmogorov complexity of $K(s\,|\,D)$, we measure the length difference between the compressed string of Ds and the compressed string of D. In math, we write:

$$g(st) \sim K(st)\,, \tag{7}$$
$$\begin{aligned} g(s \mid D) &\equiv g(Ds) - g(D) \\ &\sim K(Ds) - K(D) \sim K(s \mid D)\,. \end{aligned} \tag{8}$$

The last approximation in Eq. 8 is simply derived from Eq. 6. Some intuition behind such computation is that we expect a small quantity of $g(s \mid D) = g(Ds) - g(D)$ if s has similar patterns to *some* patterns in D. Usually, we expect a positive quantity or a very large quantity once s has very different patterns from *all* the patterns in D. Usually we have $g(s) \geq g(s \mid D) \geq 0$.

Finally, we need to choose a compression method as the candidate to approximate Kolmogorov complexity. Ideally, the better the compression ratio is, the better we can approximate Kolmogorov complexity. But a method with better compression ratio may indicate a slower computation in finding the common patterns for substitutions. In this work, we write a compressor based on a well-known algorithm *Deflate* [10] which is a variant of the classical *LZ77* [11] proposed by Lempel and Ziv in 1977. We also explore another similar algorithm called *LZ78* [12]. One of the major differences between *LZ77* and *LZ78* is that *LZ78* can compress a string given a known dictionary stored on the side, so called a *warm start* zipper whereas *LZ77* is a *cold start* version that always compresses from an empty dictionary. *LZ77* and *LZ78* are both lossless compression methods; also, both require low memory and are efficient in both encoding and decoding[4]. The bottom line of choosing an appropriate compression method is that we need to finish the compression faster than the content-based website analyzers, and it is better that we can finish the compression and the subsequent detection task in real time.

C. URL Classification Based on Kolmogorov measure

To classify a URL into a malicious one or not, we have two databases that keep all the URLs that we have seen in the history: one for the malicious part, denoted by D_m, and the other for the benign part, denoted by D_b. The scenario is, if we have a newly submitted URL s from a client, we would like to judge whether it is malicious or not. To answer that, we compare the URL s to D_m and D_b in turn to see which one includes patterns closer to the patterns in s. We compute $g(s \mid D_m) = g(D_m s) - g(D_m)$ and $g(s \mid D_b) = g(D_b s) - g(D_b)$ to find which one is smaller than the other. Intuitively, if $g(s \mid D_m)$ is smaller than $g(s \mid D_b)$ it means that s is closer to the malicious group and vice versa. To consider different sizes of databases D_m and D_b, we need an additional normalization step[5]. We define the Kolmogorov complexity-based measure for malicious URL detection as

$$\mathcal{M}_{D_m D_b}(s) = \frac{g(s \mid D_m) - g(s \mid D_b)}{g(s \mid D_m) + g(s \mid D_b)} . \tag{9}$$

If the measure $\mathcal{M}$ outputs a negative number, we predict s to be a malicious URL; otherwise, we predict s to be a benign one. Algorithm 1 shows the detailed steps for the proposed malicious URL detection method.

Algorithm 1: Kolmogorov-based malicious URL detection

```
   Input: a string $s$, a compressor $\mathfrak{C}$
   Output: a binary answer $y \in \{\text{malicious}, \text{benign}\}$
1  begin
2      Concatenate $D_m$ and $s$; also, $D_b$ and $s$ to form new
       strings $D_m s$ and $D_b s$ respectively ;
3      Compress $D_m s$ and $D_b s$ by $\mathfrak{C}$ ;
4      Measure the length of the compressed results,
       denoted by $g(D_m s)$ and $g(D_b s)$ for the malicious
       part and benign part respectively ;
5      let $g(s \mid D_m) = g(D_m s) - g(D_m)$ and
       $g(s \mid D_b) = g(D_b s) - g(D_b)$ ;
6      Compute $\mathcal{M}_{D_m D_b}(s) = \frac{g(s|D_m) - g(s|D_b)}{g(s|D_m) + g(s|D_b)}$ ;
7      if $\mathcal{M}_{D_m D_b}(s) \leq 0$ then
8          return "malicious" ;
9      else
10         return "benign" ;
11     end
12 end
```

[4]Efficient encoding is more important than efficient decoding in this case.

[5]It is not necessary if we just need to know whether the result is positive or negative. But it is helpful if we want to know "how malicious" a URL is.

We can compute $g(D_m)$ and $g(D_b)$ off-line to save some efforts. For the dataset D_{TM} from the Trend Micro company, D_m and D_b are collected based on the labels of content-based analysis. For the dataset D_{P+Y} from public domains, D_m is collected from PhishTank and D_b is collected from Yahoo directory. Figure 2 shows the distributions of Kolmogorov measure for the public domain dataset.[6] The Kolmogorov measure given by Eq. 9 can indeed separate malicious URLs from benign ones.

D. Combined with Other Methods

We can enhance the detection performance even further by combining the Kolmogorov measure with other detection methods to form a complete malicious URL detector. The improvement can be done based on two approaches. First, we should look for more features that can also give a good separation between malicious and benign URLs. Usually, the more diverse the features are, the better detection performance we can achieve. Based on a collected feature set, second, we can use a good classification method to effectively combine all the features for malicious URL detection. We have tried several different types of features to separate data of different properties, further discussed below. Afterwards, we utilize Support Vector Machines (SVMs) [13] as our classifier for the detection.

1) Huffman Coding: Some URLs may have common patterns no matter they are malicious or not. For instance, many URLs share common domain names, such as Google, Yahoo, Facebook or other public websites for blogging and malicious components may exist in those public websites, too. In this case, we tend not to trust a domain once we have many malicious URLs reported for that domain, even we still have the same domain, but a smaller number existed in the benign database.

If we are given three strings s_1 = "$ABCXXX$", s_2 = "$ABCYYABC$" and t = "ABC", the compressions on both $s_1 t$ and on $s_2 t$ shall give us good compression ratio[7] because they all share the common pattern "ABC". However, we would like to say that the string t is closer to the string s_2 because we observe higher frequency of common pattern "ABC" in s_2 rather than in s_1. We look for a compression method that can give us better compression for the string $s_2 t$ rather than the string $s_1 t$.

Huffman coding is known to use short codes to encode frequent items. Utilizing a compression method that is in co-operated with Huffman coding, we shall give a short outcome for a compression on the string $s_2 t$ rather than on the string $s_1 t$. The compression algorithm *Deflate* [10] takes Huffman coding into account for its compression; therefore, we expect a relatively short *Deflate* code for a string that has frequent patterns in it.

[6]It will be discussed further in Section V.

[7]We discuss simple short strings only for illustration. Readers should not be confused by the situation that short strings may not be compressed well by some common compressors.

TABLE III
EXPERIMENTAL SETTINGS

	Training set	Test set
D_{TM}	URLs collected from day 1 to day n or collected in day n only	URLs collected in day $n+1$
D_{P+Y}	2,000 malicious URLs in the period of June, 2010 to Feb., 2012 from PhishTank plus 2,000 from Yahoo	The rest of 4,000

2) Lexical Features and Statistical Features: We can consider some other features that can help us to detect malicious URLs. Some candidates are country code, with or without a particular file extension like "exe", etc. They belong to a group called lexical features. More examples of lexical features are those that were used in PhishDef [4]. Some other feature candidates include statistical features like the length of URL, the frequency of particular characters, etc. We should add some additional features to improve the performance of malicious URL detection.

3) Support Vector Machines: To effectively combine various features for malicious URL detection, we choose SVM to classify URLs into malicious or benign ones. We adopt LIBLINEAR developed by Fan et al. [14] for the classification task.

V. EXPERIMENTS

We conduct two series of experiments to evaluate how effective and how efficient the proposed method is. The first series uses only the Kolmogorov measure to detect malicious URLs. The goal of this series is to show how Kolmogorov measure can catch the different patterns between malicious and benign URLs. In the second series of experiments, we combine Kolmogorov measure with other features under the classification framework of SVM. The goal of the series is to demonstrate the Kolmogorov measure can help malicious URL detection in an "orthogonal" manner; that is, the Kolmogorov measure can give a large marginal help to other features for malicious URL detection.

We work on both datasets, the private dataset from Trend Micro D_{TM} and the public domain dataset D_{P+Y}. We use the private dataset to show the proposed method can indeed detect malicious URLs that are collected from the real world. We calculate Detection Rate and Missed Malicious Rate (defined in Eq. 10) for the evaluation. We also have the constraint that the proposed method must be efficient enough to follow the company's regulation, such as finishing filtering a million of URLs in one hour. Another dataset for evaluation is the public dataset D_{P+Y}. The goal to work on this dataset is to compare the proposed method to other methods that were proposed before. The complete datasets are described in Table II. Given the data, we would like to separate them into the training part and the test part, as shown in Table III for the evaluation. For the dataset D_{TM}, the prediction is operated in an online manner. We build a model based on the last n-day's data and use the model to predict the data coming for the next day. For the public dataset D_{P+Y}, we randomly permute the data into two equal parts and use one half of the data to build a model and use the model to predict the rest of the data.

A. Kolmogorov Measure Based Detection

1) The Data from Trend Micro: In this part of experiments, we evaluate the proposed measure on the dataset D_{TM} that was obtained from the Trend Micro company. The dataset contains around one million URLs per hour on average. Based on the company's requirement, we would like to detect as many malicious URLs as possible, with as few false positives as possible. We compute the Detection Rate (DR) and the Missed Malicious Rate (MMR) as follows:

$$\begin{aligned} DR &= \frac{TP+FP}{TP+FP+TN+FN}, \\ MMR &= \frac{FN}{TP+FN}, \end{aligned} \tag{10}$$

where TP, FP, TN and FN indicate *true positives*, *false positives*, *true negatives* and *false negatives* respectively; and the positives indicate malicious URLs in this work. Of course, while there is a trade-off, we would like both of DR and MMR to be as low as possible. A low DR gives a small amount of data for further analysis from a second or more detectors, possibly relying on content analysis; and a low MMR indicates that the detection method is sensitive enough to detect most malicious URLs. The company's requirement is to have both DR and MMR under 25%. For the computation efficiency, the lowest requirement for the proposed method is to be able to analyze all URLs in real time. It means that the one-hour data (one million of them) should be processed and given prediction completely in one hour.

First, we test the proposed Kolmogorov measure in an online manner. Because malicious URLs are significantly fewer than benign URLs, to avoid *unbalanced* learning, we sample only $1/600$ of the benign URLs to make both parts in a closer size. Figure 1 shows the hourly DR and MMR results. We use one-day data to build the model and test the model on the data of the next day and so on. We observe that both DR and MMR are basically below 25%.[8] The DR and MMR seem to have a periodic cycle every 24 hours. Both increase significantly during the night period. The interpretation is that during the night time most people are off from their work; therefore, reported URLs are less likely to be the common ones and more likely to be malicious.

Below we further discuss some different settings of Kolmogorov-based detection and their results. First, we consider using different training sets, from one to four days of data to build the model. As expected, the four-day model gives us the best performance, as shown in Table IV. In the compression of computing Kolmogorov measure, we can use different size of dictionary buffer to build different training models. In principle, a larger size of dictionary buffer produces better compression result; however, less efficient, as shown in Table V.

[8]Note that we use different scales for DR and MMR.

TABLE IV

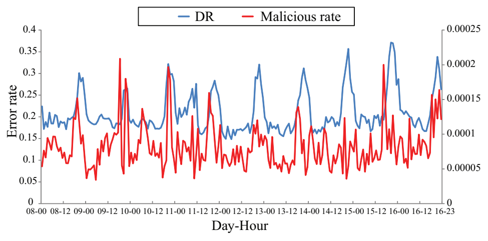
ONLINE PREDICTION ON THE DATASET D_{TM}. THE FOUR-DAY MODEL GIVES THE BEST RESULT (BOTH DR AND MMR), WHILE THE ONE-DAY MODEL GIVES THE MOST EFFICIENT RESULT (AROUND SEVEN MINUTES).

Date	1-day training		2-day training		3-day training		4-day training	
	DR	MMR	DR	MMR	DR	MMR	DR	MMR
Apr. 8	20.97%	22.06%	-	-	-	-	-	-
Apr. 9	19.94%	17.65%	18.53%	14.50%	-	-	-	-
Apr. 10	20.73%	19.76%	18.33%	16.41%	17.65%	14.20%	-	-
Apr. 11	21.80%	21.63%	19.50%	15.84%	19.02%	13.91%	17.54%	13.93%
Apr. 12	19.54%	25.36%	18.11%	17.25%	17.25%	13.57%	15.74%	13.05%
Apr. 13	20.01%	21.47%	18.22%	15.14%	16.78%	14.16%	15.59%	12.70%
Apr. 14	21.31%	20.34%	18.84%	15.59%	17.40%	12.86%	16.48%	11.99%
Apr. 15	22.90%	22.05%	20.45%	16.57%	17.89%	15.34%	17.18%	13.04%
Apr. 16	21.96%	17.65%	20.45%	12.79%	19.56%	9.85%	17.88%	8.59%
Avg. Perf.	21.02%	20.88%	19.05%	15.51%	17.94%	13.41%	**16.74%**	**12.22%**
Avg. time	**433** (s)		557 (s)		628 (s)		660 (s)	

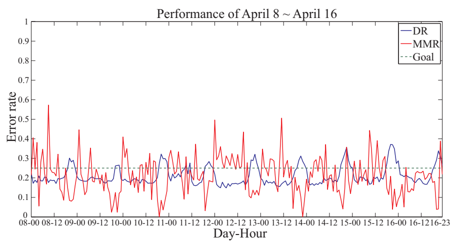
TABLE V

PERFORMANCE ON THE DATASET D_{TM} IN DIFFERENT SIZE OF DICTIONARY BUFFER.

Buffer size	1-day training			2-day training			3-day training			4-day training		
	DR	MMR	Time(s)	DR	MMR	Time(s)	DR	MMR	Time(s)	DR	MMR	Time(s)
Dict len. 2-5	25.29%	18.81%	297	24.42%	14.02%	597	24.86%	12.22%	602	22.55%	12.29%	650
Dict len. 2-6	22.71%	20.19%	335	21.49%	15.14%	571	20.53%	12.98%	605	20.06%	11.57%	653
Dict len. 2-7	21.08%	20.80%	377	19.53%	15.46%	582	18.13%	13.69%	584	17.70%	12.45%	675
Dict len. 2-8	21.02%	20.88%	433	19.05%	15.51%	557	17.94%	13.41%	628	**16.74%**	**12.22%**	660



(a)



(b)

Fig. 1. (a) DR vs. MMR in hourly detection. We observe a weekly cycle for both measures. The reported URLs during the night time are more likely to be malicious simply because people browsing the well-known URLs less frequently during the night period (roughly from 8pm to 1am). (b) DR vs. MMR after the tuning between DR and MMR.

2) The Data from PhishTank+Yahoo: We also test the proposed measure on the public dataset D_{P+Y}, which consists of data from PhishTank and Yahoo directory. Given the dataset, we demonstrate how the Kolmogorov measure can help us separate malicious URLs from benign ones.

Different from the private dataset D_{TM}, this dataset has no clear time information. The general approach to compute Kolmogorov measure in Eq. 9 is to randomly select appropriate databases D_m and D_b for malicious and benign groups respectively. We randomly divide the dataset into 10 groups where in each group, we have equal number of malicious and benign URLs. Afterwards, we use nine groups to build our databases D_m and D_b and use them to evaluate the Kolmogorov measure. If we need to separate training and testing processes, we can divide the dataset into the training set and test set before we evaluate the Kolmogorov measure.

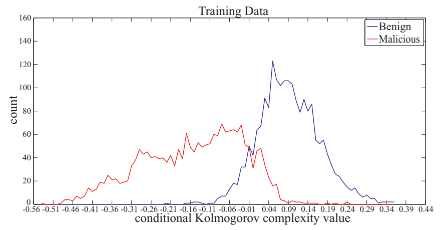


Fig. 2. The distribution of the Kolmogorov measure in Eq. 9 on the training data. The zero value appears to be a good cut point between malicious URLs (on the left) and benign URLs (on the right).

Figure 2 shows the distribution of Kolmogorov measure on the training set. We separate the dataset into two parts, 4000 URLs in each group for the latter use of classification training and testing; and in each group we further divide it into 10 batches (200 malicious URLs and 200 benign URLs) to compute the Kolmogorov measure. The separation given by Kolmogorov measure for the training set is shown in Figure 2. It demonstrates how informative Kolmogorov measure is to separate malicious URLs from benign ones.

B. Combine Kolmogorov Measure and Other Features

The goal of this experiment is to show that the proposed Kolmogorov measure can successfully improve the performance if we add the measure as an additional feature to a feature set without the measure. We take the feature set that was used in PhishDef [4] as the base feature set for comparison. There are four types of detection methods that we can consider for the comparison, listed below:

1) Kolmogorov measure only,
2) PhishDef features only classification,
3) PhishDef features and Kolmogorov measure combined classification, and
4) two-step classification using PhishDef features after a pre-screening by the Kolmogorov measure.

In these experiments, we would like to see whether the additional Kolmogorov measure can indeed help to detect malicious URLs in higher accuracy. The detail PhishDef features[9] should be referred to [4]. For each classification, we use a linear SVM method called LIBLINEAR [14] to train the model. The box constraint C is set to 2^5 at all times. Below we discuss each of the detection methods in more details.

The first method is simply the one that we check whether or not the Kolmogorov measure is negative to decide whether or not a URL belongs to a malicious one. We use the Kolmogorov measure that was used to generate Fig. 2 for the judgment.

The second method uses only the PhishDef features for classification. We repeat the feature collection procedure that was used to obtain the PhishDef's features in [4] to detect malicious URLs.

The third method uses the PhishDef features as well as the Kolmogorov measure (one more than the second method) for the classification.

The last method uses also both of the PhishDef features and the Kolmogorov measure, but based on a two-step approach for the detection. In the first step, we use the Kolmogorov measure to filter some "easy classified" URLs into the malicious group or the benign group; then in the second step, we use LIBLINEAR classifier on the subset of data that we do have enough confidence from the Kolmogorov measure.

Based on the distribution in Figure 2, we decide two thresholds to filter out some easy-classified URLs. We choose the thresholds so that we can control the malicious group to have larger than 99% accuracy; also, to have the accuracy larger than 98% in the benign group. The choice leads to the following rules: we call URLs that produce Kolmogorov measure lower than -0.057 as malicious ones and URLs that have the measure higher than 0.069 as benign ones. If a URL has the Kolmogorov measure between -0.057 and 0.069, we train a LIBLINEAR classifier to decide its final label. In this part, we need to help of PhishDef features. The complete classification result is shown in Table VI.

[9]Some of the PhishDef features are not used in this work, such as the WHOIS information. We intend not to use those that need information beyond the URL string itself.

TABLE VI
THE TWO-STEP CLASSIFICATION RESULT BASED ON THE KOLMOGOROV MEASURE AND THE PHISHDEF FEATURES, AND THE RULES USED FOR THE KOLMOGOROV PRE-SCREENING.

Threshold	< -0.057				> 0.069	
	Correct	Error	Correct	Error	Correct	Error
Malicious (Kol.)	1,319	49	–	–	–	–
Benign (Kol.)	–	–	–	–	1,200	19
Malicious (SVM)	–	–	544	88	–	–
Benign (SVM)	–	–	738	43	–	–

TABLE VII
THE COMPARISON RESULT OF FOUR DIFFERENT MODELS. THE ONE WITH THE KOLMOGOROV MEASURE PERFORMS BETTER THAN THE ONE WITHOUT THE KOLMOGOROV MEASURE; AND THE TWO-STEP METHOD PERFORMS THE BEST.

Model	TP	FN	TN	FP	Err. Rate
Kol. measure	1,726	274	1,859	141	10.375%
PhishDef features only	1,839	161	1,908	92	6.325%
PhishDef+Kol. measure	1,953	47	1,824	176	5.575%
2-step PhishDef+Kol. measure	1,938	62	1,863	137	4.975%

To compare the four detection methods, we find out that the Kolmogorov measure can improve the performance no matter it is done by a SVM classification given both of the PhishDef features and the Kolmogorov measure (from 6.325% to 5.575%), or by a two-step approach (from 6.325% to 4.975%). The two-step approach, using Kolmogorov measure for pre-screening first, followed by a SVM classification gives the best performance among the four methods.

VI. CONCLUSION

We proposed a Kolmogorov complexity-based measure for malicious URL detection. The measure is computed based on the compression algorithms *LZ77* and *LZ78*. Even the method is very simple in its concept, the experiments on both a private and a public domain datasets show that the proposed measure can indeed give good separation between malicious URLs and benign URLs and can operate in real time. The result on the commercial dataset gives us confidence that the proposed method can directly be applied to real world situations. On the other hand, we also test the proposed method on a public domain dataset to show how the proposed method can perform better than previously proposed method. We can also combine the proposed measure with other useful features, under a classification framework to give an even better detection performance.

ACKNOWLEDGMENT

This research is partially supported by Taiwan National Science Council Grants # 100-2218-E-011-010 and # 101-2218-E-011-009; also, the industrial cooperation grant # 100AG077. We also like to thank Walter Liu and Mark Kuo from Trend Micro for data pre-processing and valuable comments about this research.

REFERENCES

[1] TrendMicro, "Security in the age of mobility," http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt_security_in_the_age_of_mobility.pdf.

[2] 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*. New York, NY, USA: ACM, 2009, pp. 1245–1254.

[3] ——, "Identifying suspicious URLs: an application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 681–688.

[4] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," *IEEE INFOCOM*, pp. 191–195, 2011.

[5] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive Blacklisting to Detect Phishing Attacks," in *INFOCOM*, 2010.

[6] "PhishTank," http://www.phishtank.com.

[7] "Yahoo URL random generator," http://random.yahoo.com/bin/ryl.

[8] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications (3rd Ed.)*. Springer, 2008.

[9] C. E. Shannon, "A mathematical theory of communication," *Bell Syst Tech. J.*, vol. 27, pp. 379–423, 1948.

[10] S. David, "Data compression: The complete reference(4 ed.)," *Springer*, p. 241, 2007.

[11] A. Lempel and J. Ziv, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[12] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[13] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[14] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR a library for large linear classification." *Journal of Machine Learning Research,9 1871–1874*, 2008.