

Detecting Abusive Email Senders by SMTP Traffic Monitoring*

Yoshiaki Kasahara^{1 3}, Yoshiaki Hori^{2 3}, and Kouichi Sakurai^{2 3}

¹ Research Institute for Information Technology, Kyushu University,
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
`kasahara@nc.kyushu-u.ac.jp`

² Faculty of Information Science and Electrical Engineering, Kyushu University,
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
`{hori,sakurai}@inf.kyushu-u.ac.jp`

³ Institute of Systems and Information Technologies and Nanotechnologies,
2-1-22 Momochi-hama, Sawara-ku, Fukuoka 814-0001, Japan
`{kasahara,hori,sakurai}@isit.or.jp`
<http://www.isit.or.jp/lab2/index.e.html>

Abstract. Electronic mail (email) is one of the most classical and fundamental services in the Internet. Nowadays email abuse like spam and virus propagation has become serious problems all over the world, and many countermeasures have been devised. In this paper, we focus on finding abusive email senders from the viewpoint of network administrators, and discuss how to detect such a host solely by traffic monitoring and analysis. To avoid privacy issues and reduce processing load, these methods do not process the message body of each email message. In addition, the effectiveness of the methods is examined by using real email traffic collected in our university's campus network.

1 Introduction

Electronic mail (email) is one of the most classical and fundamental services in the Internet. As the Internet becomes popular, email is globally used as a cheap and fast substitute for traditional postal service to exchange messages. Initially, email could only deliver plain ASCII texts. Through a couple of extensions, it has become a more versatile service, which can handle richly formatted texts like HTML and deliver various file types as an attachment.

Versatility of email is also beneficial for its abusers. So-called “spam” (unsolicited commercial email (UCE) or unsolicited bulk email (UBE)) has become a massive burden on the email infrastructure due to a tremendous amount of messages. In addition, email attachment is frequently used to propagate malicious software (malware) such as computer virus. Nowadays email is almost

* This research was partly supported by a contract with the National Institute of Information and Communications Technology, entitled “Research and Development for Widespread High-speed Incident Analysis.”

useless without some kinds of countermeasures against spam and other abusive messages.

As a countermeasure, spam and malware filter is often placed on a receiver side. Filter software marks each email message as legitimate or malicious based on certain criteria, and malicious messages are discarded or placed in a separate mailbox. Several kinds of filtering methods were devised for the purpose[1]. Receiver-side filtering is beneficial to email readers, but it does not reduce the entire amount of email traffic. In addition, confidentiality should be taken into account upon content-based message filtering, because it might violate users' privacy. To reduce network and server loads, abusive email senders should be spotted and shut down.

In this paper, we discuss methods to discover abusive email sender hosts solely by analyzing network traffic between email servers (receivers) and email clients (senders) from the viewpoint of network administrators. The purpose of this work is to avoid inflicting damage to other organization with spam messages delivered from our own organization. We assume that the network administrator cannot directly control end hosts, which actually generate spam, so we try to detect such hosts by traffic analysis at the border gateway between the organization network and the Internet. We mainly focus on Simple Mail Transfer Protocol (SMTP)[2], which is globally used to deliver email messages in the Internet. In addition, we used actual traffic traces to verify the effectiveness of our methods.

2 Background and Motivation

In this section, we explain about the assumed conditions of the target network environment with respect to abusive email client detection, and the basic idea of detecting methods.

Currently, most of abusive email messages are generated by personal computers infected with malwares such as computer virus and bots. For example, Symantec MessageLabs reports that 83.2% of all spam was sent via botnets in June 2009[3]. Infected PCs are controlled by a malicious outsider to execute various illegal activities. When a PC in an organization is infected by malware and starts sending spam, the organization can be considered as both a victim and aggressor at the same time. Repeated occurrences of such incidents may degrade the overall reputation of the organization and its network, and possibly, it causes rejection of (even legitimate) email from the organization by reputation-based mail filtering systems. Therefore, prevention and early detection of such incidents are important.

In this paper, we assume that the target network to observe is a large organization network consists of more than ten thousands end hosts connected. In addition, a network operation center (NOC) maintains the backbone and external connection to the Internet, but leaf networks and end hosts are independently managed by their own departments that use the network. These departments are loosely cooperative with NOC. End hosts are heterogeneous due to different requirements of each department, which make it difficult to manage them

in a top-down manner. We believe that this kind of network configuration is somewhat common in large and traditional universities.

Due to the condition above, NOC cannot directly detect malwares running on end hosts by host-based solutions such as anti-virus software and personal firewall. Therefore, NOC wants a means to detect malwares' activities solely by using network traffic observed over the backbone and border gateways. Usually malware with spam capability has a control channel to receive commands from controller, so it is possible to detect the existence of malware by detecting command and control traffic[4]. Until several years ago, it was easier to detect such a channel because many kinds of malwares employed Internet Relay Chat (IRC) protocol[5] for their command and control, which was a well-defined open protocol. To evade detection, some of recently developed malwares use original protocols designed for the purpose, which is harder to detect without knowledge about it.

On the other hand, malwares are usually designed to do some "real jobs" such as sending commercial email and spamming on a bulletin board, and they have to obey well-known, standard protocols such as SMTP and HTTP during the work. From the viewpoint of administrators, malicious activities from their own network are the most problematic issue and should be avoided. It is a great shame for administrators when an incident is reported from an outside party. We want to catch such kinds of malware activities as soon as possible.

In this paper, we focused on malware activities sending email via SMTP. In addition, to protect users' privacy and reduce processing load, we try not to address the contents (message body) of email. Instead, we focus on behavior of SMTP connection itself and initial information exchange between servers and clients.

3 Related Works

Message filtering at a receiver side is a common measure to reduce the amount of abusive email messages. Usually each message's "spam-likeness" score is calculated based on sender's IP address, domain name, and message body itself. Based on the score, messages are discarded or filtered to a separate mailbox. There are many methods proposed to judge if a message is spam or not[1]. It is effective and beneficial for email users because the filter greatly reduces the noise level of his/her mailbox. A downside is that the filter does not reduce the amount of traffic and server load, because filtered messages have already been delivered to the receiver side.

Theoretically, the filtering methods can be applied at a sender side, but most of malwares contain their own SMTP protocol stack and they do not use standard email infrastructure in an organization network to send spam messages. Therefore, it is harder to filter such messages at the sender side. One solution is to deploy an SMTP transparent proxy to detour all outbound SMTP traffic into a message filter, but such a network configuration is not always possible.

Suppressing malware activities are also important to reduce the total amount of abusive email in the Internet, even if it is not a quick solution. Network based intrusion detection system (IDS) is effective to spot malwares when the attacking traffic is well-known[6][7]. Usually a network based IDS employs simple string match to detect intrusion activities, so it is not very suitable to spot email abuse due to the variety of message patterns.

Some kinds of malwares had been closely examined and the results were used to detect these malwares[8]. In our method, we prefer to observe more generic difference between ordinary email traffic and malware activities, because there are too many kinds of malwares exist and it is not a main purpose for us to discriminate among them.

To detect malwares, spotting traffic of a command and control channel is also possible. As mentioned before, previously IRC protocol was prominent for that[9]. Nowadays advanced botnets start to utilize other protocols such as a peer-to-peer based protocol to control malwares[10], so different methods are needed.

Another approach is to detect anomalies in SMTP traffic based on statistical analysis such as an exponentially weighted moving average (EWMA)[11]. The purpose of the research is to detect deviation from healthy traffic at the server side, so it is not directly applicable to our goal to distinguish malicious hosts from normal hosts.

Of course, these methods described above and our methods are not mutually exclusive with each other. Simultaneous deployment of multiple methods can improve the effectiveness of detection.

4 Detection based on Protocol Design

In this section, we focus on information exchange between an SMTP server (receiver) and a client (sender), and discuss discriminating legitimate clients from malicious clients based on it.

4.1 Observation on Client Names

When a client tries to send an email message to a server, first the client establishes a TCP connection to the server by connecting to TCP port 25 on the server. After the TCP connection has been established, the server sends a “greeting” message. The message must include the status code and the server name, and optionally include the name and version of the SMTP server software. In reply to the server greeting, the client usually send a “HELO” or “EHLO” (extended HELO) command including the client name to identify itself, which is a mandatory step to send a message to the server.

Usually these server and client names are dynamically generated by their hostname, or statically specified in a configuration file of SMTP software. It is (loosely) expected that the name from a single host will not change among different SMTP sessions.

RFC5321 specifies that these names must be a fully qualified domain name (FQDN) of the host, or an IP address enclosed by “[]” when an FQDN for the host is not available. In addition, these FQDN must be resolvable to an IP address by the domain name system[2].

Based on above specification, it is expected that collecting and examining client names for each IP address reveal abnormal clients. These abnormal clients may contain poorly configured clients, but still it is useful for a network administrator to narrow suspicious hosts, which need close inspection out of a large amount of end hosts.

In the ideal condition, the name of a client inside `kyushu-u.ac.jp` domain should have a domain name ended with `kyushu-u.ac.jp`. If it is not, the client can be considered as suspicious. In reality, it is not always the case, because PCs for end users are usually configured poorly in terms of the domain name system, and such a client without an appropriate domain name can send email messages as an SMTP client. In such a case, the client name passed by an HELO/EHLO command can have no meaningful domain name, or even a private IP address. In addition, we know there are several servers in our university that have domain names outside `kyushu-u.ac.jp` for society and community services.

4.2 Real Traffic Examples

In this section, a real SMTP traffic trace collected in the campus network of our university is used to observe the reality of the discussion in Sect. 4.1.

The trace was collected at the gateway of the campus network to the Internet from 00:00 to 23:56 on July 1st, 2008 (the data of last few minutes was lost due to a technical problem). It only includes the beginning of each outbound SMTP session from the campus network to the Internet, including SMTP client names. It does not contain any other data such as email sender/recipient addresses and message body. Table 1 is the basic statistics of the trace data.

Table 1. The number of sessions, IP addresses, and client names in the session trace

Total number of sessions (outbound)	816,848
Unique IP addresses	473
Unique client names	2,465

The entire our university campus network uses a class B or /16 prefix network, which includes about 2^{16} IP addresses. The addresses are divided into /24 prefix networks and assigned to each department in our university. Usually a user’s email message is sent to an internal SMTP server of the network, and the server delivers collected messages to the outside network. Currently about 200 subnets exist in our network, so we expected there should be roughly around

200 SMTP clients sending email to outside. Actually, there are more than twice hosts sending email directly to the Internet as seen in Table 1,

In addition, an anomaly is immediately spotted in Table 1. There are too many (2,465) client names sent via HELO/EHLO commands compared with the number of unique IP addresses (473). We examined how many names were used by each IP address, and discovered one IP address that had sent 2,101 email messages using 1,932 different names. That means that accidentally we succeeded to spot a malicious SMTP client by examine the client names. The client had not been detected by other monitoring systems such as an IDS in our network. We manually inspected the session trace of the malicious client, and found that the SMTP client changed its name on every SMTP connection, probably for camouflage. The names were not completely random, and they included many famous service providers like `netscape.net`, `hotmail.com`, `excite.com`, and so on.

As a result, we verified that a simple validity check of client names could sometimes reveals abnormal clients by an actual case.

On the other hand, not all the malicious clients masquerade its identity by changing their names. A malware running on a host can easily obtain the valid domain name of the network and send email using a valid client name. In such a case, the method is not effective.

Next, Table 2 shows further classification of client names observed in the session trace (except a spammer spotted in the previous discussion). The table shows that only less than half of clients had correctly configured their names in terms of the SMTP protocol specification. That means that it is difficult to detect abnormal clients solely by checking whether the name is a valid FQDN and inside our domain or not.

Table 2. Details of client names observed in the trace (a spammer excluded)

Unique IP addresses	472	
Unique client names	533	100%
Domains inside <code>kyushu-u.ac.jp</code>	237	44.5%
Hostname only (no domain name)	136	25.5%
Raw IP addresses	129	24.2%
Domains outside <code>kyushu-u.ac.jp</code>	23	4.5%
<code>.local</code> domain	8	1.5%

We found some clients using domain names outside `kyushu-u.ac.jp` as expected. We verified the name by manual inspection and concluded that most of them were valid (not malicious) SMTP clients, which host outside services. It is difficult to treat these hosts as valid automatically, so a whitelist is needed to exclude these hosts from suspicious hosts.

There were 35 IP addresses that used multiple client names, and the maximum number of names per a single IP address was 8. Manual inspection revealed that most of them were names without domain, private IP addresses, and a loop-back address (127.0.0.1). We guessed that these IP addresses acted as a NAT (network address and port translation) box which was a gateway to a private IP network with multiple clients. The number of messages sent by these hosts was very small, so we could not verify if the clients were malicious or not.

Currently we are collecting a longer traffic trace to find other methods to detect malicious clients, including analysis of clients' temporal behavior.

4.3 Prototype Implementation

The analysis in the previous sections had been done offline. To detect a malicious activity in timely manner, we implemented prototype software to capture the real-time traffic of SMTP, analyze it, and report a possible abuser.

The prototype was implemented by a combination of a freely available packet capture program “ngrep”⁴ and a scripting language “Ruby”⁵.

In addition to the standard Berkeley Packet Filter logic, ngrep has a unique function to filter captured packet by its payloads using regular expressions similar to a popular Unix command “grep.” In this prototype, the function is used to capture clients' HELO/EHLO in SMTP traffic. For simplicity, the textual output of ngrep is directly fed into an analysis script instead of processing captured raw packets. Figure 1 shows a sample output of ngrep that captures client names. It is a simple task to extract client names corresponding with each IP address from the output.

Every time the script captures a client name of an IP address, it first checks whether the name had already been seen or not. If it is a new name, add a counter for the IP address, and put the name into the known name set. When the counter for an IP address exceeds a certain threshold, it fires an alarm by sending an email with some auxiliary information.

At first all the client names were simply stored into a hash table (a built-in data type of Ruby language) for later look-up, but soon we realized it consumed too much memory when a malware actually started working, because a kind of malware used more than ten thousands of client names per day. To stabilize the memory footprint, we tried a Bloom filter[12] to store the names for the sake of space-efficiency.

A Bloom filter consists of a fixed-length bit array and multiple hash functions. An element is added to the bloom filter by first calculating each hash value of the element to add. The hash values are treated as array positions, and the bits of the bit array are set based on these hash values. To look up if an element is in the filter or not, calculate hash values of the element and check if each position of the bit array is set or not. Only when all the bits are set, the element is considered to be added already. You can consider it as a black box where you

⁴ <http://ngrep.sourceforge.net/>

⁵ <http://www.ruby-lang.org/en/>

```

filter: (ip or ip6) and ( src net 123.*.0.0/16 and dst port 25 )
match: ^(HELO|EHLO)

T 2008/07/01 00:00:00.517711 123.*.*.*:65094 -> 205.*.*.*:25 [AP]
EHLO host1.dom1.example.ac.jp.

T 2008/07/01 00:00:00.611925 123.*.*.*:30156 -> 70.*.*.*:25 [AP]
EHLO host2.dom2.example.ac.jp.

T 2008/07/01 00:00:00.683612 123.*.*.*:42761 -> 82.*.*.*:25 [AP]
EHLO mx.dom3.example.ac.jp.

```

Fig. 1. A sample output of ngrep capturing client names (IP addresses and client names are anonymized)

can add arbitrary number of elements, and query if an element had been already added or not, with a certain false positive rate (it depends on the length of a bit array and the number of hash functions).

4.4 Detected Anomalies by Prototype

The prototype has been running from April 15th, 2009 at the boundary between our campus network and the Internet. Until July 7th, it had captured 45,165,367 SMTP transactions from 1,405 unique IP addresses in our university, and detected 10 suspicious clients. 9 hosts were confirmed to send malicious mails (spam and virus mails) after closer investigation, and one was a false positive. The false positive was caused by a NAT router of our campus-wide wireless access network.

Currently the threshold to fire an alarm is 50 names per IP address based on the observation in Sect. 4.2 since the maximum number of client names used by an IP address was 8. Actually, the value was too high for one of the hosts detected, because it only sent less than 30 mails per day, and more than two days were needed to detect the host. Probably the absolute threshold value is inappropriate because the sensitivity depends on the amount of traffic generated by a host. As a future work, other kinds of statistical values should be used for detection.

We also collected client name samples generated by abusive clients. Figure 2 shows some real examples of client names. A set of strings within a box is generated by a single client. They seemed to be generated by a certain rule or algorithm. Additionally, we noticed that some clients shared the name generation algorithm. Probably these clients were infected by the same (or similar) malwares.

pobox.com rfi.net ibiblio.org uiuc.edu mit.edu cpan.org hagernas.com genome.wi.mit.edu engin.umich.edu	Sdwrf Yuvchcjhe Ccg Myff Ztyrfbnw Pqyjcwt Krbom Atxcc Pgqaxxjmj
CBLMFUGH VFTMOEEBIG TFAHAKA TVCZTBELLY VWNXXVAT HGHILOOAT HJIEBMY LLXKLKPYL UQGAYDZB	xtydhlptxcgkoswbfjnrvaeimquydhlp xjoswbfkoswbfjnrvaeimquydhlptxcg xinrvaeimquydhlptxcgkoswbfjnrvae xwcgkoswbfjnrvaeimrvaeimquydhlpt xxcgkoswbfjnrvaeimquydhlpptxcgko xqvaeimquydhlptxcgkoswbfjnrvaeim xswbfjnrvaeimquydhlptxdhlptxcgko xbgkosxlqvbkgotydirvbgqlquyeimrv xbgkoswbfjnrvaeimquydhlptxcgkosw

Fig. 2. Real examples of collected SMTP client names generated by malwares

5 Detection based on Behavioral Analysis

5.1 Behavior of Malware-induced Mail Clients

In this section, we will discuss about a detection method using client's behavior such as intervals between connections. This time the malicious host detected in Sect. 4.2 is mainly discussed.

The host sent 720 messages from 10:28 to 10:51, 178 messages from 12:54 to 13:00, and 304 messages from 14:01 to 14:07. We do not have traffic traces of other days, but we can guess that the host had been infected by a malware at a certain time, and then started sending spam. Therefore, we should note a sudden appearance of an email sender, which had not been sending email before, or sudden increase of the amount of email from an IP address. A problem is how to distinguish a malicious client from a new mailing-list server that might also suddenly starts sending many email messages.

To find a method to distinguish a malware-induced mail client from a normal client, we analyzed the behavior how the malicious client contacted to other mail servers. The malware program sent many email messages mechanically, which might have different characteristics from an ordinary sender that delivers email messages sent by humans.

Figure 3 shows the cumulative distribution of connection intervals by the malicious client. The x-axis denotes time intervals from 0.01 second to 100 seconds. The y-axis denotes the cumulative distribution of intervals for the corresponding x-axis values. Figure 4 shows the cumulative distribution of connection intervals

by a host which is known to be maintained properly. Unfortunately, in this case we could not find any noticeable differences between the malicious host and a legitimate host.

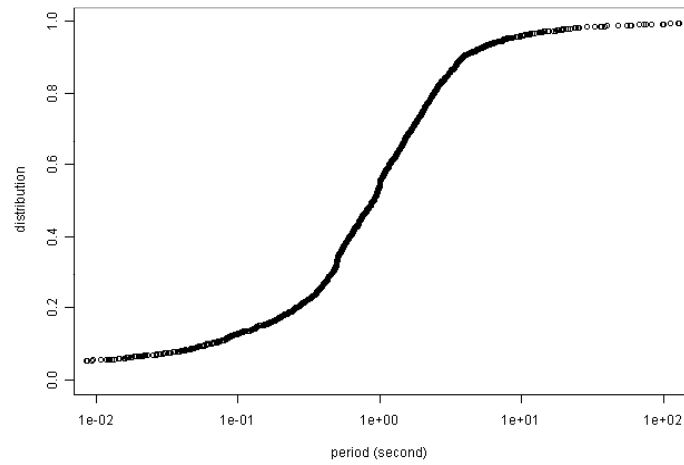


Fig. 3. Sending intervals of HELO/EHLO messages by a suspicious client

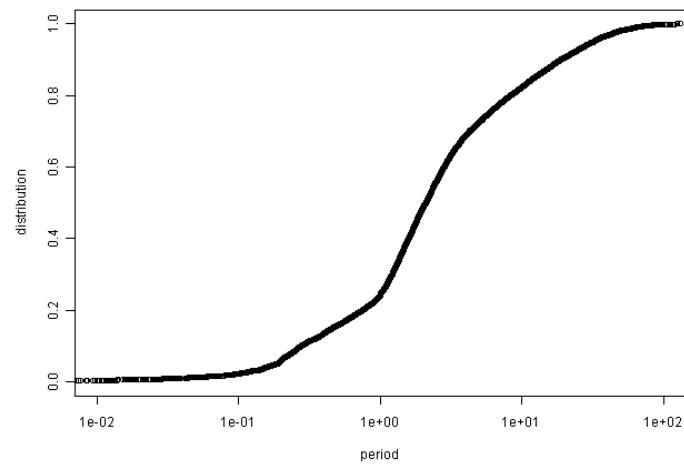


Fig. 4. Sending intervals of HELO/EHLO messages by a legitimate client

A possible reason is that a client tends to connect many servers with different network paths, so even if a client tries to send email in the same interval, intervals between connections varies. In addition, popular mail software such as Sendmail and Postfix has a queue to store email messages before delivery, so it might also affect the interval distribution of a normal client toward more “mechanical.”

5.2 Anomaly Detection for Detecting Malicious Clients

To detect malicious email clients by its abnormal behavior, it is possible to apply general “anomaly detection” methods. In general, there are two directions to achieve the objective.

1. Describe an abuse as a rule like a signature pattern

Traditionally a malware detector such as anti-virus software uses pattern match with a snippet of malware binary as a signature. Carefully selected signatures are effective to detect known malware with low false positives. On the other hand, signatures are not very useful against variants and newly emerged malware because there might be no snippet available. In addition, signature creation is a time-consuming task. Some anti-spam systems employ keyword matching against the message body to detect spam mail, but it is not very efficient because it is easy to evade detection by obfuscating the message text such as intentional misspellings.

2. Describe normal behavior as a rule

Another approach is to provide rules to describe normal behavior and detect anomaly based on the rules. Usually it is not feasible to write such a rule precisely by hand, so we are considering applying machine learning to automatically generate rules from previous traffic patterns, and detect deviations from the rules. An obvious issue of the method is how to determine the “normal” traffic. An anomaly started before the system collecting data might be treated as a “normal” behavior. In addition, this approach usually has a certain ratio of false positives and negatives.

6 Conclusion

In this paper, we discussed about methods to detect abnormal (possibly malicious) email clients by using SMTP traffic monitoring. We analyzed a real traffic trace collected in a campus network to verify the usability and weak points of our methods.

As a future work, we are planning to observe temporal behavior of email clients by using long-term traffic traces. It is a remarkable event when a host that had not sent anything before suddenly starts sending many email messages. The problem is to determine whether the host is malicious or not. We need more information for the judgment. We are also trying to find other characteristics of

SMTP traffic for abuse detection, because clients can manipulate some characteristics (such as HELO/EHLO messages) easily to evade detection.

Real incidents in our university can be used to evaluate and refine the effectiveness of our methods. We are going to implement more detection methods in our prototype to evaluate the performance of the methods and expand the capability of the detection system. It should be also beneficial to users of end hosts as the system can detect activities of malwares on their hosts. Early detection can reduce the duration of active incidents.

In this paper, we mainly discussed abuse of SMTP. Some other communication systems are also actively abused such as the web and instant messengers. To detect more variety of malwares, these protocols should be considered as a candidate of anomaly detection.

References

1. Garcia, F. D., Hoepman, J., Nieuwenhuizen J.: Spam filter analysis. Proc. of the 19th IFIP International Information Security Conference (WCC2004-SEC), pp. 395–410 (2004)
2. Klensin, J.: Simple Mail Transfer Protocol (RFC5321). Internet Engineering Task Force (2008)
3. MessageLabs Intelligence: Q2/June 2009, Symantec MessageLabs. http://www.messagelabs.com/mlireport/MLIRreport_2009.06_June_FINAL.pdf (2009)
4. Cooke, E., Jahanian, F., McPherson, D.: The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. Proc. of 2005 The Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI 2005), pp. 39–44, USENIX (2005)
5. Kalt, C.: Internet Relay Chat: Client Protocol (RFC 2812). Internet Engineering Task Force (2000)
6. Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer Networks, 31(23-24), pp. 2435-2463 (1999)
7. Roesch, M.: Snort – Lightweight Intrusion Detection for Networks. Proc. of the 13th Systems Administrator Conference (LISA' 99), pp. 229–238 (1999)
8. Wong, C., Bielski, S., McCune, J. M., Wang, C.: A Study of Massmailing Worms. Proc. of the second ACM workshop on Rapid Malcode (WORM 2004), pp. 1–10 (2004)
9. Kugisaki, Y., Kasahara, Y., Hori, Y., Sakurai, K.: Bot Detection based on Traffic Analysis. Proc. IEEE The 2007 International Conference on Intelligent Pervasive Computing, pp. 303–306 (2007)
10. Holz, T., Steiner, M., Dahl, F., Biersack, W., Freiling, F.: Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. First USENEX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08) (2008)
11. Luo, H., Fang, B., Yun, X.: Anomaly Detection in SMTP Traffic. The Third International Conference on Information Technology: New Generations (ITNG 2006), pp. 408–413 (2006)
12. Bloom, B. H.: Space/time trade-offs in hash coding with allowable errors. Communication of the ACM, Volume 13, Issue 7, pp. 422–426 (1970)