Detecting Machine Generated Domain Names Based on Morpheme Features

ZHANG Wei-wei GONG Jian LIU Qian
College of Computer Science and Engineering
Southeast University
NanJing, China
{wwzhang, jgong, qliu}@njnet.edu.cn

***Abstract*—To detect machine generated domain names, we proposed a method to exclude human generated domain names by analyzing the basic morphemes in the character strings of domain names (The basic morphemes in English are word roots and affixes while in Chinese are initials and finals). Experimental results show that the analysis of the morphemes can make a great progress in the efficiency and accuracy of detection.**

Keywords-DNS; domain name; morpheme; word root; word affix

I. INTRODUCTION

As an important infrastructure of the internet, DNS is primarily responsible for name resolution (mutual mappings between domain names and IP addresses), and closely associated with various network applications. So a new research trend in the field of network security is based on DNS activities to detect the malicious services, such as botnets, phishing sites, malicious software downloads.

Currently, the main defense has been the blacklist-based detection method. Due to the inherent defects of the maintenance and updating of the blacklist, attackers often algorithmically generate a long list of domain names. Recent botnets such as Conficker, Kraken and Torpig have brought in "DNS domain-fluxing" technique that bots algorithmically generate a large number of random domain names and request name resolutions. Spammers also randomly generate domain names in their spam emails to evade the blacklist-based detection.

To detect machine generated domain names, the main strategy is using machine learning to analyze the statistical lexical features in the character strings (length of domain name, character distribution, number of special characters, etc.). Related works can be traced back to 2007, Jan Goebel detects the Botnet IRC command and control channel by checking the specific key words and digital sequences in their IRC nicknames [1]; Ma (2009) detects phishing sites and malicious URLs in the e-mail advertising based on the statistics of lexical features (length of the domain names, host names, number of dots in the URL, etc.) [2]. Sandeep Yadav (2012) detects the automatically generated domain names by looking at distribution of alphanumeric characters as well as bigrams in all domain names that are mapped to the same set of IP-addresses [7]; Fariba Haddadi(2013), with the improved SBB neural network algorithm, alleviates the previous dependence of machine learning on prior knowledge [8]. Moreover, since 2009, Sanjeet, Pawan and Samuel have introduced the word segment techniques in the field of natural language to extract and restructure keywords from domain names for DNS probing and proactive forecast of blacklist [4-6].

Among the detection methods mentioned above, the machine learning method based on the statistical lexical features has lower computational complexity, but the attacker can easily evade detection through the prior feature statistics. Though the word segment methods can improve detection accuracy with the analysis on the semantic level, the conditions are too harsh that requiring domain names are generated entirely based on the dictionary. Taking advantages and disadvantages of both kinds of methods into account, we proposed a new method. Based on the basic lexical features, with further analyzing the basic morphemes in character strings to exclude human generated domain names, we can effectively improve the accuracy of machine generated domain detection. Moreover, compared with the huge corpus in the natural language model, the number of morphemes is relatively small (the number of word roots commonly used in English is only 900, and the total of initials and finals in Chinese spelling is only 47), which can guarantee a lower system overhead.

II. MOTIVATION

Detecting machine generated domain names is inherently a binary classification problem, which means determining a domain name is machine generated or human generated. Detection method can be roughly divided into two categories: 1) directly detecting by looking for the unique features which differ the machine generated domain names from the human generated domain names; 2) indirectly detecting by excluding the human generated domain names. Regardless of the detection method, the key problem is to find the features which can distinguish machine generated domain names from human generated domain names.

Considering in the progress of generating a domain name, both of them first need to select out a specified number of characters from a particular character set, then compose characters into labels with some certain rules, and finally combine with a selected suffix (top level domain). Due to the different motivation when generating domain names, there are some differences in the choice of character set, the length of domain name and the combination rule of characters, as shown in Table I.

TABLE I. FEATURES DESCRIPTION OF MACHINE / HUMAN GENERATED DOMAIN NAME

	Character Set	**Length of Domain Name**	**Combination Rule of Characters [3,7]**
Human Generated	[0-9][a-z]-_	length of domain name / number of labels / maximum length of label are all relative smaller	Containing English words or look like English (well-formed and pronounceable strings)
Machine Generated	[0-9][a-z]-_, and special characters (i.e. <>=)	length of domain name / number of labels / maximum length of label are all relative larger	No meaningful or pronounceable character strings

DNS first appeared mainly because it was difficult to remember and use the IP addresses. Thus the domain name was introduced for convenience and popular promotion, so human generated domain names are usually "simple", "pronounceable" and "easy to remember". However, machine generated domain names are often used by attackers in their malicious activities. To evade detection, the attackers often automatically generate "bulks" of "random" domain names.

According to traditional method based on machine learning, statistical lexical features include: length of domain name, distribution of alphanumeric characters, number of special characters, and so on. All these features are exterior features of domain names, which neglect the inherent features, so it is easy to evade detection. For example, intelligent machines can obtain these statistical lexical features of human generated domain names in advance, and then use them to automatically generate domain names, which can effectively evade the majority of the existing detection methods. Therefore, the key to solve the problem is to obtain the inherent features of domain names.

The inherent features of human generated domain names are the combination rules between the characters, which generally obey the natural language rules. For example, people in Europe and America often use English words to construct domain names, while the Chinese often use Chinese Spelling. The basic units of English are "word roots" and "word affixes (prefix/suffix)". Compared with millions of English words, there are only 900 common "word roots" and less than 500 "word affixes". Similarly basic morphemes in Chinese Spelling consist of 23 "initials" and 24 "finals". These basic morphemes constitute the inherent features of domain name character strings. Through computing morpheme features, such as "whether the character string contains morphemes", "the number of morphemes contained", "whether the morphemes can be further connected to form a word", we can exclude the human generated domain names, and indirectly detect the machine generated domain names.

III. DETECTION METHOD

The detection algorithm proposed in this paper consists of three parts: First, we propose a way to group together domain names. Then, for each such group, we compute a set of metrics to characterize their string features. Finally, based on a supervised machine learning algorithm C4.5, we could effectively identify each group as machine-generated or human-named.

TABLE II. METRICS SET

Metric	**Description**
m1	average length of domain names
m2	average number of labels in domain names
m3	average max length of label in domain names
m4	average ratio which digitals occupies in domain names
m5	average ratio which letters occupies in domain names
m6	average number of special characters in domain names
m7	average number of tokens in domain names
m8	average length of tokens in domain names
m9	average ratio which meaningful tokens occupies in domain names

A. Clustering Method

In order to compute metrics, we proposed a way to group together domain names with the same second level label (for example, "google" is the second level label of "www.google.com"). The idea behind is coming from the observation that many malicious domain names have the same second level label and even the same third level label. Different from the second level domain names used by Sandeep Yadav [7], we choose second level label due to the habit of the registrant that they usually registered the same second level labels with multiple top level domain names.

B. Metrics

As shown in Table II, the first six metrics (m1-m6) are the basic statistical lexical features (detail introduction can be seen in literature [1-3, 7, 8]). And the last three new metrics (m7-m9) are used to characterize the inherent features of human generated domain names. Token here refers to a meaningful string, such as a single morpheme, or a multiple adjacent morpheme group (prefix + root + suffix in English, initial + final in Chinese spelling), or a letter abbreviation.

Before compute these three new metrics, we proposed two heuristic methods to find out all morphemes contained in the domain names: 1) Start from the first letter of the character string, and try to find the longest possible token each time; If not found, continue to repeat the previous search progress from the second letter; Otherwise, continue to repeat the previous search progress from the letter follows close behind the longest token; Keep on searching until finding all the tokens; 2) First by travelling each letter of the entire character string, find out the longest token; Then use recursive algorithm on both sides to keep up looking for the longest token of the remaining strings; Recursively execute until finding all the tokens. The first heuristic algorithm follows the naming convention of most people and searches tokens from left to right, while the second heuristic algorithm is the supplementary to the first one. We performed both algorithms, and found out the best result (larger result of formula 1). After finding out all tokens, we could compute metric m7-m9 according to formula 2-4.

$$e_i = \frac{token\_num_i}{\max(token\_num_1, token\_num_2)} + \frac{token\_sum\_len_i}{\max(token\_sum\_len_1, token\_sum\_len_2)} \quad (1)$$

$$m7_i = \frac{\sum_{d \in D_i} |Tokens(d)|}{|D_i|} \quad (2)$$

$$m8_i = \frac{\sum_{d \in D_i} \frac{\sum_{t \in Tokens(d)} len(t)}{|Tokens(d)|}}{|D_i|} \quad (3)$$

$$m9_i = \frac{\sum_{d \in D_i} \frac{\sum_{t \in Tokens(d)} len(t)}{letter\_num(d)}}{|D_i|} \quad (4)$$

Among the above formulas, $token\_num_i$ represents the number of tokens found by the *ith* algorithm, $token\_sum\_len_i$ represents the total length of all the tokens found by the *ith* algorithm, D_i represents the *ith* domain set, *Tokens(d)* represents all tokens found out in domain name *d*, *len(t)* represents the length of token *t*, *letter_num(d)* represents the number of letters in domain name *d*.

C. Classification Algorithm

Since the decision tree classification algorithm provides accurate classification results [9], we selected supervised machine learning methods C4.5 to classify each group as machine-generated or human-named.

IV. Algorithm Evaluation

A. Dataset

We first describe data sets and how we obtained them:

- *JS_Domain_Set:* The domain names which can be correctly resolved in DNS response packets were collected from an access point on the Jiangsu Province boundary of CERNET from April 18 to 30 in 2013.
- *Human_Domain_Set*: Since the more popular the domain name, the higher probability of generated by human, we chose top 1000 domains in Alex list [10] without those once appeared in the blacklist [11-16]. Considering Alex only offers second-level domain, we further chose FQDNs (fully qualified domain names) from *JS_Domain_Set* which have the same "second-level domain name" with above.
- *Machine_Domain_Set* ： The domain names that once appeared in the blacklist [11-16] constitute this sample set.

TABLE III. Dataset

Dataset	Number of Second Level Labels	Number of Second Level Labels which have more than 20 sub Domains
T1	*H1*:380; *M1*:750	*H1*:155; *M1*:152
T2	*H2*:380; *M2*:751	*H2*:155; *M2*:151
JS_Domain_Set	873121	7140

TABLE IV. Comparison of detection results between *A1* & *A2*

Detecting Method	***T1* as a test set** ***T2* as a sample set**	***T1* as a sample set** ***T2* as a test set**
A1	Accuracy: 80.13% False Negative: 8.47% False Positive: 11.40%	Accuracy: 79.74% False Negative: 10.13% False Positive: 10.13%
A2	Accuracy: 47.89% False Negative: 5.86% False Positive: 46.25%	Accuracy: 58.17% False Negative: 3.92 % False Positive: 37.91%

In order to facilitate the access of C4.5 algorithm standard sample sets and the comparison between the subsequent results, we randomly halve each domain name sets of (ii)(iii) into *H1*, *H2*, *M1*, *M2*, and then exchange them to recompose into two standard datasets (*T1: H1-M1*, *T2:H2-M2*). *T1* and *T2* could be mutual training sample set and test data set, as shown in Table III.

For the detection of machine generated domain names, Sandeep Yadav proved by experiments that "bigrams distribution" is better than traditional statistical lexical features, such as "length of domain name", "number of numeric/alpha/dot/special characters" [7]. Since our method is also based on the traditional statistical lexical features, to prove the effectiveness of morpheme features, we chose "bigrams distribution" as a comparison which directly classifies machine generated domain names by calculating the KL distance mentioned in [7].

B. Results

For the convenience of description, *A1* represents the detection based on morpheme features in this paper, and *A2* represents the detection based on the distribution of bigrams in [7]. Both of them take *T1* and *T2* as samples in turn, and the other one as a test dataset.

As shown in Table IV, *A1* is much better than *A2* in the detection accuracy rate and false positive rate. Further analyze the reason that our method has relatively high false positive and false negative is as follows.

The large website, like "58.com.cn", contains 7000 sub domains. A lot of subordinates name arbitrarily, which shows similar features of machine generated domain names, and results in false positive.

As shown in Fig. 1, the core thought of our detection method is that human will follow the language rules when he generates a domain name. The trick of repeating words and suffixes of domain names back and forth is a blind spot of our detection method, so we need to further improve the algorithm.

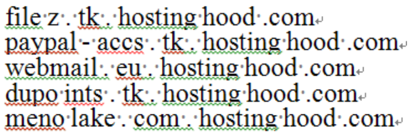
file·z·.·tk·.·hosting·hood·.com↵
paypal·-·accs·.·tk·.·hosting·hood·.com↵
webmail·.·eu·.·hosting·hood·.com↵
dupo·ints·.·tk·.·hosting·hood·.com↵
meno·lake·.·com·.·hosting·hood·.com↵

Figure 1. A Case of False Negative

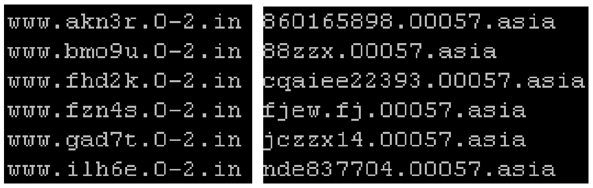
```
www.akn3r.0-2.in   860165898.00057.asia
www.bmo9u.0-2.in   88zzx.00057.asia
www.fhd2k.0-2.in   cqaiee22393.00057.asia
www.fzn4s.0-2.in   fjew.fj.00057.asia
www.gad7t.0-2.in   jczzx14.00057.asia
www.ilh6e.0-2.in   nde837704.00057.asia
```

Figure 2. Domain Name Examples Detected only by *A1*

Further apply the *A1* and *A2* algorithm to the actual measured dataset *JS_Domain_Set*. This paper mainly analyzes the domain names that can be detected by *A1*, but cannot detect by *A2*. As shown in Fig. 2, for the detection of such random strings, *A1* has higher detection efficiency.

Above all, the morphemes contained in domain names can be effectively used in detecting machine generated domain names, which has higher accuracy than traditional statistical method. Moreover, this method has inherent advantages in the detection of random character strings.

V. CONCLUSIONS

In order to effectively intercept attacker algorithmically generating a long list of domain names, statistical lexical features cannot meet the real needs due to its poor efficiency and higher false positives and false negatives. Moreover, intelligent machines could obtain these statistical lexical features prior, and generate names with the same statistical lexical features of human generated domain names. So we proposed a new method to exclude human generated domain names by analyzing the inherent morpheme features in the character strings. Experimental results show that our method has a higher accuracy rate and a lower false positive than the statistical method based on bigrams distribution. Moreover, our method has an inherent advantage in the detection of random character strings.

Our method mainly focuses on the letter sequences in the domain names, while the inherent characteristics of the digital sequences need further studying in the next stage. Finally, as to domain names generated entirely based on the dictionary, the algorithm needs to make a further improvement with some heuristics, such as repetitive morphemes detection. Nevertheless, morpheme features analysis still has a broad prospect.

REFERENCES

[1] Jan Goebel, Thorsten Holz, “Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation,” HotBots’07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, April 2007.

[2] S. S. J. Ma, L.K. Saul and G. Voelker, “Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs,” Proc. of ACM KDD, July 2009.

[3] Y. He, Z. Zhong, S. Krasser and Y. Tang, "Mining DNS for Malicious Domain Registrations," Proc. CollaborateCom, pp.1-6, 2010.

[4] Sanjeet Khaitan, Arumay Das, Sandeep Gain and Adithi Sampath, “Data-driven Compound Splitting Method for English Compounds in Domain Names,” CIKM '09 Proceedings of the 18th ACM conference on Information and knowledge management, pp. 207-214, 2009.

[5] Pawan Prakash, Manish Kumar, Ramana Rao Kompella and Minaxi Gupta, “PhishNet: Predictive Blacklisting to Detect Phishing Attacks,” INFOCOM'10: Proceedings of the 29th conference on Information communications, March 2010.

[6] Samuel Marchal, Cynthia Wagner and Thomas Engel, “Semantic Exploration of DNS,” NETWORKING 2012 Lecture Notes in Computer Science Volume 7289, pp. 370-384, 2012.

[7] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A. L. Narasimha Reddy, Supranamaya Ranjan, “Detecting Algorithmically Generated Domain-Flux Attacks with DNS Traffic Analysis,” IEEE/ACM Transactions on Networking, vol. 20, no. 5, October 2012.

[8] Fariba Haddadi, H. Gunes Kayacik, A. Nur Zincir-Heywood, Malcolm I. Heywood, “Malicious Automatically Generated Domain Name Detection Using Stateful-SBB,” EvoApplications'13 Proceedings of the 16th European conference on Applications of Evolutionary Computation, pp. 529-539, 2013.

[9] J.R. Quinlan, “Learning with Continuous Classes,” Proceedings of the 5th Australian joint Conference on Artificial Intelligence, Singapore: World Scientific, pp. 343 –348, 1995.

[10] Alexa, http://www.alexa.com/topsites

[11] PhishTank, http://www.phishtank.com

[12] DNS-Black-Hole, http://www.malwaredomains.com

[13] Malware Domain List, http://www.malwaredomainlist.com

[14] Abuse: AMaDA, https://palevotracker.abuse.ch/

[15] Abuse: Zeus Tracker, https://zeustracker.abuse.ch/

[16] P. Porras, H. Saidi, V. Yegneswaran, “An Analysis of Confickerd’s Logic and Rendezvous Points,” Tech. rep., March 2009, Avaliable: http://mtc.sri.com/Conficker/