P. Hallam-Baker
VeriSign Inc.

Internet Draft
Document: draft-hallambaker-consent-00.txt
Expires: October 2005

July 2004

Proof of Consent Mechanism

Status of this Memo

Abstract

We propose a mechanism Proof of Consent that allows an email recipient to provide verifiable proof of æopt-inÆ consent to receive email. Proof of Consent may be used to automatically whitelist email from mailing lists and email forwarded from another email server.

Proof of Consent is designed to require minimal state maintenance by both the email sender and the recipient and to be deployable with minimal impact on existing email infrastructure.

1. Requirements

The Proof of Consent mechanism provides a verifiable and revocable proof that a recipient consented to receive email from a specified source. It is designed to address several of the existing problems of maintaining mailing lists and other consent based bulk mail such as newsletters and solicited advertisements.

We note that other protocols such as Really Simple

Syndication (RSS) and Network News Transport Protocol (NNTP) offer facilities that are similar to mailing lists without the need for a proof of consent mechanism. Despite the existence of these mechanisms email remains the most commonly used means of obtaining data of this type and it is appropriate to address these requirements in the context of email mailing lists despite the existence of alternative protocols that already meet them.

1.1 Subscription Consent Problem

The SMTP protocol does not provide a means to allow a mail server receiving a message alleged to have been sent to a mailing list message to determine whether it was solicited or not.

1.2 Reputation Attacks

An SMTP mail server may be falsely accused of having sent messages to a non-subscriber.

As the situation currently stands there is no way to determine the truth or falsehood of such allegations. A party may even sign up for a newsletter with opposing views so as to be able to complain about the messages sent and hope to cause the mailing list to be put on a blacklist.

Events of this type have occurred in connection with mailing lists of every political persuasion. When complaints are made about a blacklisting they are typically met with further hearsay accusations that the accused was ænotoriousÆ.

1.3 Unsubscribe Problem

Mailing list users often find it difficult to unsubscribe. In many cases requests to be unsubscribed are sent to the entire list.

The un-subscription problem can be a serious problem when an intermediary such as a mailing list gateway is involved.

1.4 Abandoned Subscription Problem

Users often fail to unsubscribe from mailing lists, causing huge volumes of mail to accumulate unread in an unused account or an unread mail folder. In some cases the subscribed email account is also abandoned.

Often a mail user will subscribe to a mailing list on a speculative basis and find that they do not read the list

often.

1.5 Abandoned List Problem

Mailing lists are often created and then abandoned after a period of time after falling into disuse. This can create serious problems if a spammer after discovers an abandoned list without an administrator and starts to send messages to the list.

1.6 Maintenance Problem

The management of a high volume mailing list requires a considerable amount of effort, largely because of the need to manage the problems of subscribers who are unable to unsubscribe, have abandoned subscriptions etc. Another increasing concern for mailing list administrators is that their lists may be blocked by blacklists, often through no fault of their own due to reputation attacks.

2. Deployment Constraints

The deployed email infrastructure is the result of more than twenty years of development, much of which has taken place in an ad-hoc fashion. As such it is vital that any proposal to change that infrastructure be compatible with the infrastructure as deployed and not merely as it is described in theoretical specifications.

2.1 SMTP Deployment Limitations

Many SMTP servers are poorly configured and perform arbitrary and in many cases capricious modifications to messages as they are transmitted.

2.2 SMTP Client Limitations

Adding features to SMTP clients is a slow process. The features offered by mail readers have not changed significantly in the past five years, basic principles of mail delivery have been unchanged for almost ten years. As a result few end users are likely to upgrade their mail client in order to be able to take advantage of a protocol meeting the requirements described.

2.3 Separate Servers for Incoming and Outgoing Mail

Enterprises are increasingly using separate mail servers for incoming and outgoing mail. Even if the end user interacts with a single server it is likely that incoming mail will be pre-processed by some form of mail proxy, particularly if anti-spam filtering is being performed.

2.4 Network Protocol Access Limitations

Many ISPs limit or block completely use of certain Internet protocols. These blocks may be imposed for a variety of reasons that include preventing spam, service differentiation and caprice.

2.5 Existing Features Insufficiently Observed

Many of the problems with mailing lists could be addressed by simply deploying the existing proposals in [RFC2369]. These provide a mechanism that allows mailing lists to use URIs to specify how users can perform functions unsubscribe from a list.

A header of particular interest in this specification is the Mailing-List header that uniquely identifies a mailing list.

3. The Proof of Consent Protocol

The chief deficiency in the email protocols with respect to the requirements is that an incoming mail server has no means of determining whether a recipient did legitimately consent to opt-in.

Most mailing list applications support the use of a challenge/response protocol to verify subscription requests. The mailing list sends the subscriber a challenge token consisting of a string of characters. To confirm subscription to the list the subscriber returns the token to the mailing list.

The Proof of Consent mechanism proposes the use of a second token that is created by the subscriberÆs incoming mail server and forwarded together with the challenge token to the mailing list. The mailing list then includes a copy of the token in every email message sent to provide the proof that it was solicited.

This mechanism minimizes the need for state maintenance at each stage in the mail transfer protocol, avoiding the need for additional per user records to be stored at either the incoming or outgoing servers.

3.1 Initialization

The incoming mail server establishes a shared secret k. In the case that there are multiple incoming servers the shared secret may be established manually or by means of some form of key agreement protocol using public key based credentials.

Once established, the shared secret is maintained for an extended time period such as a year. Incoming mail

servers must keep a record of all shared secrets that have ever been used and the validity intervals in which they were used.

3.2 Confirmation Code

During the subscription process we establish a confirmation code that is cryptographically bound to the mailing list name, the recipient email address and the date of subscription by means of the shared secret:

Code = MAC (mailing-list, recipient, date)

Where:

list
: The string the mailing list will use in the Mailing-List header

recipient
: The email address of the recipient

date
: The day on which the subscription took place.

Each message sent from the mailing list contains a Confirmation-Code header as follows:

```
Confirmation-Code: <date> <confirmation>
```

The mailing list already needs to maintain a per-subscriber record of mailing addresses. The additional state required to support the confirmation code protocol is negligible.

We require the subscription date to be stored explicitly for two reasons, first it requires the mailing list administrator to take notice of the age of subscriptions, secondly it allows the incoming mail server to reject mail with a stale confirmation code that is many years old. This allows a mail server to reject mail sent to a mailing list that a previous holder of an account name subscribed to.

The incoming mail server can authenticate a confirmation code (but not the attached message) by means of the shared secret. Note that it is not necessary for the MAC algorithm to be standardized, it is sufficient for the sender and receiver to use the same one.

3.3 Subscription Process

The confirmation code is generated during the subscription process and communicated to the mailing list server.

We assume that any subscription process involves a confirmation process by means of an email callback loop challenge. The incoming mail server detects a mailing list request for subscription confirmation and causes it to be redirected so that the appropriate confirmation code can be added.

The callback loop challenge contains a new header constructed as follows:

Challenge-Code: <mailing-list> <recipient> <opaque>

Where
mailing-list
The string the mailing list will use in the Mailing-List header
recipient
The email address of the recipient
opaque
An opaque code defined by the mailing list confirmation server to be used for confirmation purposes.

If the user responds to the confirmation request the appropriate challenge code is generated and forwarded to the indicated address along with the opaque code to establish that the user did intend to subscribe.

3.4 Legacy Subscriptions

The confirmation code protocol must support gradual introduction. It must be possible for a mailing list to deploy the protocol without having to re-subscribe existing users. It would be advantageous if this can be achieved in such a way that allows a mailing list administrator to be able to deploy the protocol incrementally and still be able to establish that unsolicited messages are never sent.

When a mailing list server sends a message to a legacy subscriber the confirmation-code header is still present but only the date field is filled, the confirmation field is absent. This alerts the incoming server to the fact that the message is purported to be a legacy subscription.

If the incoming server supports the confirmation code protocol it may query the user to determine whether the subscription is actually authorized and if so provide the relevant confirmation code to the mailing list server to avoid the need for subsequent authorizations.

A similar procedure may be employed when the confirmation code protocol is deployed on an existing incoming mail

server. In this case it is recommended that the service provide some mechanism to allow the user to send confirmation codes to a group of mailing lists at the same time.

Mailing list administrators may defend themselves against malicious allegations by having a copy of the mailing list signed by a digital notary at the time the protocol is deployed. The signature format may be chosen in such a way that validity of each entry in the list may be determined independently without revealing any information about any other list member. Various schemes suggest themselves including use of Merkle hash trees or the XML Signature manifest object. It is recommended that any mechanism chosen use a random mask value within each entry to prevent attackers from finding out that a specified party has subscribed to a list.

This information could be made available through some form of protocol, however it is likely that requiring existing users to re-subscribe will prove more attractive.

3.5 Revocation of Confirmation Codes

The mechanism described only provides for the authenticity of subscription requests to be established. No assurance is provided for un-subscription requests, nor is the protocol easily modified to achieve this directly.

The confirmation code is cryptographically bound to the mailing list identifier. An incoming mail server or spam filtering application can filter incoming mail on the basis of the mailing list identifier. Although this requires the service to maintain state the overhead is minimal and saves considerable resources in the long run. A mailing list may choose not to observe requests to unsubscribe but there is no incentive to do so if the messages are unlikely to be read.

4. Security Considerations

4.1 Replay Attack

The consent to receive mechanism provides proof that the recipient of an email message has consented to receive messages from a specified source. It does not provide definitive proof that a particular message originates from the source specified.

An attacker that obtains a consent token for a particular recipient/sender combination can generate an arbitrary volume of messages containing the token.

This vulnerability is unlikely to represent a significant risk in the context of spam mitigation. It is highly unlikely that a spammer could obtain a sufficiently large number of consent tokens to make bulk distribution of spam through this mechanism feasible.

The vulnerability may be eliminated through use of the consent token mechanism in combination with an authentication mechanism.

References

[RFC2369] http://www.ietf.org/rfc/rfc2369.txt

[ListSpec] http://www.nisto.com/listspec/

Copyright Statement

Author's Addresses

Phillip Hallam-Baker
VeriSign Inc.
Email: pbaker@verisign.com