COAT : Collaborative Outgoing Anti-Spam Technique

Adnan Ahmad and Brian Whitworth
Institute of Information and Mathematical Sciences
Massey University, Auckland, New Zealand
[Aahmad, B.Whitworth]@massey.ac.nz

Abstract. Spam and anti-spam techniques are part of email since its birth. Spam is electronic garbage with no anticipating recipient and almost always deleted. In 2010, around 89% of all emails were spam, resulting in an estimated 260 billion spam emails sent every single day. Most of the current anti-spamming systems focus on incoming spam but these messages still travel the internet world and waste bandwidth, storage and processing resources. This research proposes a collaborative outgoing anti-spam technique to reduce the spread of spam on the internet. The technique targets outgoing emails and its use would free the internet from 260 billion spam a day. During real-time experiment, it blocked 99.95% of the total spam generated with 99.57% elimination at sender side.

Keywords: Anti-Spamming, collaborative, outgoing.

1 Introduction

Simple mail transfer protocol SMTP [1] is the common mechanism for transporting electronic mail among different hosts reliably and efficiently. Most email systems use SMTP to send messages from one server to another, which then retrieved by an email client. SMTP is a relatively simple, text-based protocol, which specified some recipients and then transfers the text to them. SMTP provides no mechanism for accountability and fairness and let sender place messages directly into receiver's inbox. This simplicity of SMTP is exploited by spammers, who take it as a tool for marketing. Spam is electronic garbage that wastes internet bandwidth, storage, and processing [2].

Over the past years, opportunistic sending of static messages through compromised hosts has evolved into dynamically generated, subtly obfuscated messages sent on a massive scale by special purpose malware. Unsolicited email creates problems for internet, clogs mailboxes, slows the servers and lowers the productivity. Although current spam prevention techniques have achieved some success in reducing the amount of inbox spam, the spam sent is still growing from 87.7% in 2009 to 89.1% in 2010 [3]. The arms race between spamming and anti-spamming techniques is ongoing. As filters improve and block some types of spam, spammers develop permutations to defeat the filtering technique. Besides, increasing efficiency at receiver side does not stop an increase in inbox spam sent, resulting in the same (or even more) spam messages in the end users' inbox. Also, current anti-spamming techniques work to prevent malicious mails ending up in the inbox. But it is not sufficient to save inboxes, when spam affects the whole internet. To address this issue, needs more sophisticated techniques to prevent the spam affecting not only the inbox but the whole internet as well.

The rest of the paper is organized as follows: Section II summarizes the current state of the art. Section III describes the method adopted for COAT implementation details. Section IV explains the results, while section V concludes the proposed work.

2 Related Work

Current popular methods for mitigating spam can be categorized into one of the three classes based on their approach, i.e. contents analysis [7], sender reputation [8, 9, 5, 10] and community collaboration [4, 6].

Content based filtering [7] is the most common technique for filtering spam on the basis of contents they contain and can be divided into two types. The first is done by defined rules and used when all classes are static, and their components are easily separated according to some features. The typical example is the rule based expert systems. The second type is done using machine learning techniques and used when the characteristics are not well defined. These techniques attempt to generate on a set of samples, quasi or semi automatically a classifier with an acceptable error rate.

The second class of anti-spamming classifies email based upon who is sending rather than what the contents are. A blacklist [8] maintains the listing of problematic hosts and do not receive emails from them. An email whitelist [9] identifies the people one accepts email from – this includes friends, family, and other contacts. The identification of spam on webmail service reputation is explored in [5], and the transport level characteristics of email flow are investigated in [10] to differentiate spam from legitimate email. Most of the webmail services use rudimentary reputation system.

The collaborative systems [4] do not rely upon semantic analysis but on the community to identify spam messages. Once a message is tagged as spam by one SMTP server, the signature of that message is transmitted to all other SMTP servers. This class requires the collaboration of multiple SMTP servers to implement the system.

The proposed work differs from the other techniques in a way that all of them categorize mail messages at receiver side, whereas COAT works at the sender side and reduces outgoing spam rather than inbox spam. We have hardly found any work in literature about saving the internet bandwidth and resource wastage by spam.

3 Method

This section outlines in detail the algorithm, architecture and constraints with reference to the proposed research.

3.1 Algorithm

When an email is generated and transmitted to the sender's SMTP server for its delivery to the recipient, the sender SMTP initiates the COAT plug-in. The plug-in has the capability to work as a stand-alone unit, so the SMTP syntax needs no change.

When the sender plug-in receives the message, it first checks whether the sender is a legitimate user by consulting the SMTP server database. This step ensures that the true identity of the sender is known and it (along with further steps) helps to eliminate session hijacking done by compromised machines. If the sender exists, the plug-in further checks whether the sender address matches any of the blacklist or standby spammer list, as around 80% of spam messages are send by addresses already in one of the eight black lists [8]. Next, the receiver's address is checked whether he has ever sent email to the sender. A whitelist is maintained for all users and is updated whenever an email is received. The whitelisting present in literature is maintained by users however, the presented technique maintain whitelisting at the SMTP servers without

the involvement of the users. If this module returns true, the email is forwarded to the recipient without any further processing. However, if the sender address is not in the whitelist of the receiver, then the content analyzer analyzes the contents of the email and raise flag in case of spam. The severity of the raised flag decides the generation of challenge for sender and the spam counter value. The sender plug-in algorithm of collaborative outgoing anti-spam technique has been shown in Fig. 1.

```
1:  SenderPlug-in(msg)
2:          Extract sender_address
3:          Auth_agent(sender_address)
4:          Send auth(get_pwd)
5:          IF not chk_sender(sender_add, password)
6:                  Return false
7:          Blacklist_agent(sender_add)
8:          IF   external_module (sender_add)
9:                  Return false
10:         Else IF    local_blacklist (sender_add)
11:                 Return false
12:         Whitelist_agent(sender_add,receiver_add)
13:         IF Search(Receiver_add)
14                  Return true
15:         Content_analyzer (msg)
16:         Chunk_generator (msg_body)
17:         For all body[i] Є msg_body
18:                 Check against n rules (body[i])
19:                 Scoring (body[i])
20:                 Update msg_score
21:                 IF body[i] is not the last msg
22                          Goto 17:
23:         Decide_inc( body_scr)
24:             IF (body_scr<L_min)
25:                     Return true
26:         Bot_defeat_agent (sender_add)
27:         Generate challenge
28:         IF not Send_challange(sender_add, challenge)
29:                 Return false
30:         Spam_counter(sender_add, n);
31:         IF check_counter(sender_add) > 10
32:                 Return false
```

Fig. 1. COAT sender side algorithm

Once the receiver's SMTP server receives the email, it sends a request to collaborative agent for further processing. First, the collaborative agent authenticates the sender SMTP server, and ensures that it has followed the suit of outgoing spam filter. Like multiple other collaborative techniques, COAT requires the corporation of SMTP servers to reduce spam from internet. After consulting the collaborative agent at receiver end, the module delivers the email to the user inbox.

To make the technique error resilient, distributed spam detection module triggers with user feedback. This module takes the user input on a specific spam mail using a user interactive module, which further communicates with the misclassification notifier to send the user feedback to the sender SMTP server. On receiving this feedback, the SMTP server puts the sender on the standby spammer list and closely observes its future activity. The COAT receiver plug-in algorithm is presented in Fig. 2.

```
1:  Rec_Plug-in(msg)
2:      Extract Domain name(sender_add)
3:  Collaborative_agent(domain_name)
4       IF not check_senderdomain(dominname)
5:              Return false
6:  Transfer to user
7:   IF misclassification notifier = true
8:      Extract sender_address(notification)
9:              Pass to Blacklist_agent at sender side
```

Fig. 2. COAT receiver side algorithm

The technique comprises of six components at sender side: i) authentication agent, ii) blacklist agent, iii) whitelist agent, iv) content analyzer, v) challenge generator and vi) spam counter, and two components at receiver side: i) collaborative agent and ii) misclassification notifier. The detailed architectural diagram is shown in Fig. 3.

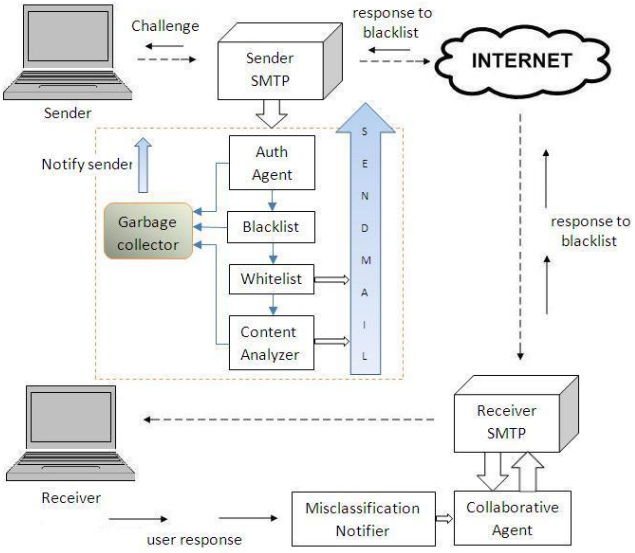

Fig. 3. Detailed architectural diagram of proposed scheme

3.2 Sender Side Components

Detailed description of sender side components is as follows:

Authentication Agent. On receiving the email from SMTP server, the plug-in extracts the sender address and sends it to authentication agent in order to authenticate the sender. This module interacts with the address database and checks whether this is a legitimate user account and have the privilege to send email. The benefit of introducing this module is that no unauthorized user can use the trusted SMTP server to send malicious messages. This module also guarantees that no email can be sent from some spoofed address. Spoofing was the major problem while testing our prototype

implementation, which can be eliminated most of the times by introducing authentication. Furthermore, this module (along with challenge generator) eliminates session hijacking done by compromised machines.

Blacklist Agent. As reported in [8], around 80% of spam originated by the senders already listed in some blacklist. To utilize this, blacklist agent checks the sender address to ensure that the sender does not belong to any of the blacklist. It keeps three types of listing for potential spammers: standby spammer list, local and global blacklists. If some email receives large scoring factor from content analyzer or negative user feedback, the sender is added to the standby spammer list but remains free to send emails. If multiple warnings are received, she is added to the local blacklist and will not be able to send emails for a specific amount of time. Likewise, in twenty four hours, if some sender receives more than twenty five bounce emails due to non-existing addresses, she is added to local blacklist. Addresses added to local or global blacklists do not have the privilege to send emails. After seven days user added here is removed, whereas standby spammer list updates every 48 hours.

The local blacklist, which is used for the reputation of local SMTP server, keeps track of local users and is not shared among other SMTP servers. To reduce the complexity, memory and communicational overhead of global blacklisting, COAT uses existing blacklists to check the reliability of the sender. Incorporation with spam counter at sender side and misclassification notifier at receiver side, this module helps to prevent future spam through new addresses.

Whitelist Agent. This module maintains whitelist across every user. The whitelists found in literature are maintained by users and so increases user overhead as shown in [11, 12]. However, this issue is resolved as COAT maintains one whitelist across every user by keeping an eye on the incoming emails and updates it whenever an email is received.

If the receiver address found in the whitelist, the email is forwarded to her without further processing. This allows the sender to send any number of emails to users from whom she had received any email in the past. Besides, if some receiver sends negative feedback about the sender, the receiver's address is removed from the sender's whitelist by whitelist agent. The idea behind this component is that most of the emails we send are intended to the persons we are already in contact with and had exchanged some emails. If the receiver address is not found in sender's whitelist, the content analyzer is invoked to analyze the contents of the mail body.

Content Analyzer. The content analyzer is responsible for filtering the contents of the email and is invoked only if the receiver does not exist in sender's whitelist. We have used spambayes [13], which is an open source Bayesian based email classification system. The modified working of spambayes is as follows:

Let m denotes the message, b denotes the body of m and x_i denotes the current selected text block such that all $x_i \in b$, then $f1$ be a function that assigns a floating point score SCR to the selected text block x_i in the range 0.0 to 10.0.

$$SCR(b) = \left[\sum_{i-1}^{n} scr(x_i) \rightarrow r, \forall\, x_i \in b \wedge 0 \leq r \leq 10 \right] \ldots (i)$$

Let *f2* be a function that makes a decision *D* on the message *m* to assign it one of four classes: *L_Mn, H_Mn, H_Md* and *H_Mx*.

$$f2: D(m) \rightarrow \{ L\_Mn, H\_Mn, H\_Md, H\_Mx \} \quad \ldots \quad (ii)$$

$$D(m) \rightarrow SCR(b): \begin{pmatrix} 0 \leq SCR(b) < 2, & D(m) \rightarrow L\_Mn \\ 2 \leq SCR(b) < 6, & D(m) \rightarrow H\_Mn \\ 6 \leq SCR(b) < 8, & D(m) \rightarrow H\_Md \\ SCR(b) \geq 8, & D(m) \rightarrow H\_Mx \end{pmatrix} \ldots (iii)$$

If the scoring factor for a particular email increases from the upper bound threshold (*H_Mx*), it is strongly considered as spam and drops at the garbage collector which notifies the sender. If the email has a scoring factor lower than the upper bound threshold but higher than the medium bound threshold (*H_Md*), it still considered as dangerous but approved for transmission. This class further transferred to bot defeat agent and a higher value of spam count is associated with it. If the email has a scoring factor lower than the medium bound threshold but greater than the lower bound threshold (*H_mn*), still this email would be transferred to bot defeat agent along with a relatively lower value of spam count. However, if the email has scoring factor lower than the lower bound threshold (*L_mn*), it would be considered safe and approved for transmission without further processing.

Bot Defeat Agent. About 88.2% of all spam sent in 2010 was generated from bots [6]. The responsibility of this module is to stop the mail bots to generate emails through trusted SMTP servers as well as allow legitimate mails to pass through. If the content analyzer suspects an email of being spam, this module generates a challenge response and sends it to the sender. This module ensures that one must be human to propagate spam and no automatic program can generate spam messages. This module will not affect auto response, as the receiver's address already exists in sender's whitelist, and it will not affect common emails as it only triggers with the high spam contents in the message.

Spam Counter. This module keeps a watch on the number of suspicious messages sent by a sender over specific amount of time. The spam counter checks the content analyzer rating attached with the email and adds a count against sender: three if the rating class is *H_Mx*, two if it is *H_Md* and one if it is *H_Mn*. This count is kept for a couple of hours and during this time interval, if some sender ends up having spam count more than threshold (ten for prototype implementation), she is added to the standby spammer list. Other such implementations bound the user to send extra emails (even legitimate) to her contacts, whereas COAT does not bound any number of legitimate mails.

3.3 Receiver Side Components

On receiving the email, the receiver SMTP server sends a request to collaborative agent to deliver the email to the recipient. Detailed descriptions of receiver side components are as follows:

Collaborative Agent. This component is used to reduce the risk of mischievous SMTP servers for spam propagation. It also stops the infected host machines (spam zombies) which bypass connection level blocking of well-known spamming email relays by using direct client connections to the recipient's mail transfer agent (MTA). This component ensures that the sender SMTP follows the suit of outgoing spam filter. Like multiple other collaborative techniques [4, 5, 6], the presented collaborative agent maintains a list of trustworthy SMTP servers. When an email is received, it extracts the IP and domain key of the sender MTA and delivers the email only if they match the sender SMTP. The updating process for this database is done only by administrator of the SMTP server and no automatic tool is introduced for this task.

Misclassification Notifier. If the user finds some email as spam, she reports it to the misclassification notifier which collects the feedback on all emails that are nominated as spam to verify new spam or false positive. On receiving multiple such responses, this module issues a request to the sender SMTP server about the sender spreading spam. Upon receiving such request, the blacklist agent at sender side adds the sender to the standby spammer list and carefully monitors her future behavior.

4 Results

We tested our implementation in labs with seven SMTP servers for more than three months using a group of 140 students to participate in the email system. Moreover, three professional marketing spammers were assigned to target the system. The statistical outcomes for individual module as well as for the whole system are discussed in this section.

The experiments were divided into four phases. In the first phase, spammers used authenticated addresses on one of seven SMTP servers and launched campaigns through spam bots. As authenticated users, they easily passed the authentication and blacklist but failed at bot defeat agent. Then, they launched more sophisticated spam bots which successfully passed the challenge response. At the end of first phase, 92% of the total spam were blocked. In the second phase, only the intelligent bots of the first phase were used. Initially they were able to transmit some spam to the recipient (after getting higher rating class from content analyzer and higher spam count). However as the blacklists were updated, even these bots were not able to send more spam. 99.82 % of the total spam were eliminated at the end of this phase before transmitting anything over internet.

In the third phase, a couple of self-written SMTP servers were used to test the efficiency of the receiver module. The collaborative agent successfully identified all such emails and dropped them at receiver side. In the final phase, designated marketing

spammers themselves sent spam using legitimate user accounts. They passed authentication, bot defeat agent, and (even some) intelligently written spam was able to get *L_Mn* from content analyzer. These emails were received at collaborative agent and end up in receiver's inbox. The users upon receiving the spam notify the misclassification notifier, which updates the sender SMTP about the address. The spammers were added to the blacklists and further propagation of spam from those addresses was stopped. Even after putting much more effort at generating spam, 99.95% of spam messages were eliminated at the end of all the four experimentation phases.

One of the problems faced during the experiments was the time window for users to respond. At day times, no more than 0.001% of total spam was able to propagate to all targeted users. However, as the users were university students so there were times when not many of them were online due to same geographical location, which is the time when around 0.1% of spam messages end up in user's inbox. But as soon as the downtime overs and users started notifying the misclassification notifier, the spammers could not able to send more spam through the same account. By implementing this technique in diverse geographical locations, there would be more users using their mail boxes at any particular time and the time window for spam would be much smaller.

During all the four phases, 0.43% of whole spam generated (including self-written SMTP) were transmitted on internet and only 0.05% successfully ends up in user's inbox. The reduction of 99.57 % spam releases the internet from resource wastage and the SMTP servers don't need to buy ten times extra resources just to manage spam messages [6]. The efficiency comparison of various anti-spamming techniques is done as well. Efficiency is the percentage of spam messages that are blocked by the system. The schemes considered are content based, blacklist, whitelist, distributed reputational system and COAT. This comparison is illustrated in Fig. 4.

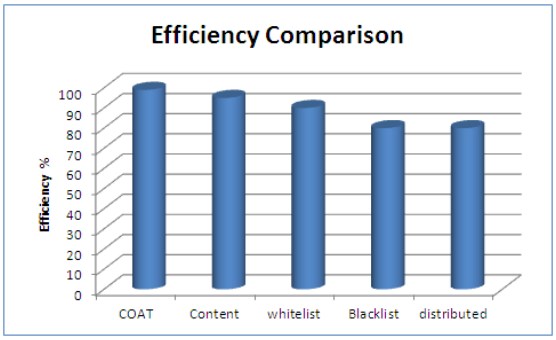

Fig. 4. Efficiency percentage by COAT and existing techniques

The efficiency of COAT is slightly greater than all the other presented techniques but the main advantage of COAT is the saving of resources that are wasted by spam even in presence of other techniques. COAT eliminated around 99.57% of all spam messages at the sender side however, other techniques only categorized them (95%) correctly and saved at separated locations.

Additionally, the participants evaluated the overhead and false positive for each scheme, which are illustrated in Fig. 5 and Fig. 6 respectively. User overhead is the time and complexity at user end to maintain her email account and to report spam to the system. It also includes the care that a user needs to take while writing her email.

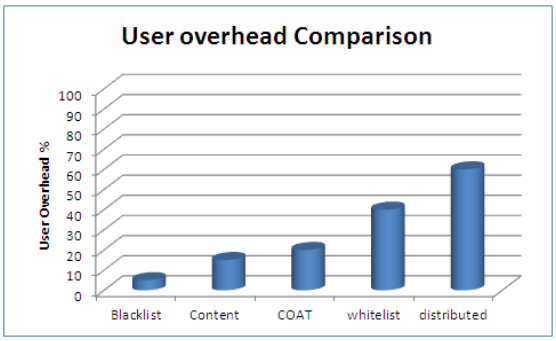

Fig. 5. User overhead percentage by COAT and existing techniques

These results show that COAT is not putting much overhead on the user as compared to whitelist and other distributed schemes. However, the cost of reducing resource wastage is in the form of slightly greater overhead against content based and blacklist. Still the overhead is not greatly different from content based which is one of the most popular anti-spamming techniques.

The false positive comparison of these schemes is illustrated in Fig. 6. False positive in COAT are the legitimate mails against which the senders respond to the bot defeat agent. For other systems, it is the number of legitimate users ending up in blacklist or legitimate emails tagged as spam.

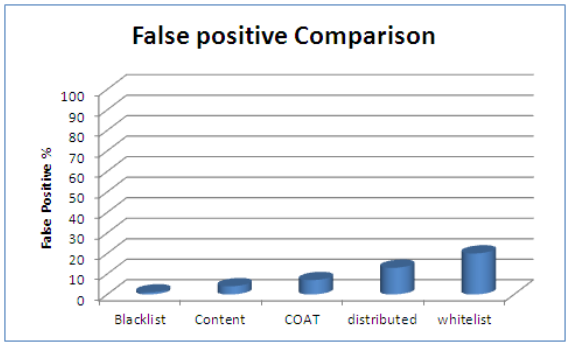

Fig. 6. False positive percentage by COAT and existing techniques

The above graph shows that COAT is not significantly worse than blacklist and content based but indeed seems slightly better than other techniques in reporting false positive. However, its main advantage is the anti-spam effect at sender side not at the inbox, after the spam has travelled the internet and been delivered.

The proposed scheme has blocked around 99.95% of total spam generated, and around 99.57% were eliminated at the sender side and they did not even waste any bandwidth, processing or storage resources, and still this rate can increase by diverse geographical implementation.

5 Conclusion

Everyday billions of spam messages transmit over internet and waste bandwidth, storage and processing resources. These messages have no anticipating recipient and almost always deleted. Anti-spamming inbox defense techniques categorize them into spam or not, but still save them at different locations. If anti-spamming is a categorization problem, the waste of resources remains. This paper highlights this issue and presents a new research direction to stop the spread of spam at sender side to save the internet bandwidth and resources. Like multiple other collaborative techniques, the proposed technique requires the corporation of SMTP servers to reduce spam from internet. We believe if a couple of big email providers incorporate with each other to reduce spam, the whole internet community would be the beneficiary at a great reduced cost.

References

[1] Klensin, J.: RFC 2821: Simple Mail Transfer Protocol. AT&T Laboratories, http://www.ietf.org/rfc/rfc2821.txt April 2001.

[2] Messaging Anti-Abuse Working Group: E-mail Metrics Program: The Network Operators' Perspective. report no. 2, June 2006; www.maawg.org/ about/FINAL_1Q2006_Metrics _Report.pdf. Retrieved 30th July 2010.

[3] MessageLabs intelligence Annual Security Report 2010. www.messagelabs.com/ mlireport/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf. Retrieved 30th July 2011.

[4] Prakash, V. V., O'Donnell, A. J.: Fighting Spam With Reputation Systems. ACM Queue 3(9): 36-41 , 2005.

[5] Taylor, B.: Sender Reputation in a Large Webmail Service. In: 3rd Conference on Email and Anti-Spam, (CEAS) Mountain View, CA, USA, July 2006.

[6] Haskins, R.: The Rise of Reputations in The Fight Against Spam. http://linuxworld.syscon.com/read/48128.htm. Retrieved 30th July 2012.

[7] Khorsi, A.: An Overview of Content-Based Spam Filtering Techniques. Informatica, 31:269-277, 2007.

[8] Dietrich, J. C., Rossow, C.: Empirical Research on IP blacklisting. In: 5th Conference on Email and Antispam, (CEAS) Mountain View, CA, USA, August 2008.

[9] Erickson, D., Casado, M., McKeown, N.: The Effectiveness of Whitelisting: A User-Study. In: 5th Conference on Email and Anti-Spam, (CEAS) Mountain View, CA, USA, August 2008.

[10] Beverly, R., Sollins, K.: Exploiting Transport-Level Characteristics of Spam. In: 5th Conference on Email and Anti-Spam, (CEAS) Mountain View, CA, USA, August 2008.

[11] Garriss, S., Kaminsky, M., Freedman, M. J., Karp, B., Mazieres, D. Yu, H.: Re: Reliable Email. In 3rd Symposium on Networked Systems Design and Implementation, San Jose, California, May 2006.

[12] Golbeck, J., Hendler, J.: Reputation Network Analysis for Email Filtering. In Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA, July 2004.

[13] Meyer, T. A., Whateley. B.: Spambayes: Effective Open-Source, Bayesian Based, Email Classification System. In: Conference on Email and Anti-Spam (CEAS), Mountain View, California, USA, July 2004.