Lightweight Detection of Spamming Botnets

Masaru Takesue
Dept. Applied Informatics, Hosei University, Tokyo 184-8584 Japan
E-mail: takesue@ami.ei.hosei.ac.jp

***Abstract*—A botnet is one of the largest problems against the Internet society because it is used to mount Distributed Denial of Service (DDoS), to steal users credentials, to send spam email, and so on. To cope with the problem, this paper presents a method of detecting spamming botnets, exploiting the information in a hash table of spams produced in our spam filter. To partition the spams based on the similarity of their messages, we cluster the spams in the hash table in three steps by 1) removing the collision (due to the hashing) in each bucket of the table, 2) merging the clusters obtained in step 1), using the fingerprints of message bodies, and 3) further merging the second-step clusters based on the spam-specific words in the *Subject* headers of spams. We identify a bot using the IP address in the first internal *Received* header that is prepended to the list of *Received* headers by the first internal server of a receiving organization. By simulation, we can cluster about 18,000 real-world spams in about 4,000 seconds with no misclustering on our commodity workstation. The active IP space for bots to send spams is almost the same as the one reported in the literature, except of a slight expansion.**

***Keywords*-Spam, bot, botnet, clustering, fingerprint, spam-specific word.**

I. INTRODUCTION

A botnet consists of the home and office computers infected with bot malware and controlled by a botmaster. Botnets are one of the largest problems against the Internet society since they are used to mount serious attacks such as Distributed Denial of Service (DDoS) and users' credentials stealing, and to misuse and waste network resources and recipients' invaluable time by sending spam email (e.g., about 88% to 92% of spams originate from botnets [1]). The size of a botnet ranges from tens of hosts to more than ten thousand hosts [2]; surprisingly, a botnet that Symatec discovered consists of about 400 thousand hosts [3].

Since a large percentage of emails originate from botnets, an analysis of spam email is effective to identify bots and botnets that are spamming [2][4], though there are several approaches to the identification of bots and botnets that are not necessarily spamming (as described in Section II). An email message consists of several kinds of headers and a body. One *Received* header is prepended to the list of *Received* headers every time the email is relayed. Then the list would provide information about the route from the origin to the recipient of the email.

The botmaster usually obfuscates the route information so that it is difficult to identify the origin of spamming bot. The only exception is *the first internal line*, i.e., the *Received* header prepended to the list of *Received* headers by the first internal server of the organization the recipient belongs to [5]. The first internal line gives the IP address of the outside machine that directly delivered the email across the Internet to the receiving organization. The *Received* headers before the first internal line can be falsified by the sender.

This paper presents a method of detecting botnets. For a continuous detection of spamming bots, we integrate botnet detection into spam filtering since currently every organization or user installs a spam filter. Then we can detect spamming botnets worldwide if the botnets detected in the individual spam filters are periodically sent to, for instance, a Botnet Analysis Center to figure out spamming botnets worldwide. We assume our spam filter [6] in the paper.

The botnet detection integrated in spam filtering has two problems: First, we can use only restricted kinds of information for the detection because it is produced in the filter. Second, to identify the member bots of each botnet, we partition the spams classified by the filter into clusters each corresponding to a botnet. Then the clustering has to be lightweight enough for a home or office computer since the clustering of a large number of objects generally needs a very large computing power [7][2].

Our filter detects spam based on the fingerprints of message bodies and spam-specific words in the *Subject* headers of the received email. We start the clustering with a hash table of spams produced in the spam filter, since each bucket of the table is a linked list of spams that have the same hash value. We cluster spams in a hierarchical way to reduce the computation power required in the clustering.

After the clustering, we detect the IP addresses in the first internal lines of the spams in each cluster to identify the botnets and their members. Experimental results show that the clustering of about 18,000 real-world spams can be performed with no misclustering in about 4,000 seconds on our commodity workstation. The active IP space for bots to send spams is almost the same as the one reported in [8], though some new subspaces appear to be emerging. The size distribution of detected botnet significantly differs, depending on the size of partial IP addresses (i.e., the upper 8, 16, 24, and 32 bits) used in the botnet detection.

In the rest of the paper, Section 2 describes related work. Section 3 outlines our spam filter and proposes the scheme for clustering spamming bots. Section 4 describes experimental results, and Section 5 concludes the paper.

 ISBN: 978-1-61208-010-9

II. Related Work

To communicate with the bots in a botnet, its botmaster needs a command and control (C&C) channel. The C&C channels with the Internet Relay Chat (IRC) are currently dominant, though the channels with the HTTP or P2P are appearing to remedy the weak point of centralized structure of IRC channels. We classify the approaches to bot detection into four categories; network-based, C&C-based, DNS-based, and spam-based approaches.

The approach based on network traffic detects bots by observing 1) the session parameters such as a source address/port, a destination address/port, the number of packets and bytes, the start and end time of the session, and the transport layer protocol used [9], 2) the network activity on suspicious IRC and HTTP traffic such as port scanning, spamming, and binary down loading [10], or 3) Windows API socket function calls and their arguments [11].

C&C-based approach exploits 1) behavioral characteristics of IRC bots such that bots are idle most of the time and respond faster than a human upon receiving a command [12] and 2) regularity or invariant characteristics in botnet behavior such as one-to-many connection, simultaneous and immediate responses from bots in a small fixed period, and synchronized malicious actions by bots [13], or 3) flow characteristics of IRC C&C traffic such as bandwidth, duration, and packet size and timing [14].

In a DNS-based approach, the connections to botmasters are redirected to a local sinkhole that performs the 3-way TCP handshake with bots to detect their IP addresses [15]. Another DNS-based technique infers bots by observing lookup behavior into DNS blackhole lists (DNSBLs) [16], since botmasters tend to query these lists to inspect if their bots are blacklisted. Yet another technique detects botnets by identifying the features of a group activity in the DNS queries that a large number of bots in a botnet simultaneously send [17].

The last approach, spam-based one, is most relevant to our approach. Spam campaigns, i.e., coordinated mass emailing of spam, are used as the primary indicator to detect the membership in a single botnet [2]. In identifying the membership, 1) email messages are clustered into spam campaigns, using a number of fingerprints, 2) dynamic IP addresses are inferred into a single one of the same machine, using a probabilistic method, and 3) spam campaigns are merged into a single botnet to cope with the cases where a number of spam campaigns are initiated by the same botnet. These steps run on a cluster of 120 computers since the steps poses formidable computing challenge for a single computer. In [4], URLs embedded in email content are used to detect botnet-based spam emails and botnet membership. Then botnet hosts are identified by regular expressions of URL-based signatures to reduce the false positive rate against polymorphic URLs.

III. Our Method of Detecting Spamming Bots

This section first outlines our spam filter, next presents our strategy for clustering and describes our methods for clustering spams and detecting spamming bots and botnets.

A. Outline of our spam filter

Our spam filter is a cascade of three rule-based filters, each of which produces or collects information required for filtering [6]. Since the filter in a stage of the cascade sends the email it cannot classify to the next stage, the false (positive and negative) rates gradually decrease while the email passes through the cascade.

Let a *token* refer to the hash value of a 50-byte character string produced with the hash function described in [18], and the *fingerprint* (*FP*) of an email be a set of the all tokens for overlapping 50-byte character strings (sliding by one byte at a time) in the email's message body. Then the first filter in our cascade characterizes an incoming email by a set of 32 tokens, called *partial fingerprint* (*pFP*), that have the smallest least-significant 8 bits of the tokens in the FP for the email. The pFPs for earlier spams are stored in the filter and used to classify an incoming email, comparing its pFP with those saved in the filter.

The second filter classifies email, consulting the lists respectively of spam and legitimate email addresses in the *From* headers; this filter also checks the mail address format in the headers. The last filter classifies emails, using the lists respectively of spam-specific and legitimate email-specific words in the *Subject* headers. Evaluated with about 20,000 real world emails, our cascaded filters achieve the false negative rate of 0.025 with no false positive and is lightweight, i.e., classifies about 93 emails per second.

For a detected spam, our spam filter stores an entry consisting of the pFP and other items in a bucket of a hash table, called *eMail Hash Table* (*MHT*); an MHT bucket is a list of entries. The bucket including a specific spam is indexed by yet another fingerprint, the *summary FP* (*sFP*; also referred to as *origin index*), that equals $\sum_{i=0}^{31} \mathrm{token}_i$, where token_i is the i-th token in the spam's pFP.

Let the *dominant spam* for an incoming email stand for a spam whose pFP saved in the filter most matches the email's pFP and the number of matching tokens is not less than a threshold N_{dom}. To decrease the false rates, the filter puts a spam and its origin index into the MHT bucket with the dominant spam's origin index (referred to as *dominant index*) when the spam has the dominant spam, or stores only the spam in the bucket with its origin index otherwise.

B. Strategy for clustering

Let S denote the numbers of spams to be clustered, and T be the average number of tokens per FP. If we would perform spam clustering by FP, the clustering needs $S^2T \log T$ comparisons, where S^2 is for comparing each spam with another one, and $T \log T$ is for comparing each

token in the FP for a spam with all tokens in the FP for another spam, assuming the FPs organized into search trees.

The average size of a message body in our collected $S \simeq 2 \times 10^4$ emails is equal to about 4×10^3 bytes, so that $T \simeq 4 \times 10^3$. Then the total size of FPs equals about 800×10^6 bytes. Assuming one file I/O per 1,000 bytes and the I/O delay (millisecond order per disk seek) of 10^6 times greater that the CPU clock cycle time (nanosecond order), then the clustering of S spams directly by FPs in a single step would need $(S^2 T \log T)/10^3 \simeq 10^9$ seeks. This is equivalent to the time for about 10^{15} CPU clocks, and hence, about 10^6 seconds on a current home/office computer; note that one day equals about 8.6×10^4 seconds.

We reduce the computing power above through these strategy, assuming that in the bots clustering, we can use only the information collected in the spam filter:

1. Perform clustering in a number of steps.
2. Use different metrics in different clustering steps.
3. Produce clusters based on the similarity only of the *leaders* (i.e., the representative spams) of clusters.
4. Verify the clustering correctness in each step by the similarity of FPs.

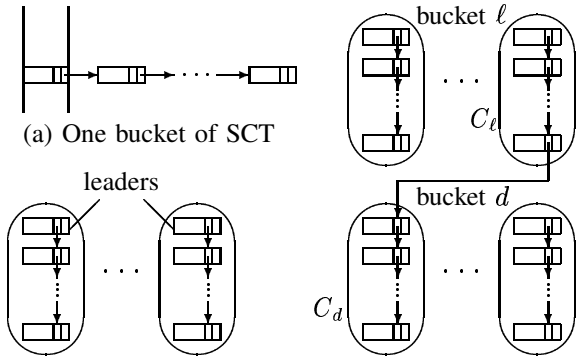
Strategy 1 will reduce the computing power because the clustering is separately applied on each cluster produced in the previous step. Strategies 2 and 3 will decrease the computing power, with a penalty of misclustering. Strategy 4 compensates our clustering method for the probable misclustering due to Strategies 2 and 3.

C. Clustering of spam

For spam clustering, we modify our original filter so that the *Subject* headers and the first internal lines of the detected spams are also saved in the MHT entries. Moreover, we store the spams' fingerprints FPs in a number of files, call *FP files*. In addition, we copy all produced MHT entries into another hash table, called *Spam Cluster Table* (*SCT*), since the original filter deletes stale MHT entries every 1000 spams to save the memory resource.

We start our spam clustering with SCT since each SCT bucket is a preliminary cluster of spams since it is a list of spams with similar message bodies in the sense that they produce the same origin or dominant index. We define a pair of spams as being similar if the rate of the number of identical tokens in their FPs to the number of tokens in one of the FPs is greater than or equal to a threshold, $R_{\rm sim}$. Then our spam clustering proceeds as listed below and shown in Fig. 1; an SCT bucket is shown in Fig. 1.(a) and the *leader* of a cluster is the head spam in the cluster (see Fig. 1.(b)):

1. Cluster the spams in each SCT bucket by (partially) resolving the collisions (i.e., by further hashing the spams) due to the indexing by sFP (Fig. 1.(b)).
2. If a cluster C_d produced in step 1 is located in the bucket with the leader's dominant index d, merge cluster C_d with a cluster C_ℓ in the bucket with the leader's origin index ℓ if the leader is similar to some spam in cluster C_ℓ (see Fig. 1.(c)).
3. Merge the clusters generated in step 2 if their leaders have a number of common spam-specific words greater than or equal to a threshold, $N_{\rm swd}$.



(a) One bucket of SCT

(b) Clusters produced from one bucket; step 1

(c) Merging of clusters; steps 2 and 3

Fig. 1. Clustering by removing aliases and merging based on dominant spams and spam-specific words.

To resolve the collisions in step 1, we put the spams with the same $\mathrm{XOR}_{i=0}^{31}\mathrm{token}_i$ into one cluster (XOR: eXclusive-OR). Then the spams in a obtained cluster have the same $\sum_{i=0}^{31}\mathrm{token}_i$ (= sFP) and the same $\mathrm{XOR}_{i=0}^{31}\mathrm{token}_i$.

A cluster C_d in bucket SCT[d] obtained in step 1 may consist only of the spams that have the same dominant index d but have the individual origin indices different from d (see Section *A*). Let ℓ be the original index of the leader of cluster C_d. Then we merge, in step 2, the cluster C_d with a cluster C_ℓ in bucket SCT[ℓ] if cluster C_ℓ includes at least one spam similar to the leader of cluster C_d (see Fig. 1.(c)).

In step 3, we merge two clusters generated in step 2 when their leaders' *Subject* headers have a number of common spam-specific words not less than a threshold $N_{\rm swd}$ and their leaders are similar to each other. Note that after each clustering in steps 1 to 3, we verify (see Strategy S4) using the FP files whether each spam in a cluster is similar to the cluster's leader. Thus the searching space for the clustering is greatly reduced, preserving the correctness of clustering if the threshold $R_{\rm sim}$ for similarity is appropriate.

D. Detection of spamming botnets

Major techniques for concealing botmaster's identity are to use a member bot of his (or her) botnet as an open proxy or an open mail relay [19]. Although restricting the usage of open email relay (since it is a misconfigured relay that exchanges information with any other computer on the Internet) is recommended in RFC2505 [20], the botmaster

can instruct his bot to start a new open proxy or relay server so that he can send email via the open server. Then the *Received* headers in the received email contain the open server's network address, revealing nothing about the botmaster's identity. Moreover, the email received from the server (i.e., bot) then appears to come from a legitimate home or office user infected by the bot malware.

The first internal line gives the IP address of the outside machine that directly delivered the email across the Internet to the recipient's organization. Assuming that an incoming email originates from a botnet, then the first internal line includes the IP address of a member bot of the botnet, from the discussion above. Thus we identify the IP addresses of bots, using the first internal lines in SCT.

If two sets of IP addresses of the spamming bots in separate clusters have a common IP address, it is probable that the all bots in the clusters belong to the same botnet but the clusters are sending different spams, assuming a single bot malware infection per host. Then we can merge the clusters to obtain a larger size of botnet. As described above, it is not so easy to identify the botmasters' IP addresses due to the proxy and relay servers.

IV. EXPERIMENTS

We experiment using the message sources of real-world emails received by the Mozilla mailer on the Linux system from Oct. 3, 2008 to Dec. 27, 2008 and from May 25, 2009 to Oct 16, 2009; the total number of collected emails reaches about 20k (6k and 14k, respectively). The emails are from U.S.A., E.U., Japan, and other countries and are written in English (about 90%), Japanese (about 5%), or other languages. Of the collected 20k emails, the legitimate email and spam are about 9% and 91%. We experiment on our workstation with dual-processor Xeon (2.66 GHz).

We have the three threshold values, N_{dom}, N_{swd}, and R_{sim}, for the clustering and its verification (see Section III). From the results of preliminary experiments, we set the thresholds thereafter so that $N_{\mathrm{dom}} = 6$, $N_{\mathrm{swd}} = 1$, and $R_{\mathrm{sim}} = 0.4$, since these values have produced largest clusters with no misclustering. To reduce the delay of disk I/O, the FPs of spams are stored in 63 files, in ascending order of email number (attached by person) and are accessed by selecting one of the files by the email number and sequentially searching in the selected file.

A. Spam clustering

To figure out the effects of clustering in each step, we evaluate three clustering methods respectively with only step 1 (denoted by S1), with steps 1 and 2 (S2), and with steps 1 to 3 (S3). The results are shown in Table 1. A smaller number of clusters mean a greater average cluster size. The average cluster size with method S1 is increased with S2, and is further increased with S3. The maximum cluster size is almost the same with the three methods. The execution time required for clustering increases in the order of S1 to S3, but S3 needs much larger time, 170,908 seconds (i.e., about 47 hours) than the other methods do.

To investigate the reason of the large execution time of S3, we experiment S3 with N_{swd} of 3; the results are listed in the parentheses of Table 1. When $N_{\mathrm{swd}} = 3$, the number of clusters slightly increases, and hence, the average size decreases, as compared with those when $N_{\mathrm{swd}} = 1$. However, the execution time greatly decreases to 12,659 seconds (about 3.5 hours). This is because the candidate clusters for merging with a cluster have to be verified before really merged, using the FPs in the files, and a smaller N_{swd} produces a larger number of candidates and hence need a greater verification time.

Table 1. Clustering characteristics and performance; results in the parentheses are obtained when $N_{\mathrm{swd}} = 3$.

	S1	S2	S3
Number of clusters	11,240	8,967	8,530 (8,814)
Average cluster size	1.3	1.6	1.7 (1.6)
Maximum cluster size	225	228	228 (228)
Execution time [sec]	2,251	4,095	170,908 (12,659)

Next, we measure the size distribution of the largest 30 (denoted by top-30) clusters; the results are shown in Fig. 2. The difference of size distributions with S3 and S2 is small, though the execution time with S3 is much greater than the time with S2 (see Table 1). Moreover, in the top-30 clusters produced with S2 and S3, no false positive and negative is found by inspection. Thus we can conclude that S2 is better than S3 in terms of performance/cost; we evaluate with method S2 in the rest of this section.

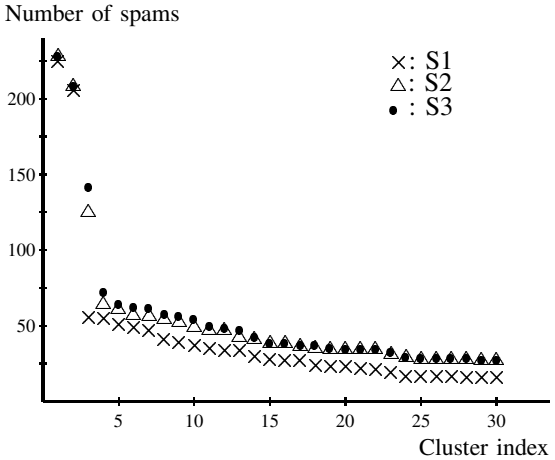


Fig. 2. The distribution of the number of spams in the top-30 clusters.

B. IP space of spamming bots

Next we plot the upper 8 bits of IP addresses (denoted by IPs/8) of spamming bots in the top-20 clusters; the result is shown in Fig. 3, where the dot symbol (·) means that at least one spam in the cluster has the IP/8. The most active IPs/8 space consists of 57.*–97.*, 113.*–126.*, and 188.*–222.*; this result almost completely matches a previous result [8]. Surprisingly, 91% of the spams detected by our spam filter are from the active space. Unfortunately, subspaces 188.*–222.* and 113.*–126.* are expanding downward, and new active subspaces may be emerging around 160.* and below 40.*, as you can see in the figure.

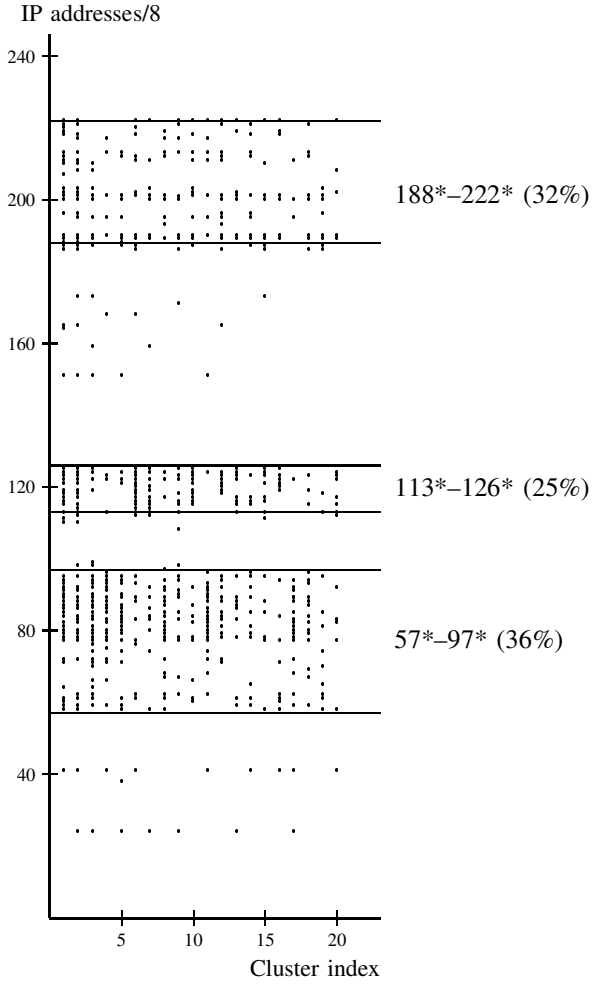


Fig. 3. The upper 8 bit of IP addresses of spamming bots in the top-20 clusters.

C. Botnets and their members

To investigate if the spamming bots in multiple clusters are controlled by the same botmaster, we inspect the IPs/16, IPs/24, and IPs/32 (respectively denoting the upper 16, 24, and 32 bits of IP addresses) that are common to the bots in the top-20 clusters. The result is shown in Fig. 4, where the symbols ∘, △, and × respectively mean that at least one bot in a cluster has the same IP/16, the same IP/24, and the same IP/32 as those at least one bot in another cluster has. For instance, at least one bot in cluster 10 (denoted by C10) has the same IP/16 as that at least one bot in C1 has, and has the same IP/32 as that at least one bot in cluster C2 has. Note that we can consider Fig. 4 as an adjacency matrix for a graph, supposing that each cluster is a node of the graph and symbols ∘, △, and × represent 1's in the matrix.

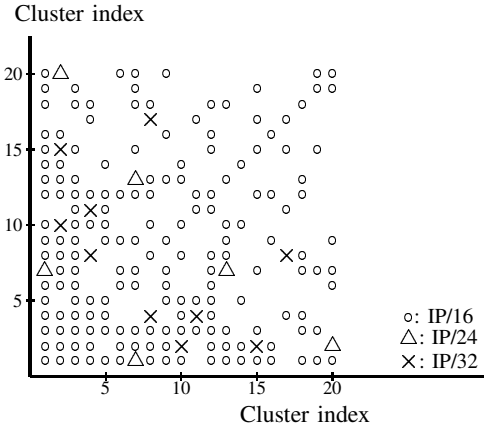


Fig. 4. Partial IP addresses (IPs/n) shared by the top-20 clusters; a symbol at (x,y) means that clusters x and y have at least one common IP/n.

When the bots in some clusters share at least one common IP/16, IP/24, or IP/32 (denoted by cIP/16, cIP/24, or cIP/32), we assume here that the bots in the clusters belong to the same botnet. Then inspecting based on cIP/16, the all bots in the top-20 clusters belong to a single botnet, because a transitive closure of nodes (i.e., clusters) labeled by the ∘ (i.e., IP/16) shown in Fig. 4 means that they belong to a single botnet. Likewise, based on cIP/24 (△), the bots in clusters {C1,C7,C13}, the bots in {C2,C10,C15,C20}, and the bots in {C4,C8,C11,C17} belong to three different individual botnets. Moreover, based on cIP/32 (×), the bots in {C2,C10,C15} and the bots in {C4,C8,C11,C17} are under the control of two botmasters, respectively.

The decision on botnet membership also depends on the dynamics of IP address. It is reported that more that 102 million dynamic IP addresses are identified in a month-long Hotmail user-login trace, and 97% of mail servers setup on dynamic IP addresses send out solely spam emails [21]. Under this situation, we may be able to figure out much larger sizes of botnets than described above. The detection of dynamic IP addresses is, however, out of scope of the paper, and we are reporting a method of botnets detection taking dynamic IP addresses into account in a future paper.

 ISBN: 978-1-61208-010-9

V. CONCLUSIONS

We have presented a lightweight method of detecting spamming bots and botnets, exploiting the information used in our spam filter, which provides us with a hash table of preliminary clusters each of a bucket of detected spams. Beginning with the hash table, we have clustered the spams through three steps, first by partially removing the hash collision in each bucket of the hash table, second by merging the obtained clusters based on the similarity of message bodies of the leaders (i.e., the representative spams in the clusters), and last by further merging the second-step clusters, using spam-specific words in the *Subject* headers of their leaders. A bot was identified by the IP address in the first internal line of the spams' *Received* headers, and a botnet was detected by merging the clusters based on the common (partial or total) IP addresses of their member bots.

By simulation with about 20,000 real-world emails, the clustering without step 3 is desirable in terms of computation cost and clustering potential. Then about 18,000 spams are clustered in about 4,000 seconds. The cluster sizes of the largest 30 clusters distribute from 27 to 228 spams, and the active IPs/8 (i.e., the upper 8 bits of the IP addresses) of the largest 20 clusters consists of 57*–97*, 113*–126*, and 188*–222*; some new subspaces appear to be emerging. The distribution of detected botnet sizes significantly differs, depending on the size (i.e., the upper 8, 16, 24, and 32 bits) of common IP addresses; we have to investigate the effect of dynamic IP address for more accurate sizes of botnets.

Our next work is a botnet detection system distributed worldwide and based on the lightweight detection of spamming bots presented in the paper: Suppose that we can identify spamming bots by spam filters running on home/office computers, and assume that the identifiers and related information for detected bots are sent to a server for bot/botnet analysis. Then we will be able to promptly and accurately figure out botnet structures.

REFERENCES

[1] Messaging Anti-Abuse Working Group (MAAWG), "Email Metrics Program: The Network Operators' Perspective," *Report #12–Third and Fourth Quarter 2009*, MAAWG, 2010.

[2] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, I. Osipkov, G. Hulten, and J. D. Tygar, "Characterizing Botnets from Email Spam Records," *1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[3] L. MaLaughlin, "Bot Software Spreads, Causes New Worries," *IEEE Distributed Systems Online*, Vol. 5, No. 6, 2004.

[4] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming Botnets: Signatures and Characteristics," *ACM SIGCOMM*, 2008.

[5] J. Goodman, "IP Address in Email Clients," *Conf. on Email and Anti-Spam (CEAS)*, 2004.

[6] M. Takesue, "Cascaded Simple Filters for Accurate and Lightweight Email-Spam Detection," *Intl. Conf. on Emerging Security Information, System and Technologies (SECURWARE)*, 2010.

[7] R. Xu and D. Wunsch II, "Survey of Clustering Algorithms," *IEEE Trans. on Neural Networks*, Vol. 16, No. 3, 2005.

[8] M. Kokkodis and M. Faloutsos, "Spamming Botnets: Are we losing the war?," *Conf. on Email and Anti-Spam (CEAS)*, 2009.

[9] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale Botnet Detection and Characterization," *1st Workshop on Hot Topics in Understanding Botnets (HotBot)*, 2007.

[10] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," *15th Annu. Conf. on Network and Distributed System Security (NDSS)*, 2008.

[11] Y. Al-Hammadi and U. Aickelin, "Detecting Botnets Through Log Correlation," *Workshop on Monitoring, Attack Detection and Migration (MonAM)*, 2006.

[12] E. Cooke, F. Jahanian, and D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," *Workshop on the Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.

[13] M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, and S. Yamaguchi, "A Proposal of Metrics for Botnet Detection Based on its Cooperative Behavior," *Int. Symp. on Applications and Internet Workshops (SAINTW)*, 2007.

[14] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting Botnets with Tight Command and Control," *31st IEEE Conf. on Local Computer Networks*, 2006.

[15] D. Dagon, C. Zou, and W. Lee, "Modeling Botnet Propagation Using Time Zones," *13th Network and Distributed System Security Symp. (NDSS)*, 2005.

[16] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing Botnet Membership Using DNSBL Counter-Intelligence," *2nd Conf. on the Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[17] H. Choi, Hanwoo Lee, Heejo Lee, and H. Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic," *7th IEEE Int. Conf. on Computer and Information Technologies*, 2007.

[18] U. Manber, "Finding Similar Files in a Large File System," *Winter USENIX Technical Conf.*, 1994.

[19] D. Boneh, "The Difficulties of Tracing Spam Email," *www.ftc.gov/reports/rewardsys/expertrpt_boneh.pdf*, 2004.

[20] G. Lingberg, "Anti-Spam Recommendations for SMTP MTAs," *RFC 2505*, 1999.

[21] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How Dynamic are IP Addresses?," *ACM SIGCOMM Conf.*, 2007.

 ISBN: 978-1-61208-010-9