PROTECTION AGAINST SPAM USING PRE-CHALLENGES

Rodrigo Roman[1], Jianying Zhou[1], and Javier Lopez[2]
[1]*Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613;*
[2]*E.T.S. Ingenieria Informatica, University of Malaga, 29071, Malaga, Spain.*

Abstract: *Spam* turns out to be an increasingly serious problem to email users. A number of anti-spam schemes have been proposed and deployed, but the problem has yet been well addressed. One of those schemes is challenge-response, in which a challenge is imposed on an email sender. However, such a scheme introduces new problems for the users, e.g., delay of service and denial of service attacks. In this paper, we introduce a *pre-challenge* scheme that avoids those problems. It assumes each user has a challenge that is defined by the user himself/herself and associated with his/her email address, in such a way that an email sender can simultaneously retrieve a new receiver's email address and challenge before sending an email in the first contact. Some new mechanisms are employed to reach a good balance between security against spam and convenience to email users.

Keywords: electronic mail; anti-spam; internet security.

1. INTRODUCTION

Email is one of the most valuable tools for Internet users, with which people at anywhere can communicate instantaneously regardless of the distance. However, this tool can be used for bad purposes too, and there is no doubt that the worst use of email is spam.

Spam, or unsolicited commercial email, can be defined as advertising messages (mostly for fraudulent products) neither expected nor desired by the intended receivers. Since it is very easy to flood users' mailboxes with little investment, spam is a big threat to email systems, resulting in the loss of time and money to email users.

A lot of research in the area of anti-spamming has been done in the past years, trying to seek effective solutions to the spam problem. One of those solutions is *challenge-response*. When a sender sends an email, he/she is first given a challenge from the receiver that must be solved before the email reaches the receiver's mailbox. However, such a scheme introduces new problems, for example, *delay of service* (when a sender waits for arrival of a challenge from a receiver), and *denial of service* (when challenges are redirected to a victim's address that is spoofed by spammers as the sender).

Our Contribution. In this paper, we propose a *pre-challenge* scheme, which is based on challenge-response mechanism, preserving its benefits while avoiding its drawbacks. It assumes that each user has a particular challenge associated with his/her email address, in such a way that an email sender can simultaneously retrieve a new receiver's email address and challenge before sending an email in the first contact. Our scheme enables management of mailing list and error messages. Our scheme is easy to be integrated into existing email systems as it is a standalone solution, without changing the other party's software and configuration.

The rest of the paper is organized as follows. In section 2, we summarize the existing solutions against spam and analyze their limitations and/or problems. After that, we present a new solution in section 3, and further discuss it in section 4. Finally, we conclude the paper in section 5.

2. PREVIOUS WORK

The original SMTP protocol [1] was introduced in 1982, with only minor modifications [2] in the past 20 years. The main problem in SMTP is the lack of authentication. When an email is received, it is not possible to know whether the source of the email is who claims to be. This is precisely the flaw that spammers make use of. However, as the SMTP protocol has been standardized and widely deployed, most of the research focuses on avoiding spam while maintaining the actual SMTP protocol and email infrastructure in order to ensure compatibility. This implies that anti-spamming solutions must be based on the operation with email headers and contents or on specific implementations at the application level.

One of the headers that can provide information regarding an eventual spam of the incoming email is the "*Received:*" headers, which give information about the client MTA. There are some projects [3] that try to identify misconfigured email MTAs or major sources of spam. However, it does not work effectively against individual spammers, and innocent client MTAs might be blocked.

Another header that can be used against spamming is the receiver's address, with policy or password-like extensions. In policy-based systems [4], policies are encoded inside the address and an email is discarded at its destination if the policy is not fulfilled. In password-based systems [5–7], the receiver's address is extended with a sequence of characters that act like a password, which can be obtained with a proof of computational task [10]. These solutions work well in some scenarios (e.g., using mail addresses in computer-based systems like web forums). However, as the email addresses created in such schemes are very hard to remember, they may cause problems when used by humans.

There are several works dealing with email content analysis based on artificial intelligence (AI) and statistical techniques [8,9]. They try to distinguish whether an email comes from a legitimate user or from a spammer by assigning a "spam score" to any incoming message. This approach may lead to false positives, and spammers may try to bypass the classifier algorithms.

Other implementation approaches against spam include *micropayment*, *challenge-response*, and *obfuscation* schemes. Micropayment schemes [10-13] are applied to email systems in order to prevent spammers sending millions of emails. They require the user or client MTA to compute a moderately hard function in order to gain access to the server MTA. As a result, a spammer will not be able to send a large number of emails to a certain server MTA. Such an approach is difficult to be applied to those client devices with very weak computing capability (e.g., mobile phones).

In challenge-response schemes [14,15], whenever an email from an unknown user is received, a challenge is sent back to that user. The solution to that challenge can be simple (e.g., just reply), complicated (e.g., solve a CAPTCHA [17]), or time consuming. Only when the correct response is received, the emails from that user are allowed to enter into the receiver's mailbox. These schemes do not work when a human user is not involved in sending emails (e.g., in the case of mailing lists). Moreover, these schemes may introduce new problems such as delay of service and denial of service.

In the obfuscation scheme, email addresses are displayed in an obfuscated format (e.g., John HIPHEN Smith AT yahoo DOT com), from which senders can reconstruct the real email addresses. It does not require any software from the user side or from the server side. However, the problem with this scheme is the constraints that the human users face when constructing the obfuscated addresses. As the combinations are limited, it allows AI-based harvest programs to easily retrieve real addresses. Moreover, once the email is captured by the spammer, there is no protection against spam (unless other solutions are utilized).

3. A PRE-CHALLENGE SCHEME

3.1 Overview

As stated, our pre-challenge scheme is based on challenge-response mechanism in the sense that both of them impose a challenge that must be solved by a potential sender. However, in the pre-challenge scheme, the sender retrieves the receiver's email address together with his/her challenge simultaneously (see Fig. 1). Once the challenge is solved, the answer will be included inside the email.

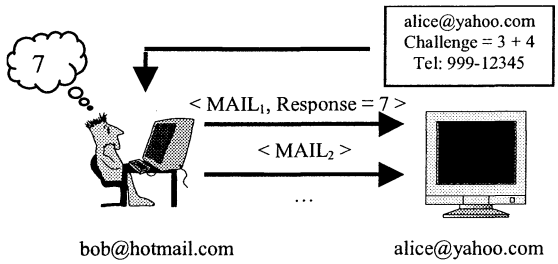

Figure 1. Basics of the pre-challenge scheme

When a mail from an unknown sender arrives, the receiver's system tests whether that mail contains an answer to the challenge. If the test turns out positive, the sender is *white-listed.* That means future mails from this sender will get into the receiver's mailbox without being checked again.

The goal of our scheme is to check whether there is really a human behind a sender's computer. The reason is that spammers use automatic programs to send their propaganda, and they feed these systems with email addresses obtained by searching web sites and mail servers. However, it is a bit hard for these programs to retrieve a challenge that matches an email address and even harder to answer each of these challenges. Therefore, whenever a spam arrives to destination, it will be automatically discarded if no correct answer to the challenge is attached.

In comparison with a challenge-response scheme, our pre-challenge scheme preserves its benefits while avoiding its drawbacks, as we explain in the following:

- In a challenge-response scheme, there is a delay in obtaining the receiver's challenge. On the contrary, in our pre-challenge scheme,

because the receiver's challenge is available in advance, the sender can directly solve the challenge and send the email to the receiver [13].

- With a challenge-response scheme, if spammers forge a sender's address in their mails, the challenges will be sent to that address, launching a possible DDoS attack [16]. This attack will not take place in the pre-challenge scheme because a receiver need not reply an unknown sender's request for a challenge.
- A challenge-response scheme can work with mailing lists only if some rules are manually introduced, and it cannot handle mail error messages properly. As we will show in section 4.1 and section 4.4, the pre-challenge scheme manages mailing list systems and processes mail error messages without any problem.

Another benefit of the pre-challenge scheme is the continuous protection against email harvesting. When a correct email address is retrieved by a spammer, he/she needs to get the solution of the current pre-challenge at the same time to make the address usable, but the user can change the pre-challenge at any time (see section 3.2), making the combination <email,solution> useless.

3.2 Challenge Retrieval and Update

A challenge is defined by an individual human user. Each user has one challenge at a time to be used by all incoming emails, and the challenge can be updated at any time at his/her own discretion. The challenge can range from a simple question or mathematical operation to a hard-AI problem that only a human can solve [17].

Normally a user's challenge is published next to this user's email address. Since any potential sender must retrieve the email address of the receiver before contacting him/her, challenge and email address can be accessed at the same time. However, in certain cases, a challenge may not be accessed directly. Instead, a URL may be provided to retrieve the challenge.

Since the challenge is not restrained to obfuscate a valid email address, which has a fixed structure (name, domain), the user has more freedom to produce it. When stored inside a website, the challenge can take advantage of its form and content – personal information, the theme, or visual appearance of the website, etc. Challenges may also be retrieved using a majordomo style service [22]. To prevent spammers from using this service as

[13] The frequency of challenge update is a security parameter decided by the receiver, based on his/her own experience, to control the risk of replay attacks from spammers.

a collector of valid email addresses, the service must return a false challenge for every non-existent user.

3.3 Data Structures

The pre-challenge scheme requires certain data structures to accomplish its tasks. The two most important structures are the actual challenge (or a URL), and the solution to the challenge. By using these structures, it is possible to advertise the actual challenge and to check whether an incoming mail has solved the challenge. Additionally, the solutions to old challenges must be stored, as discussed later.

Other data structures needed by the scheme are the *white-list* and the *reply-list* (both used by some challenge-response schemes), and the *warning-list*, that is a structure specifically created for our new scheme. Each of those structures contains a list of email addresses and, optionally, a timestamp which indicates the time an email can be in the list.

White-List. The *white-list* contains email addresses in such a way that emails coming from those addresses are accepted without being checked. Some email senders may even be *white-listed* by a receiver at the set-up phase if they are already known. Those senders are marked in order to send a confirmation when receiving their first message (see section 3.5). This list could be manually modified by a human user.

Reply-List. The *reply-list* contains email addresses of those users to which the local user has sent email to, and has not replied yet. The use of this list is justified because the local user is the one who initiated the communication with those users; hence, there is no need to check any challenge when replies are received. This list will be managed automatically by the local user's system.

Warning-List. The *warning-list* contains email addresses of users that have sent an email containing the answer of an old challenge. The existence of this list is justified because an email message with an old response will cause a reply from the receiver indicating the new challenge. With this list, the local user does not need not send that reply more than once. This list will be reset every time when the challenge is updated, and will be managed automatically by the local user's system.

3.4 Security Levels

The pre-challenge scheme can be configured to work at two security levels, *high security* and *low security*. The main difference between these two levels is how the *reply-list* is queried.

The scheme starts working at the high security level of protection. High security means that all queries in the *reply-list* are done by looking for a <user, domain> match, and the matched entry will be erased from the *reply-list*. On the other hand, low security means that all queries in the *reply-list* are done by looking for a <*, domain> match.

The reason why the pre-challenge scheme needs these two levels of security is that some email accounts have different addresses for receiving and for sending email. This usually happens with mailing lists, and this issue will be discussed in section 4.1.

3.5 Architecture

Now we explain the design of our pre-challenge scheme. Suppose user *B* wants to send an email to user *A*. To simplify the explanation, we assume that user *A* is using the pre-challenge scheme while user *B* is not.

1. *A*'s system checks if *B*'s address is listed in the *white-list*. If this is the case, the email reaches *A*'s mailbox. Additionally, if that mail is the first message *A* received from *B*, *A* sends a confirmation email to *B*.
2. Otherwise, if *B* is listed in the *reply-list*, the email reaches *A*'s mailbox and *B* is added to the *white-list*. We should point out that the query to the *reply-list* is different according to the level of security being applied, as seen in section 3.4. In case of using a high security level, *B* is erased from the *reply-list* because *A* received the reply expected from *B*.
3. Otherwise, *A*'s system checks whether the challenge of the email has been solved. If it is solved, the mail reaches *A*'s mailbox and *B* is added to the *white-list*. Additionally, *B* receives a confirmation email.
4. Otherwise, if the email has a solution to an old challenge, *A*'s system checks if *B* is listed in the *warning-list*[14]. If that is the case, the mail is discarded. If it is not listed, *B*'s address is added to the *warning-list* and *B* gets a reply containing information about the new challenge.

[14] Note, the *warning-list* will be reset whenever the challenge is updated.

5. Otherwise, the email is discarded without any reply to *B* indicating this fact. The problem of accidental discard of a legitimate email will be addressed in section 4.3.

It should be noted, however, that discarding the email does not mean the user cannot read it. The scheme can be configured for labeling the message with a "spam score" and placing it in a special fold of the mailbox if the owner of that mailbox desires so.

3.6 Spam Scenarios

When a spammer wants to send his/her advertisements to a final user that operates the pre-challenge scheme, he/she basically faces two scenarios.

Scenario 1. The spammer only retrieves the email address of a target, but not his/her challenge. When the spam is sent to the target, it will be silently discarded because no solution to a (current or old) challenge is included.

Scenario 2. The spammer only retrieves the email address of a target, and impersonates as a sender that happens to be in the receiver's *white-list*, due to the lack of authentication in the email infrastructure. All schemes that use a *white-list* share this problem, but this is not a serious issue because spammers must find the white-listed senders for all the addresses he/she want to spam. And for millions of addresses to spam, this is unprofitable.

It could seem that a spammer, using little investment (solving one challenge), can send many pieces of spam to a given email address (a replay attack). It could also seem that a group of spammers interchange their solved challenges of the corresponding users in order to lessen each spammer's effort on accessing the victims' mailboxes. However, what spammers want is to send millions of messages. And since the challenges are different for every user and a challenge can be solved only by a human, the task of repeatedly solving or sniffing a new challenge per user, or hiring cheap labor in order to send spam, becomes unprofitable.

4. FURTHER DISCUSSION

Here we further discuss how our scheme works for users in a mailing list, and whether our scheme can make a challenge easily available to users and make users to be sure on the delivery status of an email. We also discuss how to manage mail error messages.

4.1 Mailing Lists

Mailing lists [18,19] share a common behaviour: upon registration, they send a challenge to the user in order to prove that the user is a real person. As a result, it seems not possible to use challenge-response schemes with mailing lists.

Fortunately, there is a solution to this problem in the pre-challenge scheme. Since all mails from the same mailing list come from the same domain, a user can switch to the low security level (see section 3.4) whenever he/she wants to subscribe to a mailing list. At the low security level, all the incoming mails from the mailing list domain (including all the challenges and all the messages from the mailing list) that have a match in the *reply-list* are accepted into the user's mailbox and their senders are *white-listed.* When the user finally receives the first mail of the mailing list, he/she switches to the high security level (see Fig. 2).

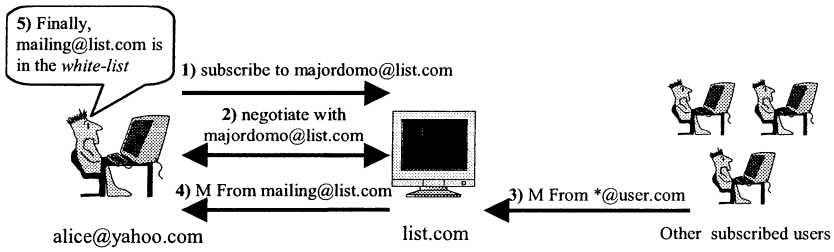

Figure 2. Process of subscription to a mailing list

The risk of inserting a spammer inside the user's *white-list* while the user is at the low security level is very low, because the spammer's email address must have the same domain as the people in the user's *reply-list*, and because a user normally only subscribes to a few mailing lists in a year.

Also, the user can set up the system not for adding the incoming mails to the *white-list* when running at the low security level, but for adding to a temporary *white-list* instead. He/She will decide later whether to add (manually) them into the final *white-list.*

4.2 Availability

It is clear that some availability problems exist when the challenge is not published along with the email address. If a sender cannot obtain the

challenge of a new receiver and solve it, his/her email may not be able to reach the receiver's mailbox.

It might be good to provide both the challenge and a URL that points to the challenge for better availability. In case the URL does not work, the challenge (even if outdated) can still be used by an email sender to get in touch with a new receiver. (The receiver will reply with the latest challenge on receiving the answer of an old challenge.)

Finally, there is an availability problem that is common for both pre-challenge and challenge-response schemes: A challenge easy for a normal user might be impossible to solve for a disabled user. For example, a blind user will find impossible to solve a challenge based on images without help.

4.3 Accessibility

One of the main issues in the pre-challenge scheme is that an incoming email from a new sender without the answer of the receiver's challenge is automatically discarded, and the sender is not notified. This approach avoids the increment of Internet traffic due to the responses to spammers' mails, but also introduces a problem: a normal sender is not sure whether the receiver really got the email.

A possible solution is to define a standard prefix in each email address that is enabled with the pre-challenge scheme. In such a way, the sender knows clearly that a challenge should be answered in his/her first email to such a receiver and a notification is expected should the email reach the receiver's mailbox.

There is an alternative solution if the pre-challenge scheme is implemented at the MTA level. In this solution, the sender is warned of the invalid answer of challenge using the error reporting mechanism of the SMTP delivery negotiation protocol. This protocol works as follows:

1. The client MTA sends the contents of the email to the server MTA. After that, the server MTA checks if the email must be accepted or rejected by searching the answer to the pre-challenge.
2. If the negotiation fails, the client MTA creates an email that includes the cause of the error and the undelivered email. That email is sent to the original sender, if the client MTA does not manage his/her emails.

By using this solution, the final user will receive an error message if he/she sends an email with an invalid answer of a challenge, without increasing the Internet bandwidth in most cases. We have more discussions on managing error messages in section 4.4.

4.4 Managing Mail Error Messages

During the SMTP delivery negotiation between two MTAs, if an email cannot be delivered to its recipient, the client MTA has to send the original sender an email containing an error message. Errors can range from an invalid recipient to over-quota mailboxes, or (as seen in the previous section) pre-challenge errors.

A problem arises when the error message is not created by the MTA of the client that implements the pre-challenge scheme. An example is shown in Fig. 3. In the example, the error happens at MTA lvl 2, thus MTA lvl 1 creates and sends an error message back to the original sender. But MTA is a computer and will not include any answer of a challenge inside the error message. Therefore, it will not reach the client's protected mailbox – a problem of availability.

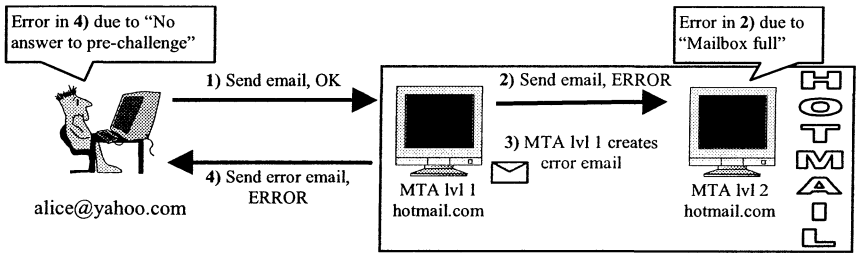

Figure 3. Problems while dealing with error messages

This problem can be solved based on two premises. First, error messages can be identified with the "message/delivery-status" header, and have attached the email that caused the problem. Second, all emails have a unique ID issued by the original client MTA, stored in the "Message-ID" header.

When an error message arrives, the pre-challenge scheme accepts the email if both address of the recipient and ID of the original message are inside the *reply-list*. Thus, it is necessary to add the ID of outgoing emails to the *reply-list*.

A spammer may try to bypass this scheme by forging both the unique ID and the recipient of the original message. This requires the spammer to wiretap the communication channel, which is unprofitable for massive spamming.

5. SUMMARY

In this paper, we presented a pre-challenge scheme for spam controlling, based on challenge-response mechanism but avoiding its drawbacks. Our scheme is a standalone solution, since there is no need to install software or change the configuration in the sender's side. Our scheme allows email senders to have no delay in reaching the receiver's mailbox, and prevents the denial of service attack if the origin of the email is forged. It also manages mailing list messages and error messages properly. Finally, our scheme offers protection against email harvesting.

This scheme can be used jointly with other major anti-spam solutions, because the type of protection that the pre-challenge scheme provides is centered in the protection of email against harvesting, thus leaving the door open to other solutions such as content analysis. Moreover, the scheme could also be integrated with authentication solutions like DomainKeys [20] or Identity-Based Encryption [21], hence thwarting attacks like using forged senders to bypass the *white-list* checking.

REFERENCES

1. J. Postel. *Simple Mail Transfer Protocol.* RFC 821, IETF, August 1982.
2. J. Klensin. *Simple Mail Transfer Protocol.* RFC 2821, IETF, April 2001.
3. SBL. http://spamhaus.org/.
4. J. Ioannidis. *Fighting Spam by Encapsulating Policy in Email Addresses.* NDSS'03, February 2003.
5. E. Gabber, M. Jakobsson, Y. Matias, and A. Mayer. *Curbing Junk E-Mail via Secure Classification.* 1998 Financial Cryptography, pages 198-213, February 1998.
6. R. J. Hall. *How to Avoid Unwanted Email.* Communications of the ACM, 41(3):88-95, March 1998.
7. L. F. Cranor and B. A. LaMacchia. *Spam!* Communications of the ACM, 41(8):74--83, August 1998.
8. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. *A Bayesian Approach to Filtering Junk Email.* AAAI'98 Workshop on Learning for Text Categorization, July 1998.
9. P. Cunningham, N. Nowlan, S. J. Delany, and M. Haahr. *A Case-Based Approach to Spam Filtering that Can Track Concept Drift.* ICCBR'03 Workshop on Long-Lived CBR Systems, June 2003.
10. C. Dwork and M. Naor. *Pricing via Processing or Combatting Junk Mail.* Crypto'92, pages 139-147, August 1992.
11. C. Dwork, A. Goldberg, and M. Naor. *On Memory-Bound Functions for Fighting Spam.* Crypto'03, pages 426-444, August 2003.
12. M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber. *Bankable Postage for Network Services.* 8th Asian Computing Science Conference, December 2003.
13. Microsoft Penny Black Project. http://research.microsoft.com/research/sv/PennyBlack/.
14. SpamArrest. http://spamarrest.com/.
15. SpamCap. http://www.toyz.org/cgi-bin/wiki.cgi?SpamCap.

16. J. Mirkovic, J. Martin, and P. Reiher. *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms*. Technical Report #020018, Dept. of Computer Science. Univ. of California.
17. L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. *CAPTCHA: Using Hard AI Problems for Security*. Eurocrypt'03, pages 294-311, May 2003.
18. Ezmlm Mailing List. http://www.ezmlm.org/.
19. Mailman Mailing List. http://www.list.org/.
20. Yahoo DomainKeys. http://antispam.yahoo.com/domainkeys/.
21. D. Boneh and M. Franklin. *Identity Based Encryption from the Weil Pairing*. Crypto'01, pages 213-229, August 2001.
22. Majordomo Mailing List. http://www.greatcircle.com/majordomo/.