Clustering Malware-generated Spam Emails With a Novel Fuzzy String Matching Algorithm

Chun Wei[*] weic@cis.uab.edu

Alan Sprague[*] sprague@cis.uab.edu

Gary Warner[*] gar@cis.uab.edu

ABSTRACT

In this paper, a fuzzy-matching clustering algorithm is introduced to group subjects found in spam emails which are generated by malware. A modified scoring strategy is applied in dynamic programming to find subjects that are similar to each other. A recursive seed selection strategy allows the algorithm to detect similar patterns even when the spammer creates a variation of the original pattern. A sliding threshold based on string length helps to minimize false-positives.

The algorithm proves to be effective in detecting and grouping spam emails using templates. It also helps spam investigators to collect and sort large amount of malware-generated spam more efficiently without looking at the email content.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications – Electronic mail

K.4.1 [**Computers and Society**]: Public Policy Issues – Abuse and crime involving computers

General Terms

Algorithms, Experimentation, Security.

Keywords

Electronic Mail, Spam, Data Mining, Forensics, Cyber Crime.

1. INTRODUCTION

Malware-generated spam emails usually exhibit certain patterns[3]. For example, a spam email subject like "Dear editor5@domain.org April 84% OFF" is generated by using a template with variation in certain fields. The template uses a pattern like: "Dear" followed by recipient's email address, followed by current month, followed by a discount percentage, followed by "OFF". If we see another subject like "Dear pyihy@domain.org May 84% OFF", then we know that they are probably generated by the same template, and therefore are likely tied to the same spammer. Detection of these patterns is useful to investigators and anti-spam workers who seek to attribute spam messages to the criminal or botnets[1][4] who sent them.

Spam emails like the example above can be grouped using fuzzy string matching, also known as approximate or inexact string matching. In this paper we apply fuzzy string matching to sets of spam subjects to identify related messages. The resulting groups are very likely to be generated by malware using similar templates.

2. CLUSTERING ALGORITHM

2.1 Similarity of Strings

The most common way to measure disagreement between strings is through edit distance, also referred as Leveshtein distance [2]. Because we want the similarity rather than distance, we use dynamic programming to find the alignment between a pair of strings s and t that maximizes the number of matches. The resulting number of matches between strings s and t is called their inverse Levenshtein distance, written as ILD(s,t).

However, we prefer the measure of similarity between a pair of strings to be always between 0 and 1. We want it to express the *portion* of the strings that match. The Jaccard coefficient accomplishes this but is defined for sets instead for strings. Therefore, we define the Jaccard coefficient for strings in a way analogous to sets. The *Jaccard* coefficient on sets A and B is defined by:

$\text{Jaccard}(A, B) = (|A \cap B|)/(|A| + |B| - |A \cap B|)$

where |A| and |B| are the size of set A and B, and $|A \cap B|$ is the size of intersection.

It yields a value between 0 and 1. Having the number of matches from the alignment, we define the Jaccard coefficient for strings s and t by:

$\text{Jaccard}(s,t) = \text{ILD}(s,t)/(|s| + |t| - \text{ILD}(s,t))$

2.2 Similarity on Spam Subjects

Next we define: a *token* is a sequence of nonblank characters in a subject; tokens are separated by blanks. A subject will be regarded as a sequence (or string) of tokens.

If tokens are compared using exact matching, then Jaccard("Viagra 80mgx30 pills", "Price for Viagra 50mgx60 pills") = 2/(3+5-2) = .333 because there are two matching tokens.

However, we want to allow tokens to partially match to each other, when two tokens p and q have the same number of characters, say n characters: *length(p)* = *length(q)* = n. Match(*p*, *q*) = m/n where m is the number of matching characters. The matching is done like this: for each character $(p_1, p_2, \ldots, p_n)$ in *p* and $(q_1, q_2, \ldots, q_n)$ in *q*, compare p_i with q_i. This computation is

[*] Dept. of Computer and Information Sciences, Univ. of Alabama at Birmingham, 1300 Univ. Blvd., Birmingham, AL, USA 35294. Tel: (205)934-2213

more rapid than dynamic programming, because we only consider tokens with same number of characters.

Now Jaccard("Viagra 80mgx30 pills", "Price for Viagra 50mgx60 pills") = 2.7/(3+5-2.7) = 0.51, because the partial match score of "80mgx30" to "50mgx60," is 0.7.

2.3 Clustering Spam Subjects

A pattern matching problem is: given a pattern P, find in a set of strings S={$S_1, S_2, \ldots, S_n$} all strings matching to P. Our problem is to find in S all interesting patterns and the corresponding strings. We applied a recursive seed selection algorithm. Let S be the set of subjects, and let S_0 = S. To form one group (one cluster): Select an arbitrary subject s in S_0, usually the first one. Then let Group(s) = {t: match(s,t) $\geq h_1$} (where h_1 is a similarity threshold) and remove Group(s) from S_0. After a group Group(s) is formed, attempt to enlarge it by selecting an s' in Group(s) (but s' relatively far from s), and adjoin all subjects t such that match(s',t)$\geq h_1$. s' is picked from Group(s) where match(s', s) is minimum. New s' is repeatedly selected until no new t can be drawn to the group. The result of this is that one group has been formed, and S_0 has shrunk. We repeat this, until S_0 is empty. Now we have partitioned the set S of subjects into groups, some large and some small. Next, discard all groups containing fewer than h_2 subjects (where h_2 is a threshold).

We also introduced a slack variable ε to fine tune the similarity threshold h_1,

$h_1 = base \pm \varepsilon$

ε is based on a standard normal distribution function of the string length (for spam subject, it is the number of tokens). The statistics (average string length and standard deviation) used in the function can be easily computed when loading the spam data.

For a pair of strings that are longer than average string length, ε is subtracted from the base threshold and the tuned threshold will be slightly lower. And for a pair of strings that is shorter, ε is added to the base threshold and the threshold will be slightly higher. Parameters in the function can be tuned to limit the value range of ε to be a small portion of the base threshold.

3. EXPERIMENTS

The spam emails were contributed by a spam researcher, who has a "catch all" policy to collect emails on his domains sent to non-existing email accounts which are believed to all be spam. Starting from March 2008, there are approximately over 10,000 up to 20,000 spam messages and 3000 to 6000 distinct email subjects every day. Note there is a 1:3 ratio between subjects and email messages.

We tested on three days data using different base thresholds. Results showed that a base threshold between 0.45 and 0.5 yields satisfactory accuracy rate (Table 1). Since this is the first stage of clustering spam emails, we want to be safe rather than generating too many false-positives. Other clustering methods will be built on the results generated by this method.

The recursive seed selection strategy was able to pick up variations in pattern. For example,

Viagra (Sildenafil) 100mg x 30 pills
Price for Viagra 100mg x 60 pills
Viagra (Sildenafil) 50mg x 30 pills $3.00 per pill
$89.95 Viagra 50mg x 30 pills buy now
Viagra (Sildenafil) 50mg x 30 pills $3.00 per pill buy now

The sliding threshold was able to put more restraint on shorter subjects, for example:

Group 1:

3a06c0.c15a38's discount #VUUkNK. BEEST Quaelity MedDs.
3a2061bf.5640c7's discount #MeEhEi. BEEST Quaelity MedDs.

Group 2:

RE: Discount
Discount

Although groups 1 and 2 generate the same similarity score, the sliding threshold will allow group 1 to join but not group 2.

Table 1: May 1st 2008 subject groups

Base thres hold	# of groups	# of grouped subjects	# of validated groups	# of subjects validated	# of groups can be merged
0.4	32	2024	28	1926	0
0.45	31	1898	28	1886	3
0.5	30	1450	29	1435	11
0.55	37	1292	37	1292	22

4. CONCLUSION AND FUTURE WORK

This paper used modified Levenshtein edit distance combined with Jaccard similarity coefficients to cluster spam subjects. The recursive seed selection and sliding threshold help to reduce the false-negative and false-positive rate. According to our data, around 1/3 of the email subjects each day match a pattern when using fuzzy matching. The patterns we found, combined with other clustering methods, will hopefully cluster large groups of emails which can be linked to a single spammer that created them. Future work includes making the system more adaptive to changes in patterns, including foreign character sets, and weighted tokens, and considering fuzzy matching of additional attributes, such as sender name or URL path.

5. REFERENCES

[1] Dagon, D., Guofei. G., Lee, C. P. and Wenke L. (2007) A Taxonomy of Botnet Structures. *In Proc. of the 23rd Annual Computer Security Applications Conference,* 325 – 339.

[2] Levenstein, V. I. (1966). Binary codes capable of correcting insertion and reversals. *Sov. Phys. Dokl.*, 10, 707 – 710.

[3] Nhung, N. P. and Phuong, T. M. (2007) An Efficient Method for Filtering Image-Based Spam. *In Proc. of the 2007 IEEE International Conference on Research, Innovation and Vision for the Future,* 96 – 102.

[4] Ono, K., Kawaishi, I. and Kamon, T. (2007) Trend of Botnet Activities. *In Proc. of , the 41st Annual IEEE International Carnahan Conference on Security Technology*, 243 – 249.