Computer Communications 54 (2014) 48–57

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Early filtering of ephemeral malicious accounts on Twitter ☆

Sangho Lee *, Jong Kim

Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea

ARTICLE INFO

Article history:
Received 18 January 2014
Received in revised form 7 August 2014
Accepted 9 August 2014
Available online 19 August 2014

Keywords:
Online social network
Twitter
Spam
Malicious account

ABSTRACT

Cybercriminals exploit a large number of ephemeral malicious accounts for conducting large-scale simple attacks such as spam distribution on online social networks. However, conventional detection schemes relying on account or message information take a considerable time to collect such information before running detection algorithms so criminals utilize their accounts until suspension and exploit others again. In this paper, we propose a new detection scheme to filter potentially malicious account groups around their creation time. Our scheme utilizes the differences between algorithmically generated account names and human-made account names to identify malicious accounts generated using the similar algorithms. For accounts created within a short period of time, we apply a clustering algorithm to group accounts sharing similar name-based features and a classification algorithm to classify malicious account clusters. As a case study, we analyze 4.7 million accounts collected from Twitter. Even though our scheme only relies on account names and their creation time, it achieves reasonable accuracy. Therefore, we can use it as a fast filter against malicious account groups to selectively conduct an in-depth analysis.

1. Introduction

Online social networks (OSNs), which allow people to establish and manage social relationships on Internet, suffer from malicious accounts. Many famous OSNs, such as Facebook and Twitter, allow any people to create accounts of their services to attract more users. They have a simple procedure for account creation, such as requiring an email address, because people dislike sophisticated verification processes. However, the simplified procedure allows cybercriminals to create malicious accounts for various attacks including spam, phishing, and malware distribution.

Among many OSNs, we focus on Twitter because it is one of the largest OSNs and its openness attracts many cybercriminals. Twitter has more than 140 million active users and 340 million messages created each day [1]. Due to this popularity, cybercriminals aim at exploiting Twitter to distribute malicious messages to its users. Some cybercriminals also develop tools for automated account creation and messages distribution; Twitter recently bring a lawsuit against them [1]. Detection of malicious Twitter accounts is therefore a crucial problem that demands countermeasures.

Researchers propose various *reactive* schemes to detect malicious social accounts and messages. Some schemes rely on the features of malicious accounts, such as a large number of messages containing URLs and a small number of friends [2–8]. However, these schemes require a time to collect account features *before* detection because they can observe the features *only after* the malicious accounts have initiated several malicious activities. In contrast, other schemes that inspect the malice of texts [9] or URLs [10,11] contained in a message do not require a time to collect information. However, these schemes also need to wait *until* malicious accounts post at least one distinguishable malicious tweet. Moreover, they demand more resources than the account-based detection schemes because they rely on complicated methods and the number of messages is fairly larger than the number of accounts.

Cybercriminals take two opposite approaches to bypass the existing detection schemes: (i) preparing advanced accounts to *evade* detection methods and (ii) creating a tremendous number of ephemeral accounts to *ignore* detection methods. First, criminals create and maintain advanced malicious accounts with human assistance. Recent studies [12,13] verify that criminals exploit crowdsourcing systems for acquiring human-assisted malicious accounts. The lifetime of the advanced malicious accounts is long because detection systems may not distinguish them from normal accounts, and, thus, criminals utilize them for long-term sophisticated attacks. However, because the costs of the advanced malicious accounts are high, criminals may not utilize them for large-scale simple attacks due to the trade-offs between costs and benefits. The recent countermeasure against the advanced malicious accounts also relies on humans for accurate detection [14].

☆ This research was supported by the MSIP, Korea, under the ITRC support program supervised by the NIPA (NIPA-2013-H0301-13-3002).

* Corresponding author. Tel.: +82 54 279 2915; fax: +82 54 279 1805.
E-mail addresses: sangho2@postech.ac.kr (S. Lee), jkim@postech.ac.kr (J. Kim).

http://dx.doi.org/10.1016/j.comcom.2014.08.006
0140-3664/

Second, cybercriminals create a huge number of *ephemeral malicious accounts* to ignore detection. The account-based and human-assisted detection schemes require a time to collect enough evidence for detection, so that *criminals can exploit their accounts until account suspension and other accounts again*. Message-based detection schemes can be a solution against this attack; however, they demand many resources and are weak against message obfuscation and conditional URL redirection [15]. To conduct this attack, criminals implement simple account creation programs and distribute them to botnets for avoiding IP address-based filtering of account creation. These ephemeral malicious accounts are suitable for large-scale simple attacks such as spam distribution. Thomas et al. [16] verify that most suspended Twitter accounts are created for the sole purpose of spam distribution. Many of them are the ephemeral malicious accounts we focus on.

We must consider a countermeasure against ephemeral malicious accounts: *finding potentially malicious accounts around their creation time*. An addressable characteristic of ephemeral malicious accounts, which we can observe around their creation time, is that they have algorithmically generated account names differing from human-made names. Twitter has a tremendous number of users and they already occupy a huge number of account names, so algorithmically creating account names that look like human-made names and not yet registered with Twitter is not an easy task. A similar problem exists on domain flux botnets which rely on algorithmically generated domain names, so that a number of studies inspect name characteristics to detect malicious domain names [17–19].

In this paper, we propose a new scheme to filter potentially malicious account groups around their creation time, which consists of (i) clustering newly created accounts that share similar name-based features and (ii) classifying the clustered accounts. To cluster accounts, we compute the log-likelihood that the account names are generated given a Markov chain and the lengths of the names. We create the Markov chain using verified account names and use an agglomerative hierarchical clustering algorithm. To classify account clusters, we train a Support Vector Machine (SVM) classifier using the name-based features of suspicious account groups. Evaluation results show that our scheme can cluster distinguished account names and classify benign and suspicious account clusters with reasonable accuracy. Therefore, our scheme is *a fast filter* to selectively conduct in-depth testing and monitoring of potentially malicious accounts.

The main contributions of this study are as follows:

- To the best of our knowledge, this is the first attempt to find potentially malicious account groups around the time of their creation. Our scheme can infer suspiciousness of malicious account groups before they initiate malicious activities.
- We propose a clustering method for proactively and accurately grouping accounts that seem to belong to the same campaigns.
- We derive a number of features to distinguish malicious account groups from benign account groups and develop an accurate classifier for account groups using these features.

The rest of this paper is as follows. In 2 we explain the manner in which our dataset is collected and analyze its characteristics. In 3 we briefly explain our filtering scheme. In 4 we introduce our clustering method used for grouping account names that share similar features. In 5 we explain the manner in which clustered account names can be classified. In 6 we discuss a possible evasive method against our scheme and the accuracy problem. In 7 we introduce related work. Lastly, we conclude this paper in 8.

2. Data collection and analysis

2.1. Data collection

We explain our dataset sampled from Twitter public timeline. Our dataset consists of 4,687,345 Twitter accounts created between April 2011 and October 2011 (214 days). We attempt to classify whether the collected accounts are benign or malicious according to whether they are active or suspended in March 2012 as a previous study [16] does. The number of active accounts is 3,618,096 (77.19%) and that of suspended accounts is 1,069,249 (22.81%). Twitter announced that 572,000 new accounts created on March 12, 2011 [20], so that, the number of accounts we focus on is approximately 3.83% of all the new accounts created during the 214 days.

We also collect the names of the verified Twitter accounts to consider them as ground truth of human-made account names. We obtain 18,289 verified Twitter accounts from Twitter's official accounts `@verified` on February 25, 2012.

2.2. Data analysis

We analyze our dataset to identify the characteristics of malicious accounts and their groups to utilize them for implementing a detection system.

Active and suspended accounts. We first analyze how many active and suspended accounts are created each hour using our dataset (Fig. 1). On average (median), 912 accounts are created each hour of which 704 are active (77.19%) and 208 are suspended (22.81%).

We then investigate *abnormal periods of time* in which the number of suspended accounts exceeds that of active accounts, among periods of 10, 30, and 60 min. Table 1 shows that the fractions of abnormal periods are fairly small (around 1%) in comparison with that of all suspended accounts (22.81%). Even we use a short period of 1 min, the fraction becomes only 4.14%. Therefore, without specialized methods, the low-rate creation of malicious accounts will go undetected.

Interval between account creation and first tweet. We measure the time interval between the creation of Twitter accounts suspended and their first tweets (also known as a dormancy period [16]) to estimate the *time advantage* of early filtering over the reactive detection schemes. We first analyze our dataset and find 50,572 first tweets from suspended accounts. We then measure the interval between the time of account creation and the time of first tweet, and obtain a complementary CDF (Fig. 2). On average (median), the suspended accounts post their first tweets 550 min

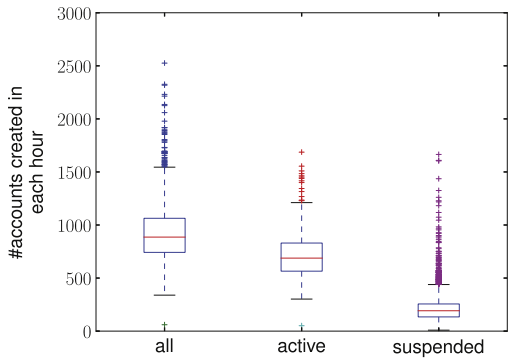


Fig. 1. Statistics of the number of active and suspended accounts created in each hour between April 2011 and October 2011 (boxplot).

Table 1
Number and fraction of periods of time where the number of suspended accounts are larger than that of active accounts.

Period (min)	Number	Suspended	Fraction (%)
60	5135	39	0.76
30	10,272	100	0.97
10	30,815	407	1.32
1	308,159	12,770	4.14

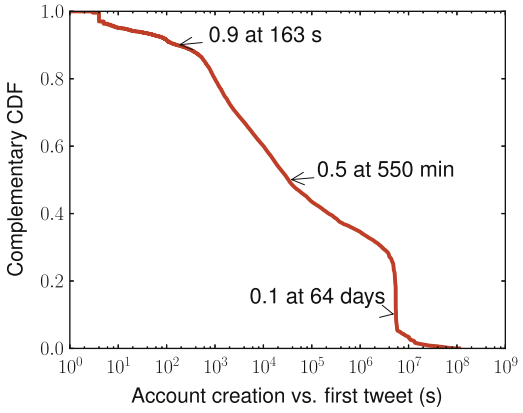


Fig. 2. Complementary CDF of the time interval between the creation of suspended accounts and the creation of their first tweets.

after their creation, so that the time advantage of early filtering over the existing reactive schemes is about *nine hours*.

Account groups. When analyzing the characteristics of malicious accounts, we must independently analyze the accounts belonging to the different spam campaigns because different campaigns may employ different strategies for generating accounts. An analysis of all suspended accounts can provide *mixed results* which are similar to the results of random account creation, so that we must identify different spam campaigns at first.

We can distinguish different spam campaigns using the URLs they advertised because each spam campaign usually advertises the same or similar URLs. We group accounts distributing the same URL to distinguish and identify the characteristics of accounts of the same campaign. When grouping, we ignore URLs of Alexa Top 1000 domains because many accounts distribute such URLs and, thus, there is no characteristics. If more than half of the accounts in a group are suspended accounts, we regard the group as a suspended group, otherwise, we regard it as an active group. Finally, we obtain 7265 active and 5723 suspended URL-based groups.

We analyze the account creation time of the top 20 largest active and suspended account groups by measuring (i) the time interval between account creation and (ii) the time interval between account creation and URL distribution. First, Fig. 3 shows that the time intervals between active account creation are longer than those of suspended account creation. The median creation time intervals for the top 20 active groups (2659–65,146 s) are longer than those for the top 20 suspended groups (22–614 s). Further, the interquartile ranges (IQRs) of the creation time intervals for the active groups (6702–102,963 s) are larger than those for the suspended groups (46–2254 s).[1] Second, Fig. 4 shows that the time intervals between active account creation and URL distribution are longer than those between suspended account creation and URL distribution. The median time intervals for the active groups (36.1–137.3 days) are longer than those for the suspended groups (1.4–96.0 days). Moreover, the IQRs of the time intervals are larger for the active groups (48.4–120.4 days) than for the suspended groups (0.1–16.3 days).

The analysis results show that the accounts belonging to suspended groups created in bulk within a short period of time just before spam distribution; namely, they are *burst* and *ephemeral*. In contrast, the accounts belonging to active groups are steadily created over a long time. Therefore, when we monitor the creation of accounts in Twitter in a short period of time, we can observe the creation of malicious accounts that will distribute the same URLs. However, we may not observe the creation of benign accounts that will distribute the same URLs because they are not created in bulk. As a result, if we confirm the tight relationship between the accounts created within a short period of time, we can find ephemeral malicious account groups belonging to the same campaigns.

Largest malicious account group. We analyze the largest malicious account group to identify how a single campaign affects Twitter. In our dataset, the largest group of ephemeral malicious accounts is the account group that distributes the spam URL `usyy.net` between August 2011 and October 2011 (which is the account group of size 24,405 in Figs. 3 and 4). During the period, 137,960 distinct accounts distribute the spam URL (3.6% of all accounts we collected). Among them, 55,034 accounts are created between August 2011 and October 2011. Our dataset consists of approximately 3.83% sample of all the newly created accounts (2.1) so that we can roughly say that the total number of accounts created between August 2011 and October 2011 and related with `usyy.net` is about 1.4 million. We anticipate that the cybercriminals who distribute `usyy.net` may create malicious accounts for every 5.7 s during the period using botnets to procure a large number of IP addresses and machines for such high-rate creation.

3. System overview and design goal

Our system for detecting ephemeral malicious account consists of two procedures: (i) clustering of accounts sharing similar name-based features and (ii) classification of account clusters (Fig. 5). We extract a number of accounts created within a period of time from the collected data, and conduct clustering and classification procedures on them. In the clustering phase, we attempt to distinguish potentially malicious accounts from benign accounts. In the classification phase, we build a classification model to classify benign and suspicious name clusters accurately.

We design our system as a fast filter for potentially malicious account groups like the fast filter for malicious webpages [21]. Our aim is not to implement *a full detection system* because we have no enough information for correctly judging whether the newly created accounts are malicious at the time of their creation, e.g., we have no profile information, no friendship information, and no messages of the accounts. Therefore, our system will transfer the potentially malicious account groups to backend detection systems that will conduct additional tests to identify whether a new account is bot (e.g., employing CAPTCHAs or human investigators) or carry out an in-depth analysis of its later requests (e.g., distributing tweets or establishing friendships) before allowing them. As a result, in our system, reducing missed detection (false negative) is more important than reducing false detection (false positive) because the backend systems can cancel the false detection but the missed detection is not.

Deciding time periods is important because it decides detection efficiency, detection accuracy, and system performance. Generally, a long time period increases detection accuracy (details are in 5.2.) However, it also decreases detection efficiency and increases execution time, because it forces our system to wait new accounts

[1] We use median and IQR because they are robust against outliers unlike mean and standard deviation.

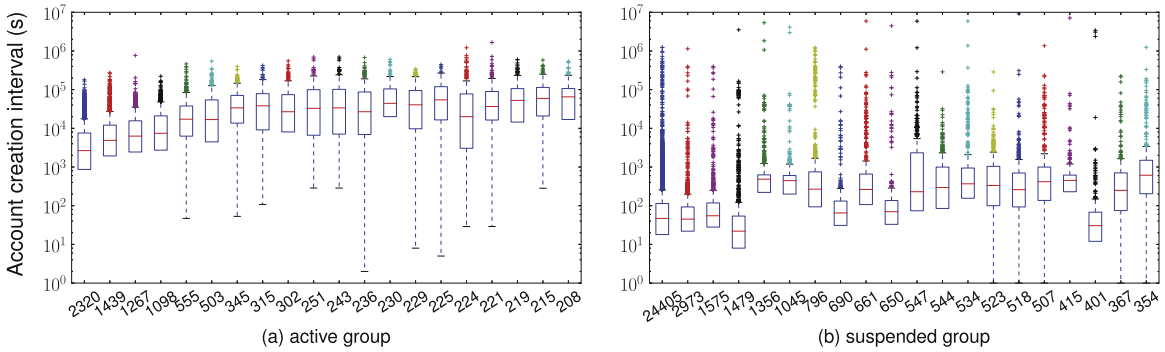


Fig. 3. Time intervals between creation of accounts that distribute the same URLs in October 2011. *x* axis represents the number of accounts in each of top 20 largest groups.

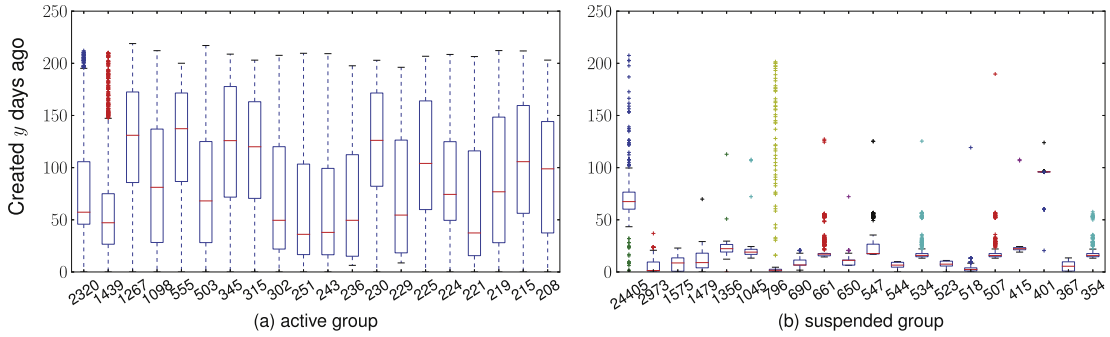


Fig. 4. Time intervals between creation of accounts and distributions of URLs in October 2011. *x* axis represents the number of accounts in each of top 20 largest group.

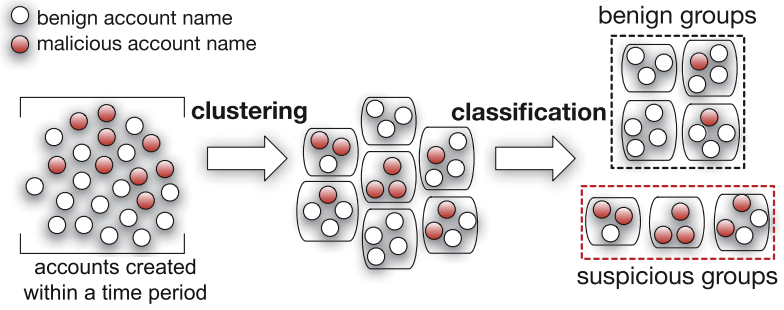


Fig. 5. System overview.

for a long time and run detection procedures with a large number of accounts (e.g., performance degradation of clustering is shown in Fig. 11). Throughout this paper, we use 10, 30, and 60 min as our reference time periods, because they show reasonable accuracy and performance, and can be directly compared with time-period-based grouping (Table 1).

4. Clustering of account names

In this section, we explain our clustering method to group names sharing the similar features and evaluate the clustering results.

4.1. Agglomerative hierarchical clustering

We apply an agglomerative hierarchical clustering algorithm [22] to cluster account names. This algorithm first creates n clusters having a single name object and then continuously merges similar clusters into larger clusters until a single cluster is formed or specific termination conditions are satisfied. When we examine two clusters C_i and C_j, we calculate their *maximum distance*,

$$d_{\max}(C_i, C_j) = \max_{n_i \in C_i, n_j \in C_j} \{dist(n_i, n_j)\},$$

where $dist()$ is the distance function (refer 4.2). If $d_{max}(C_i, C_j)$ is less than a predefined threshold value, we merge C_i and C_j, and, otherwise, we move on to examine other clusters. The algorithm terminates when all the maximum distances between clusters are larger than or equal to a threshold value.

4.2. Distance function

To measure the distance between two names n_i and n_j for clustering, we compute the likelihood that the two names are generated given a Markov chain m, which are notated as $\mathcal{L}(n_i|m)$ and $\mathcal{L}(n_j|m)$. We also count the lengths of the two names, $L(n_i)$ and $L(n_j)$. We define a distance function $dist(n_i, n_j)$ as

$$\sqrt{\left(\frac{\log \mathcal{L}(n_i|m) - \log \mathcal{L}(n_j|m)}{\log \mathcal{L}_s}\right)^2 + \left(\frac{L(n_i) - L(n_j)}{L_{\max} - L_{\min}}\right)^2},$$

where $\mathcal{L}_s$ is the smallest likelihood value observed in our dataset, L_{max} is the allowed, maximum length of account name, 15, and L_{min} is the allowed, minimum length of account name, 1. We use the log-likelihood for actual computations because the likelihood values are very small to be compared.

Markov chain. We create a Markov chain using n-grams of Twitter account names. Fig. 6 shows examples of unigram and bigram Markov chains generated using seven names with a character set $\Sigma = \{a, b, \ldots, z\}$. Each vertex represents either unigram or bigram, and each edge represents the transition probability between two unigrams or bigrams. Given the unigram Markov chain M_u and bigram Markov chain M_b, the likelihood that a name *caron* is generated is

$$\begin{aligned}\mathcal{L}(caron|M_u) &= \mathcal{L}(c, a, r, o, n|M_u)\\ &= \mathcal{L}(c) \times \mathcal{L}(a|c) \times \mathcal{L}(r|a) \times \mathcal{L}(o|r) \times \mathcal{L}(n|o)\\ &= 1 \times 1 \times \tfrac{1}{2} \times 1 \times \tfrac{1}{3} = \tfrac{1}{6},\end{aligned}$$

$$\begin{aligned}\mathcal{L}(caron|M_b) &= \mathcal{L}(ca, ar, ro, on|M_b)\\ &= \mathcal{L}(ca) \times \mathcal{L}(ar|ca) \times \mathcal{L}(ro|ar) \times \mathcal{L}(on|ro)\\ &= 1 \times \tfrac{1}{2} \times \tfrac{1}{2} \times \tfrac{1}{3} = \tfrac{1}{12},\end{aligned}$$

when we set initial probabilities $\mathcal{L}(c) = 1$ and $\mathcal{L}(ca) = 1$.

To measure the similarity in names, we compute the likelihood that the names can be generated given a Markov chain. If names are generated using the same algorithm, their likelihood values may be similar. We create a Markov chain using bigram sequences of the verified Twitter account names as shown in Fig. 6b. We compute the likelihood of top 20 largest active and suspended account groups in October 2011 given the Markov chain (Fig. 7). The median likelihood values of the top 20 active groups (between $10^{-17.6}$ and $10^{-14.8}$) are larger than those of the top 20 suspended groups (between $10^{-29.5}$ and $10^{-17.0}$). Moreover, the IQRs of the likelihood values of the active groups (between $10^{-13.6}$ and $10^{-11.4}$) are larger than those of the suspended groups (between $10^{-24.1}$ and $10^{-12.2}$). Therefore, most of the account names in active groups can be generated using the Markov chain and they are not closely related to each other. In contrast, most of the account names in suspended groups cannot be generated using the Markov chain and they are closely related to each others.

Name length. The second parameter for identifying malicious accounts is the name length. Twitter accepts account names having lengths of 1 to 15 characters. We found that cybercriminals tend to generate long account names to avoid name collisions because most of the short account names are in use by other users. Thus, if names are generated using the same algorithm, they may be similar and long. We examine the distribution of name lengths in the top 20 largest active and suspended account groups in October 2011 (Fig. 8). The median name lengths of the top 20 active groups (9–13) are shorter than those of the top 20 suspended groups (9–15). Furthermore, the IQRs of the name lengths of the active groups (3–5) are larger than those of the suspended groups (1–3). These results show that most of the account names in active groups are relatively short and these names do not have similar lengths. On the other hand, the most of the account names in suspended groups are long and these names have similar lengths.

4.3. Evaluation

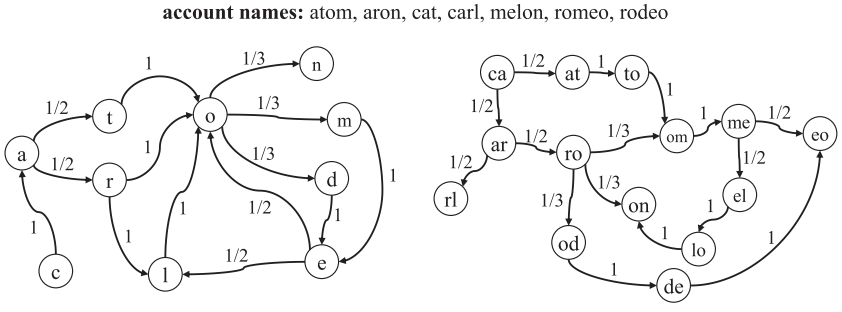
To evaluate the clustering accuracy, we consider two measures: (i) the fraction of suspended clusters and (ii) the fraction of ignored suspended accounts. We also discover a reasonable distance threshold and measure the performance of clustering.

Distance threshold. We must find a reasonable distance threshold to increase the number of *suspended clusters* while decreasing the number of *ignored suspended accounts*. Suspended clusters are the clusters that consist of a larger number of suspended accounts than that of active accounts. We demand a large number of suspended clusters for increasing detection coverage. These suspended clusters are *positive* samples in the later classification procedure.

Ignored suspended accounts are the suspended accounts belonging to small clusters having <10 accounts. Our system ignores such small clusters because they do not have enough statistical features for the later classification. Such disregard, however, will decrease the detection coverage so that we must reduce the number of ignored suspended accounts.

We attempt to discover the proper distance threshold value by applying our clustering algorithm to sampled accounts while varying threshold values. We select accounts created in randomly sampled 1280 periods of 60 min from our dataset (about 53 days), and split them to compose 7680 periods of 10 min and 2560 periods of 30 min. The number of accounts created in the sampled periods of time is 1,164,635. We then apply the clustering algorithm on them while varying distance threshold values from 0.1 to 0.7. Fig. 9 shows that the fractions of ignored suspended accounts decrease according to the increase of distance threshold and the fractions of suspended clusters have local maximums. Lastly, we select 0.5, 0.3, and 0.2 as the distance threshold values for the periods of 10, 30, and 60 min, respectively, because they achieve <10% ignored suspended accounts along with ~15% suspended clusters.

Suspended clusters. We apply our clustering method to account names created within three different periods of time (10, 30, and 60 min) and compare the fractions of suspended clusters with those of the time-period-based grouping. With our clustering method, we obtain up to 15.3% suspended clusters with the period of 60 min (Table 2). This fraction is about four times better than that of the



(a) Unigram Markov chain. (b) Bigram Markov chain.

Fig. 6. Examples of Markov chains.

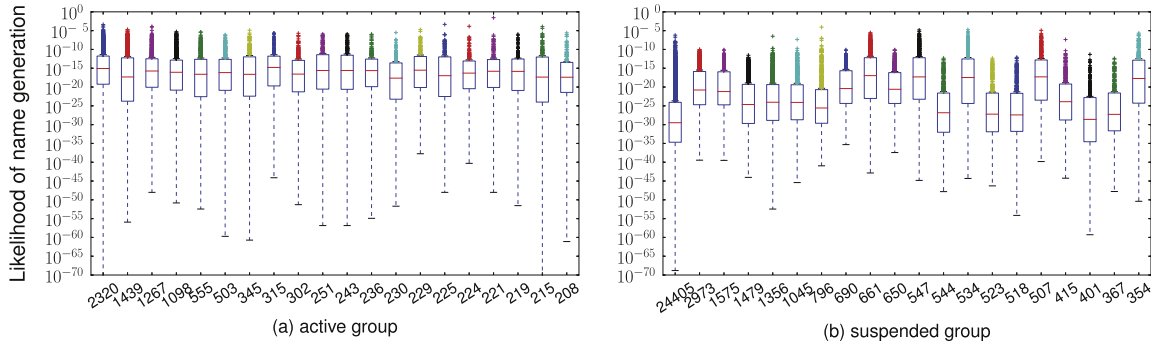


Fig. 7. Likelihood values of the names of the accounts that distribute the same URLs in October 2011 given the Markov chain. *x* axis represents the number of accounts in each of top 20 largest group.

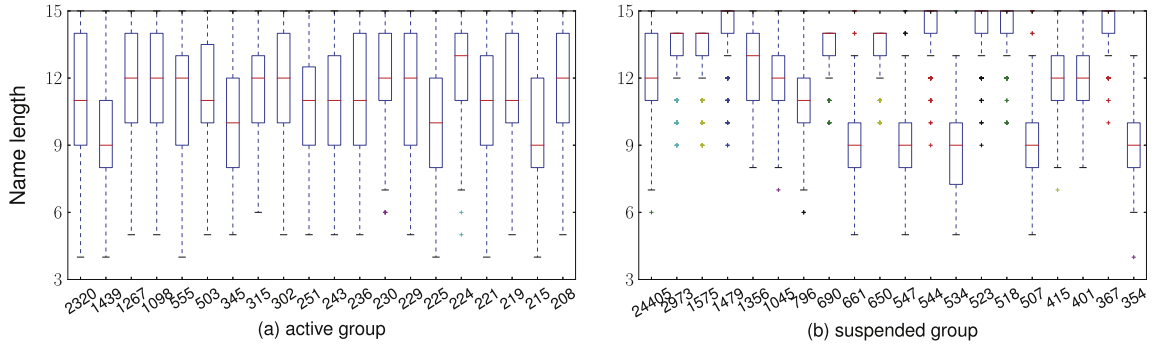


Fig. 8. Lengths of the names of the accounts that distribute the same URLs in October 2011. *x* axis represents the number of accounts in each of top 20 largest groups.

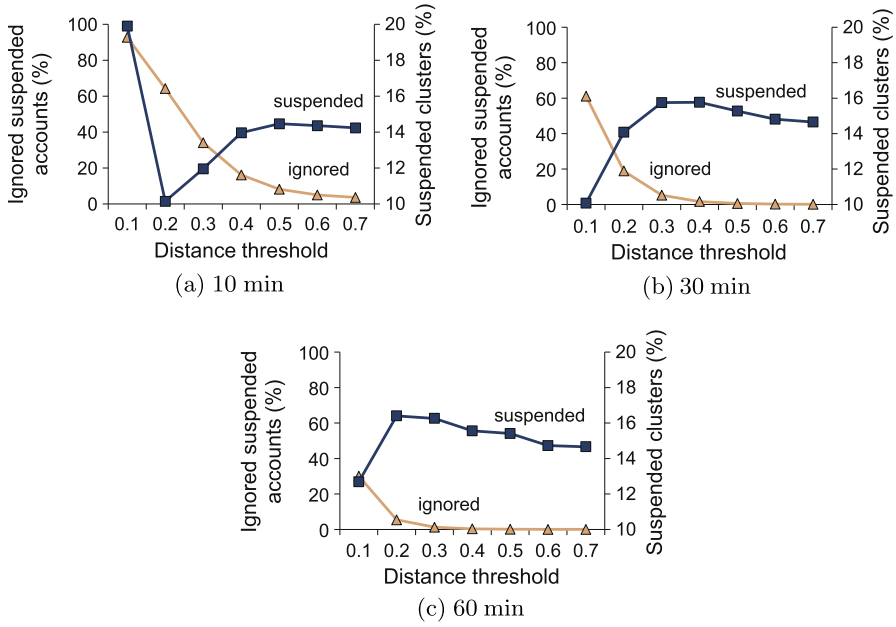


Fig. 9. Fractions of ignored suspended accounts and suspended clusters according to distance threshold values.

time-period-based grouping with 1 min (Table 1). Therefore, our clustering method can create separated groups of suspended names and active names better than the time-period-based grouping does. In addition, Fig. 10 shows the statistics of the number of clusters in each period and their sizes. The median numbers of clusters in the periods of 10, 30, and 60 min are 5, 15, and 21, respectively, and the median sizes of clusters are 24, 23, and 32, respectively.

Ignored suspended accounts. Although we achieve high fractions of suspended name clusters, we need to identify the number of suspended accounts belonging to small clusters because our scheme ignores them. Table 3 shows that the fractions of ignored suspended accounts are <10% and, especially, the fraction is only 2.41% with the period of 60 min. Therefore, our scheme can investigate most of the suspended accounts.

Table 2
Fractions of the suspended clusters with different periods of time.

Period (min)	#Clusters	Suspended	Fraction (%)
60	114,563	17,533	15.30
30	166,789	23,634	14.17
10	164,570	21,727	13.20

Clustering time. We measure the required time for clustering n names on a machine that has two Intel six-core Xeon X5650 2.67 GHz CPUs and 24 GB of main memory where $n \in \{500, 1000, \ldots, 4000\}$. To reduce clustering time, we use 12 threads to compute the distance between the names. The result in Fig. 11 show that the clustering time exponentially increases according to the number of account names. In our dataset, on average, about 912 accounts are created in 60 min (Fig. 1). Our system can cluster 1000 names in 32.94 s so that the performance of our system is sufficient to analyze our dataset.

5. Classification of name clusters

In this section, we explain the features of account groups to build our classification model and evaluate the classification results.

5.1. Features

To classify account groups, we derive a number of name-based features from Twitter account groups.

Distance between unigram/bigram distributions for a name group and for verified names. Malicious name groups are algorithmically generated, so their unigram and bigram distributions differ from those for the human-made verified names. Fig. 12a,b show that the distances between the unigram and bigram distribution for the active clusters and for the verified names are shorter than the distances between those for the suspended name clusters and for the verified names. When measuring the distances, we use earth mover's distance (EMD) because it shows better results than another famous measure Kullback–Leibler divergence (KLD) with our dataset.

Distance between length distributions for a name group and for verified names. In our dataset, we identify that the length distribution of malicious name groups differ from those of benign name groups; the malicious names are usually long to avoid name collisions (Fig. 8). Fig. 12c shows that distances between the length distributions for the active name clusters and for the verified names are shorter than the distances between those for the suspended name clusters and for the verified names.

Table 3
Fractions of the ignored active and suspended accounts against all of active (3,618,096) and suspended (1,069,249) accounts.

Period (min)	Active		Suspended	
	#	%	#	%
60	98,109	2.71	25,797	2.41
30	290,878	8.04	100,845	9.43
10	183,634	5.08	65,858	6.16

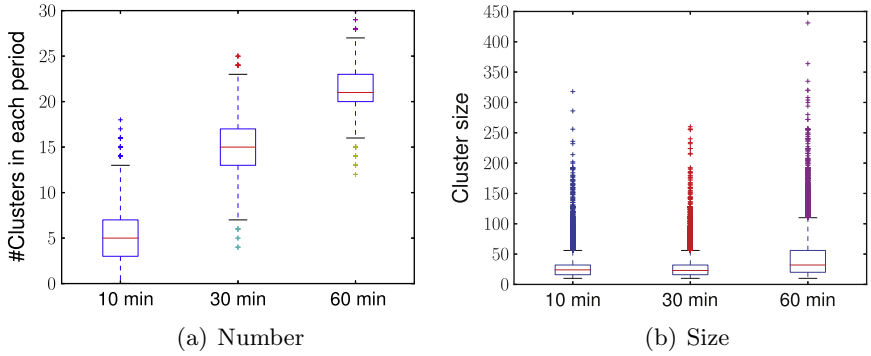
Distance between position-wise unigram distributions for a name group and for verified names. In our dataset, we observe several malicious name groups that have different unigram distributions at each character position from those for the verified names. Fig. 12d shows that the average distances between the position-wise distributions for the active name clusters and for the verified names (average between distance values for each of 15 character positions) are shorter than the average distances between those for the suspended name clusters and for the verified names.

Distance within position-wise unigram distributions for a name group. In our dataset, we also discover that the unigram distributions of each character position of the verified names are similar to each other and those for the active name groups are also. However, the unigram distributions of each character position of the suspended name groups are dissimilar to each other. Fig. 12e shows that the average distances within the position-wise distributions for the active name clusters are shorter than those for the suspended name clusters.

Edit distance within a name group. Algorithmically generated names tend to dissimilar to each other because they are created in random. To measure the dissimilarity, we compute the average edit distance between names in a group. The edit distance (the Levenshtein distance) between two strings represents the minimum number of actions, such as insertion, deletion, or substitution, necessary to transform a string into another string. Fig. 12f shows that the average edit distances within active name clusters are shorter than those within the suspended name clusters.

5.2. Evaluation

We compare classification accuracy of name groups with three grouping methods: using our clustering method, using the same URLs distributed, and using periods only. We apply our clustering method to the names of accounts created in the periods of 10, 30, and 60 min (Table 2). For comparisons, we consider the time-period-based groups and the URL-based groups explained in 2. We build SVM classifiers using the LIBSVM [23] library. For evaluation, we utilize two measures: false negative rate (FNR) and false



(a) Number (b) Size

Fig. 10. Statistics of the number of clusters in each period and their sizes.

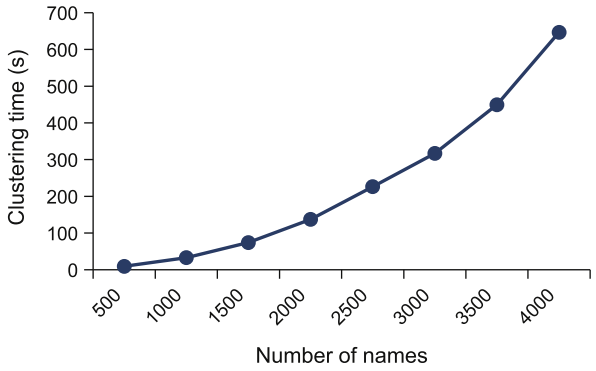


Fig. 11. Required time for clustering account names.

positive rate (FPR), where FNR shows the fraction of positive samples (suspended groups) treated as negative (active groups) and FPR shows the fraction of negative samples treated as positive.

Our goal is decreasing FNR while maintaining relatively low FPR by adjusting the ratios between positive and negative samples of training data, like related studies [9,10,24]. We conduct *fivefold cross validation* for evaluation by randomly splicing the overall dataset into five folds and selecting one of them for test and the remaining four folds for training. First, we sub-sample the training data to achieve four different training ratios between positive and negative samples, 1:1, 2:1, 3:1, and 4:1, and the testing data to achieve the 1:1 ratio for validation. We then train SVM classifiers with the training data and classify the testing data using the classifiers. We repeat this process five times with the different folds for testing and compute the mean values of the five FNRs and FPRs.

Fig. 13 shows that the classification accuracy with our clustering method is better than that of the URL-based grouping. The clustering with the period of 60 min has the lowest FNR and FPR, which are better than the results of the URL-based grouping. Among the four different training ratios for the period of 60 min, we expect that the 4:1 is a reasonable ratio because it has ~2% FNR (1.98%) and ~20% FPR (20.74%). Fig. 13 also shows that the FNRs of the time-period-based grouping are very low in some cases. However, Table 1 shows that it already misses a number of malicious accounts and has high FPRs, so that the effectiveness of the time-period-based grouping is limited.

Classification time. We measure the time required to classify whether a name cluster is suspicious. The classification time includes the feature extraction time and the prediction time with a trained model. The features explained in 5.1 can be simultaneously extracted, so we only consider the most time-consuming process, which is computing the average distance within position-wise unigram distributions. This process involves two phases: (i) generation of position-wise unigram distributions for a name cluster, for which our system requires about 2.78 ms for 30 names; and (ii) computing the average EMD of the distributions, for which our system requires about 81.12 ms. Therefore, our system requires about 83.90 ms to generate a feature vector of a name cluster. To predict whether a feature vector is suspicious, our

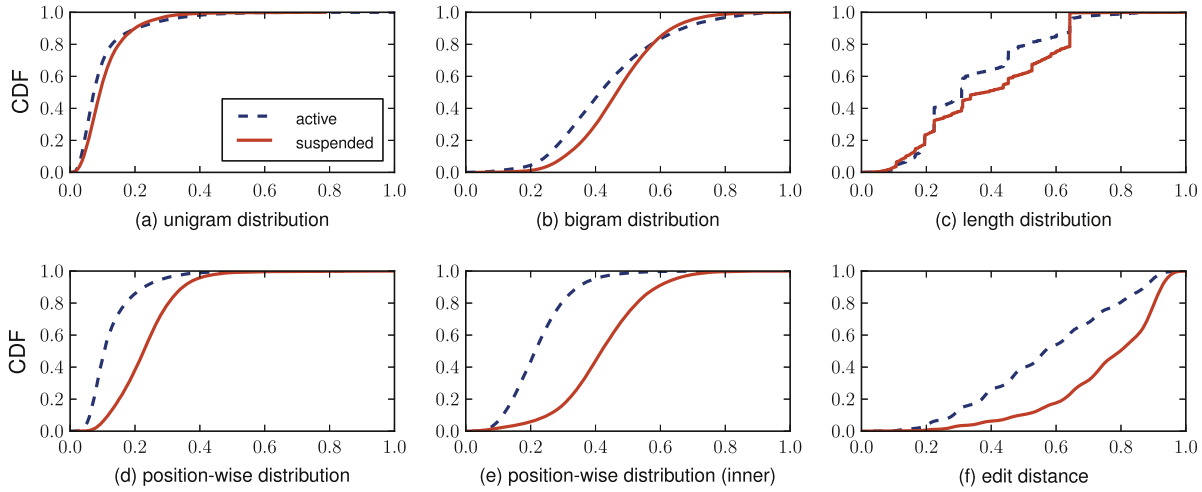


Fig. 12. Statistics of features of account name clusters (with a period of 60 min). (a)–(d) are the distances between active and suspended clusters and the verified names. (e) and (f) are the distances within active and suspended clusters. Distance values are normalized to lie between 0 and 1.

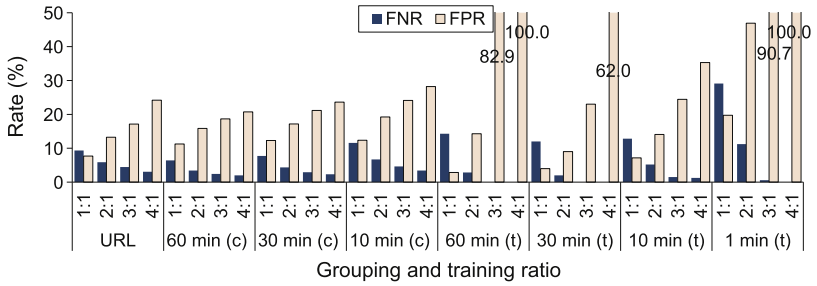


Fig. 13. Classification accuracy with different grouping methods and training ratios (positive versus negative). (c) represents our clustering method and (t) represents time-period-based grouping.

system needs about 0.28 ms. Totally, our system requires 84.18 ms to classify a single cluster, which is fairly shorter than the clustering time.

5.3. Sliding window

We applying a sliding window technique to reduce the latency of our scheme. The evaluation results in 2 show that clustering with a period of 60 min is the best result of ours. However, we cannot wait for 60 min to collect account names as we demand a (near) real-time detection scheme. To overcome this limitation, we utilize the sliding window technique, e.g., in every δ s, we apply our scheme to the account created in the previous 60 min ($\delta < 3600$).

We can check the accounts created in the previous 60 min at least once in every minute using our system. The lower bound of δ depends on the performance of clustering and classification. In our system having two Intel six-core Xeon X5650 2.67 GHz CPUs and 24 GiB of main memory, clustering and classifying $\sim$1000 accounts created in 60 min takes $\sim$35 s. Thus, the lower bound of δ is $\sim$35 when we use a period of 60 min.

6. Discussion

In this section, we discuss the limitation of our scheme, how attackers evade our scheme, and how to obstruct such evasion. We also consider how to increase the accuracy of our scheme.

6.1. Limitation

The main limitation of our scheme is that its detection rate depends on the characteristics of fabricated account names. Cybercriminals who recognize our scheme would attempt to change the characteristics of their account names. For example, when cybercriminals use words listed some dictionaries (e.g., English, Chinese, Russian, or Korean dictionaries) written with Latin letters to generate account names instead of random strings, the characteristics of such account names not much differ from normal account names. Furthermore, cybercriminals can use a set of normal account names when fabricating account names to imitate their characteristics. To cope with such cybercriminals, we need to frequently update a set of malicious account names and retrain our classifier as explained in 6.2.

6.2. Evasion

Advanced name generation. Cybercriminals who recognize our scheme can develop advanced name generation algorithms for evasion. Among possible algorithms, we consider the worst case: mimic the name-based features of active account groups, which is exactly opposite to our scheme. Although criminals also require a time to collect account names, it will be the best evasive method against our scheme.

Our evasive method is as follows. First, we create a number of groups of randomly selected active accounts. Next, we derive Markov chains using bigram sequences of each group. Lastly, we generate account names using the Markov chains.

To test the evasive method, we create an experimental dataset: (i) randomly select 500,000 accounts from our dataset and compose 100 groups of 5000 accounts; (ii) derive Markov chains using each group's bigram sequences; (iii) generate 500 names using each Markov chain (500×100); and (iv) mix them with 200,000 active account names randomly selected from our dataset. This ratio is similar with the ratio between suspended and active accounts in our collected dataset (1:3.48). Among the 50,000 names we generated, 4759 names are associated with active Twitter accounts and 106 names are associated with suspended accounts (9.73%). We then perform our clustering method on every 1000 names of the experimental dataset for simulating the period of 60 min. The number of final clusters is 9365, where 849 clusters have a larger number of generated names than active names (9.07%).

Next, we attempt to classify the experimental dataset without or with a retraining procedure. When we apply the old classifier (60 min, 4:1 training ratio, and trained without the experimental dataset) to the experimental dataset, we miss most of the malicious names (Table 4). However, when we add 4/5 of the experimental dataset to training data while maintaining 4:1 ratio, retrain a classifier, and test it with the remaining 1/5 of experimental dataset (fivefold cross validation), we can fairly decrease FNR.

In addition, we develop a dictionary-based name generation algorithm that combines names obtained from urlbabyname-search.com, and conduct similar experiments (generating 50,000 names, mixing the generated names with 200,000 active account names, and clustering and classifying them). As shown in Table 4, retraining is also effective against the dictionary-based algorithm.

To cope with retraining, criminals have to periodically update their algorithm. However, this can weaken the efficiency of account creation and reduce the free time of undetection. They have to pay additional costs for evasion.

Cybercriminals can develop other advanced name generation algorithms to evade our scheme, but, still, they need to endure additional costs. For example, they can use using non-English names, words from books, and trending keywords when creating malicious accounts. However, all the evasion methods bring non-negligible overhead to them because they have to frequently revise their account creation algorithms or update dictionaries. Such overhead is harmful to large-scale ephemeral malicious account creation, so we are sure that our scheme is meaningful to reduce the overall number of malicious accounts.

Low-rate attack. Another option that cybercriminals can choose is a low-rate attack. Our scheme ignores small clusters (<10 accounts), so they can evade our scheme by creating less than 10 accounts in every 60 min. However, they have difficulties carrying out large-scale attacks.

6.3. Accuracy

Our scheme focuses on early filtering of malicious account groups with minimal information: account names and their creation time. Surely, such scant evidence for judging malicious accounts restricts the accuracy of our scheme. However, we believe that the relatively low accuracy of our scheme is acceptable because our goal is not precise detection of malicious accounts but filtering potentially malicious accounts for further examinations. Moreover, if we can access Twitter's private data, especially the email and IP addresses, we can surely increase the accuracy of our scheme.

Table 4
Evasion versus retraining.

Evasive method	FNR (%)	FPR (%)
Old		
Advanced	86.53	9.33
Dictionary	80.64	3.92
Retrained		
Advanced	8.71	19.29
Dictionary	6.41	12.35

7. Related work

Social spam detection. We classify social spam detection schemes according to the features they focus on: account-based features [2–5], relation-based features [6–8,25], message-based features [9], and URL-based features [10,11]. The account-based features include the number of followers and friends, account age, URL ratios, and text similarities. While collecting these features takes a short time, criminals can easily modify such features. The relation-based features include the social relationships between spam nodes and their neighboring nodes, and between spam senders and receivers. These features are robust against modifications but it takes a long time to collect them. The message-based features imply the suspicious characteristics of each message. They can be used to detect spam messages from compromised accounts but are weak against message obfuscation. The URL-based features imply the suspicious characteristics of each URL or a group of URLs in messages. These schemes need to deal with conditional redirection that returns different landing URLs and content to investigators.

Fake account detection. Cybercriminals manage fake accounts (also known as Sybil accounts) for conducting various attacks. Unlike the ephemeral malicious accounts we focus on, these fake accounts aim at impersonating benign accounts. They create social relationships with each other, conduct adversarial learning to evade detection schemes, and exploit crowdsourcing systems to have help from humans. To detect fake accounts, social-graph-based [26–31], feature-based [32], clickstream-based [33], and crowdsourcing-based approaches [14] have been proposed.

Malicious name detection. Yadav et al.'s method for detecting algorithmically generated malicious domain names [17,18] is related to our study. Their goal is the detection of domain names generated by domain flux botnets. In domain flux, command and control (C&C) servers and bots share the same domain name generation algorithm. They communicate each other using periodically changed domain names to cloak their C&C channels. Their domain names are algorithmically generated so that the names differ from human-made domain names. Yadav et al. utilize the characteristics of domain names to detect malicious domain names. Antonakakis et al. [19] also utilizes the characteristics of domain names of non-existent domain (NXDomain) queries to detect botnets. However, unlike our study, their schemes have more concrete bases to group malicious domain names, such as the shared top level domains (TLDs), IP addresses, and IP-domain bipartite graphs. Therefore, we cannot directly apply their schemes to solve our problem.

8. Conclusion

In Twitter, cybercriminals create a large number of accounts to distribute tweets containing attack URLs for spam, scam, or phishing. Conventional detection schemes can detect malicious accounts only after they send at least one malicious tweet. In this paper, we proposed a novel detection scheme that detects malicious Twitter accounts at the time of their creation without waiting for the initiation of malicious behaviors. Our detection scheme clusters account names that share similar characteristics and classifies the clusters as benign or suspicious on the basis of the characteristics of the name cluster. Evaluation results show that our detection scheme can accurately and efficiently cluster and classify malicious name groups. Therefore, our scheme could be utilized as an early warning system for in-depth monitoring of possibly malicious account groups. In the future, we plan to apply our scheme to other services suffer from massively created malicious accounts such as other online social network services, web forums, and Email services.

References

[1] Twitter Blog, Shutting down spammers, <http://blog.twitter.com/2012/04/shutting-down-spammers.html>.
[2] G. Stringhini, C. Kruegel, G. Vigna, Detecting spammers on social networks, in: Annual Computer Security Applications Conf. (ACSAC), 2010.
[3] K. Lee, J. Caverlee, S. Webb, Uncovering social spammers: social honeypots + machine learning, in: ACM SIGIR Conf., 2010.
[4] A.H. Wang, Don't follow me: spam detection in Twitter, in: Int. Conf. Security and Cryptography (SECRYPT), 2010.
[5] F. Benevenuto, G. Magno, T. Rodrigues, V. Almeida, Detecting spammers on Twitter, in: Collaboration, Electronic messaging, Anti-Abuse and Spam Conf. (CEAS), 2010.
[6] C. Yang, R. Harkreader, G. Gu, Die free or live hard? Empirical evaluation and new design for fighting evolving Twitter spammers, in: Int. Symp. Recent Advances in Intrusion Detection (RAID), 2011.
[7] C. Yang, R. Harkreader, G. Gu, Empirical evaluation and new design for fighting evolving Twitter spammers, IEEE Trans. Inf. Forensics Security 8 (8) (2013) 1280–1293.
[8] C. Yang, R. Harkreader, J. Zhang, S. Shin, G. Gu, Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on Twitter, in: Int. World Wide Web Conf. (WWW), 2012.
[9] H. Gao, Y. Chen, K. Lee, D. Palsetia, A. Choudhary, Towards online spam filtering in social networks, in: Network and Distributed System Security Symp. (NDSS), 2012.
[10] K. Thomas, C. Grier, J. Ma, V. Paxson, D. Song, Design and evaluation of a real-time URL spam filtering service, in: IEEE Symp. Security and Privacy (S&P), 2011.
[11] S. Lee, J. Kim, WarningBird: detecting suspicious URLs in Twitter stream, in: Network and Distributed System Security Symp. (NDSS), 2012.
[12] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, G.M. Voelker, Dirty jobs: the role of freelance labor in web service abuse, in: USENIX Security Symp., 2011.
[13] G. Wang, C. Wilson, X. Zhao, Y. Zhu, M. Mohanlal, H. Zheng, B.Y. Zhao, Serf and turf: crowdturfing for fun and profit, in: Int. World Wide Web Conf. (WWW), 2012.
[14] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, B.Y. Zhao, Social turing tests: crowdsourcing Sybil detection, in: Network and Distributed System Security Symp. (NDSS), 2013.
[15] M.A. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, L. Schmidt, Trends in circumventing web-malware detection, Tech. rep., Google, 2011.
[16] K. Thomas, C. Grier, V. Paxson, D. Song, Suspended accounts in retrospect: an analysis of Twitter spam, in: ACM Internet Measurement Conf. (IMC), 2011.
[17] S. Yadav, A.K.K. Reddy, A.L.N. Reddy, S. Ranjan, Detecting algorithmically generated malicious domain names, in: ACM Internet Measurement Conf. (IMC), 2010.
[18] S. Yadav, A.K.K. Reddy, A.L.N. Reddy, S. Ranjan, Detecting algorithmically generated domain-flux attacks with DNS traffic analysis, IEEE/ACM Trans. Netw. 20 (5) (2012) 1663–1677.
[19] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, D. Dagon, From throw-away traffic to bots: detecting the rise of DGA-based malware, in: USENIX Security Symp., 2012.
[20] Twitter Blog, #numbers, <http://blog.twitter.com/2011/03/numbers.html>.
[21] D. Canali, M. Cova, G. Vigna, C. Kruegel, Prophiler: a fast filter for the large-scale detection of malicious web pages, in: Int. World Wide Web Conf. (WWW), 2011.
[22] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2011.
[23] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Techn. 2 (2011) 27:1–27:27.
[24] B. Zadrozny, J. Langford, N. Abe, Cost-sensitive learning by cost-proportionate example weighting, in: IEEE Int. Conf. Data Mining (ICDM), 2003.
[25] J. Song, S. Lee, J. Kim, Spam filtering in Twitter using sender-receiver relationship, in: Int. Symp. Recent Advances in Intrusion Detection (RAID), 2011.
[26] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, in: USENIX Symp. Networked Systems Design and Implementation (NSDI), 2012.
[27] H. Yu, M. Kaminsky, P.B. Gibbons, A. Flaxman, SybilGuard: defending against Sybil attacks via social networks, in: ACM SIGCOMM Conf., 2006.
[28] H. Yu, P.B. Gibbons, M. Kaminsky, F. Xiao, SybilLimit: a near-optimal social network defense against Sybil attacks, in: IEEE Symp. Security and Privacy (S&P), 2008.
[29] G. Danezis, P. Mittal, SybilInfer: detecting Sybil nodes using social networks, in: Network and Distributed System Security Symp. (NDSS), 2009.
[30] D.N. Tran, B. Min, J. Li, L. Subramanian, Sybil-resilient online content voting, in: USENIX Symp. Networked Systems Design and Implementation (NSDI), 2009.
[31] B. Viswanath, A. Post, K.P. Gummadi, A. Mislove, An analysis of social network-based Sybil defenses, in: ACM SIGCOMM Conf., 2010.
[32] Z. Yang, C. Wilson, X. Wang, T. Gao, B.Y. Zhao, Y. Dai, Uncovering social network Sybils in the wild, in: ACM Internet Measurement Conf. (IMC), 2011.
[33] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, B.Y. Zhao, You are how you click: clickstream analysis for Sybil detection, in: USENIX Security Symp., 2013.