Consent-based delivery extension for the SMTP protocol

Claudio Telmon - claudio@telmon.org

Introduction

E-mail has been one of the key success factors of the Internet, and is still one of the most deployed services. However, most messages are currently undesired from the receiver's perspective. While these include spam, phishing and messages carrying malware, many messages with a less malicious intent are also undesired: unrequested mailing lists subscriptions, harassment messages or messages from unknown people. Increasing resources are invested in delivering these useless or disturbing messages, and even more in recognizing which must be discarded before delivery. Users often need to setup more than one address, just in order to deal with spamming and privacy issues, and still may need to replace some of these addresses as they become unusable.

The option to select who to communicate with is offered by most modern communication protocols and frameworks. These include most social network frameworks, instant messaging, and Skype. In the same way, telephone directories, created for plain old telephone, are usually not available for cell phones, and people actually put a lot of effort trying to keep their cell phone numbers private and to select who to communicate with.

Moreover, many regulations mandate some form of consent prior to communication, an example being the European Directive 2002/58/EC, concerning privacy protection in electronic communications, requiring opt-in[1] in case of direct marketing [1].

This suggests that the SMTP protocol could also benefit from a consent-based framework for message delivery. In this proposal, the following key requirements have been considered:

1. Consent must be a receiver decision[2]. Reputation-based frameworks and trusted third party based authentication may help, but at the end the receiver is the only one who can actually take the final decision on which messages and senders are desirable.
2. The whole framework must be optional, both as implementation and in the decision to enable it. This will provide for a gradual deployment of the framework in the current message delivery infrastructure. In this paper, an MTA or MUA implementing the framework is called “consent-aware”, while an address for which the consent framework has been enabled is called “consent-enabled”.
3. The framework must coexist with most current antispam and antivirus tools, so that enabling the consent framework doesn't require to renounce to other effective protections.
4. There must be a way to contact address owners and ask for permission for further communication, that the receiver can optionally disable.
5. Whenever possible, undesired messages should be rejected by an MTA, instead of being accepted, delivered to the user mailbox and then eventually discarded.
6. The framework must not require any centralized service or server, as this would hardly fit in the current email delivery architecture.

The Anti Spam Research Group worked on a consent framework for fighting spam in year 2003[3]. In year 2004, the ASRG changed its charter to non-consent based work[4]. However, nearly five years

1 The US legislation and its CAN-SPAM Act adopted an opt-out approach instead, see http://en.wikipedia.org/wiki/CAN-SPAM_Act_of_2003

2 Here, the receiver is whoever has the authority to define usage policies for an address. It may be a home user, but also an organization. However, for simplicity, in the rest of the paper the “receiver” is the person (or program) who actually uses the address

3 http://www.shaftek.org/publications/asrg-consent-framework.html

4 http://asrg.sp.am/about/history.shtml

have passed and many things changed since then, most notably the appraisal of phishing; while useful against spam, the ability to express consent to message delivery may be now useful to protect against additional threats, and to provide a mean to contact addresses in a way that complies to privacy regulations. This proposal will be described using as much as possible the scheme and terminology defined in the draft work of the ASRG in 2003 [11].

Previous and related work

A big and increasing effort is being put in the attempt to limit the impact of spam and other undesired messages. Many techniques have been adopted or proposed, but most try to address specifically UBE or messages carrying malware, e.g. trying to solve the problem of forged senders. While this is a relevant percentage of undesired messages, a consent framework aims at a control also on messages that are sent from legitimate addresses, are not UBE and don't carry malware, but are undesired by the receiver; the same messages could be tolerated or even requested by other users. Also, some techniques are not suited for widespread deployment, e.g. because they knowingly base their effectiveness on actions that spammers are currently not performing: should these techniques be widely adopted, then the spammers would change their behavior, in a well known arms race.

A consent charter was firstly adopted by the Anti Spam Research Group of the IRTF in 2003. However, this charter was abandoned one year later. A draft document for a consent based framework was produced. Not much work seems to be available on explicit consent management by end users.

Ham passwords [5] are very similar to the proposal described in this paper. Basically, the idea is to publish/disseminate a password that senders would include in their messages, e.g. in the subject, in order to demonstrate that they are not bots[5] and had access to the password. The password is hard to link to the address by a non-human, e.g. they are just on the same web page and a sentence somewhere states “use this password in the subject in order to contact me”. This qualifies the message as “probably not spam”. The receiver would then inspect the message and, if he/she desires to receive further messages from that sender, add the sender address to a whitelist. However, no formalization of the concept seems to be available. Also, ham passwords are supposed to be handled by hand by the end user. As a consequence, few ham passwords can be probably handled without causing a lot of trouble. Ham passwords can be only used to verify new addresses once, since it would be very troublesome to add a ham password to every message, and as a consequence address forgery, e.g. for phishing purposes, would still be possible. Moreover, no mechanism is described in order to allow a person without a valid password to try to contact the address anyway. Nevertheless, the basic idea of providing a token only to the desired correspondents is the same adopted in the proposed framework[6].

Another similar approach, maybe more effective than ham passwords, are address name extensions[7], where receiver addresses are e.g. in the form address+token@domain Usage of these tokens can be automated. The same address without tokens can be used in order to contact the receiver when no token is available. This approach has the benefit of putting the token in the envelope of the messages, where it can be better handled by the receiver's MTA[8]. Also, sender MUAs don't need to explicitly manage tokens, since they are part of the address, and as a consequence receivers willing to adopt a consent framework have less problems with their correspondents, which don't need to change their habits or to install software. Detection of the token in the address would probably require either a change in the address format standards, in RFC 5322

5 http://en.wikipedia.org/wiki/Internet_bot

6 See also the related discussion on the ASRG mailing list, http://www.ietf.org/mail-archive/web/asrg/current/msg10474.html

7 This technique is called in many ways, including “plus addressing”

8 However, the consent token described in this paper could also be put in the envelope, see the “implementation issues” section

in particular, or some guesswork by every SMTP server. However, tokens would be exposed, since addresses appear in many headers and places in the message. An MTA couldn't probably clean up messages, as required when the message is destined to many receivers, without implementing a procedure of the same complexity of the one described in this paper. As an example, a message with a valid token sent on Cc: to a mailing list would expose the token. While the token can be revoked and and a new one can be issued, this wouldn't solve the problem, as the new address would be published again at the next message in Cc: to the mailing list. In fact, while the receiver could reserve and publish an address without tokens for consent requests, almost every published address would carry a valid token. Zoemail[9] uses name extensions.

Some solutions adopt a challenge/response approach[10] in order to verify that the sender is not forged and is a real person: as the MTA receives a message from a new address, it sends back a challenge; if the sender answers, then it is supposed not to be a bot and the address is whitelisted. The rationale is that challenges sent to forged addresses won't get a response, and that spammers using real addresses won't waste their resources trying to answer to the challenge. These solutions have the disadvantage of provoking backscatter whenever messages have a forged sender address. Also, the hypothesis that spammers won't answer to challenges may be valid for UBE, but is not useful for other kinds of undesired messages, and could lose validity if the mechanism were so widely deployed that spammers couldn't just ignore addresses that require an answer. Moreover, this approach is suited for “one-time” verification of sender address; the sender cannot be expected to answer to a challenge for every message sent, so phishing would still be easy. Finally, it doesn't address the issue of legitimate senders whose messages are undesired from the receiver's perspective.

Other solutions and proposals exist, only dealing with the problem of “first contact” whitelisting. These include the “Medina standard”[11] and the “Knock knock” proposal.[12] however, these proposals offer little protection against sender address spoofing.

Other solutions propose relevant changes to the current e-mail infrastructure, and thus are not considered here, since they miss one of the key requirement.

Reputation-based systems try to rate the reputation of a sender based on different parameters, e.g. how its messages are accepted by receivers: reputation may be increased if messages are accepted, up to a threshold where the sender is whitelisted, and decreased if messages are rejected, down to a threshold where the sender is blacklisted. In many cases, “sender” means the sending MTA or an IP numbers block. Reputation-based systems require a widespread cooperation effort, and may score well for messages for which a wide consensus can be reached about being spam or malicious. While the effectiveness may vary [6] and unusual problems may arise [7], the major limit is that it is not effective on messages that can be undesired by some users and permitted or even requested by others.

Sender authentication schemes, including sender-ID [8] and DKIM [9] deal with authentication, so they could be used in order to whitelist a limited set of correspondents[13]. This would implement a consent scheme, but it wouldn't provide any mechanism for new correspondents to solicit the consent, so it wouldn't be clear how addresses could be added to the whitelist. Also, the way a whitelist could be defined and managed is not part of these protocols, so no complete support to a consent framework is provided.

Existing services often integrate two or more of the described techniques in order to achieve better results. The consent framework presented here is not meant to replace these techniques, which

9 http://www.zoemail.com/howitworks.htm
10 http://en.wikipedia.org/wiki/Challenge-response_spam_filtering
11 http://newstriangle.tripod.com/
12 http://www.tundraware.com/Technology/Knock-Knock/
13 While blacklisting could also be possible, a consent framework couldn't be built on it, since unknown addresses should always be blacklisted, and this would be actually implemented with a whitelist.

could still be used in order to reach an even better control on delivered messages.

Note that some of these proposals are many years old, especially those explicitly dealing with consent, and never reached a widespread adoption. Most were proposed as antispam solutions, while antispam research moved in a non-consent-based direction. Consent-based research may become more interesting as consent becomes relevant per-se, e.g. in response to regulation requirements and to new user requirements.

General overview of the framework

The basic idea of the framework is that an address owner can distribute *consent tokens* to people he/she is willing to receive messages from, the same way he/she distributes a cell phone number. A message is then accepted for that address only if it carries a valid token. Tokens can be communicated between users, again as cell phone numbers are shared, based on personal evaluations on appropriateness. Messages with a special formatting can be accepted without a valid consent token, in order to allow people to attempt a "first contact" with the address owner.

Note however that the "first contact" problem may be a false problem in this framework. The problem arises from the hypothesis that somehow the "any to any" model should still be working. However, this is not the case for some of the most deployed communication channels, e.g. cell phones. Since cell phone numbers are not usually listed in public directories, there is no way for "anybody" to contact people through the cell phone, and this is usually considered a plus. People with a legitimate reason to contact someone through the cell phone usually manage to access the number through some other communication channel, e.g. via a face-by-face meeting with someone already knowing the number. Still, the ability to send messages without a valid token may be required in order to recover from some error situations.

The framework is conceived as an extension[14][2] to the SMTP protocol. Consent tokens are random sequences of characters conforming to the syntax constraints set in RFC 5322 [3] for unstructured header field bodies. For a message to be delivered to a receiver, it must carry a consent token that is acceptable for that receiver. The decision is taken by the *receiver's MTA*, that is usually the MTA that will deal with final delivery to the receiver's mailbox/mail repository[15],. For this to be possible, the receiver will need to be able to manage a list of valid consent tokens to be accepted by that MTA.

Consent tokens are very similar to Ham passwords[5]; however, their management is part of a more comprehensive framework, as discussed later on, and consent tokens are more suited for a widespread and automated deployment. Consent tokens are in general different for each couple sender/receiver, since they are generated by the receiver and then provided to senders.

Consent-tokens are exchanged off line or piggybacked on messages. Whenever someone doesn't own a consent token to communicate with somebody else, he/she may send a message asking for permission. This kind of message must carry a specific header and has some limits in size and format: text only and few lines in size. With these limits, the message is still enough to explain the reasons for the contact; at the same time, this contact request message can be easily recognized by the receiver and by the receiver's MTA/MUA and treated as such, e.g. discarded or inspected but not in any way considered as a normal message. This way, phishing should be considerably more difficult. This message still needs to be inspected by antispam tools, but due to the format and size constraints, and to the specific semantic of this kind of messages, their task should be easier than e.g. detecting spam messages in images. Also, many kinds of unsolicited messages, e.g. including images, would not be delivered. Finally, by completely blocking consent requests, protected

14 SMTP service extensions are defined in RFC 5321. "Local extensions" can be defined, with keywords beginning with "X-", to be used through bilateral agreement.

15 It could be actually any MTA handling incoming mail for the address, provided that the address owner and the MTA manager agree to manage the consent token on that MTA. However, the philosophy of the framework is clearer with this definition of a receiver's MTA.

mailboxes could be created that could be used e.g. by children with much less risk of harassment and stalking than with unprotected ones.

Consent-based delivery should be enabled on a per-recipient basis on the receiver's MTA. Receivers not willing to adopt this kind of control will not enable it, and in this case the MTA will deal with their messages in the usual way. This could open to new email service offers, based on protected mailboxes, that could deploy the existing infrastructure, even with a limited pervasiveness of the framework.

Most of the work will be made by the MUAs, both from the sender's and the receiver's perspective. However, receiver's MTA support is critical: this MTAs will need to deal with specially crafted headers and with a consent-token database.

Trust model

The goal is to implement a trust model similar to the one people adopt for private phone numbers, typically cell phone numbers. In this model, the infrastructure is only required not to disseminate the number. People then provide the number to whoever they want. Also, people owning the phone number of another person can give it away to other people, based on their judgment on appropriateness. This is usually very effective at keeping access to phone numbers limited to people the owner is overall willing to communicate with. The reason why this model works is that we are living in a world where people are connected through different social networks and communication channels, including face-to-face relationships, and no single channel is required to be the only way people can interact. In fact, when thinking at email, if someone has legitimate access to ad email address, then he/she has collected it somewhere through another communication channel.

There are, however, public numbers, e.g. published on lists and websites. In this framework, public email addresses wouldn't enable any consent-based protection, so everybody could contact them. No trusted third party is required.

However, the “phone number” model has a severe limitation: once the number falls prey of a miscreant, it is hard to change it and to avoid further undesired dissemination. The proposed model for email deals with the problem by providing a different token to each correspondent. If a token is compromised, it is possible both to detect the “weak ring”, and to invalidate the token, providing a new one to the legitimate correspondent, without affecting any other correspondent.

New e-mail headers

Three new e-mail headers are required for the implementation of this framework. A fourth header is required in order to support error notifications; while the framework can deal with error notifications, this somehow limits its effectiveness, and other protocols (e.g. Bounce Address Tag Validation[16]) may be better suited for the task.

X-Consent-token: [<address>,]<token>

address is the address the message is meant to be delivered to;

token: is a (pseudo)random string of characters conforming to the rules for unstructured headers.

X-Consent-request: <token>

declares that the message is a consent request and provides a token for possible answers

X-Consent-new-token: <address>,[<token>,]<token>

carries in piggyback a new token to be used by the receiver whenever it will communicate with the sender using <address> as receiver (so <address> is usually the sender of the current message). The

16 http://mipassoc.org/batv/

first, optional, token is a token already valid for <address>, while the second one is the one hat should replace it.

A message may carry either a X-Consent-request header, or one or more X-Consent-token header. Moreover, a message may carry a X-Consent-new-token.

X-Consent-MTA-token: <token>

carries a consent-token to be used by MTAs for error notifications related to the message carrying the header, as described in the section on error handling.

Basic MTA message handling

This section describes how a consent-aware SMTP server deals with messages with respect to the consent-based framework. Any other action performed by the MTA and not described here is performed as usual, including spam and malware detection. This basic description only deals with basic issues and one recipient per message. In this case, no action or support is required by a SMTP client, although, as shown later, client support may help.

In a SMTP transaction, a consent-aware MTA will answer to an EHLO command including the X-CONSENT extension[17]; it then verifies if it is the receiver's MTA for the address provided in the RCPT TO: command (or more generally, is configured to handle consent for the receiver's domain). If so, it will check if the address is consent-enabled.

If the MTA is not the receiver's MTA or the address is not consent-enabled, the MTA will handle the message in the usual way. If the address is consent-enabled, the MTA checks if either a X-Consent-request or a X-Consent-token headers are present in the header section of the message[18]. If the message doesn't include any of these headers, the message is refused without any further check on the body, answering with a 550 error message to the sender that clarifies the need for a Consent-token (something like “sending to this mailbox requires consent but no consent token provided”).

If the message includes a X-Consent-Request header, then the message is checked for compliance with format and size restrictions. Suggested restrictions are that a non-void Subject header must be present, and the message must be text only (MIME plain/text) and less than 512 characters[19] (size may depend on the character set). If the message doesn't comply to the format requirements, the message is rejected with a 550 error.

If the message includes a X-Consent-token header, then the token will be checked with the database of valid tokens for the receiver's address.

If the token in the header isn't found in the database, then the message is rejected with and an appropriate 550 error. If a match is found, then the message will be delivered. Security checks (e.g. antivirus) are still needed, but the message will be treated as not being spam.

Note that a consent-aware client could identify itself as such, detect the consent token in the message and provide it in the envelope, as part of the RCPT TO: command. This would enable the SMTP server both to reject the message before receiving the body, and to selectively reject recipient addresses.

Token database management

The receiver is supposed to be able to manage the token database associated with its address in the receiver's MTA, by adding and deleting tokens. This can be done through a side channel (e.g. a web

17 Note that messages can be successfully delivered even if the SMTP client doesn't support the Extended SMTP. However, consent-aware clients may need this information, as described later on.

18 The MTA can still perform any spam check on the envelope, however any decision should be taken only after X-Consent header verification.

19 While these are suggested constraints, once defined they must be adopted as mandatory for every implementation.

interface) or via an SMTP interface, whenever SMTP authentication and privacy between the receiver and the MTA are deemed sufficient[20]. The side channel can be implemented in many different ways; a specific protocol[21] is not part of this proposal and will not be further discussed. It must be noted however, that the client side of this protocol must be usually implemented by MUAs, so it may be of great benefit to achieve a standardization of this interface.

Access to this database by the MTA will be a read only access based on a key composed by {address, token} to retrieve a single record. Read-only access may help in database management, since the database may be updated by MUAs and then pushed on the MTA without problems related to concurrent access. Records may include some additional information, as described in a following section.

The management interface must enable the address owner to add and remove tokens, while a database manager must be able to add and remove addresses (and all associated tokens). Also, an address owner should be able to retrieve all the tokens related to its address, for backup and maintenance purposes.

MUA support

Both sender and receiver MUA support is needed for proper and user-friendly implementation of the framework. The basic requirements are to be able to:

- associate tokens to addresses;
- insert X-Consent headers in the messages;
- update the token database on the MTA.

A simple implementation would need to be able to associate two tokens to every address in the address book: one is the token to be used when sending messages to that address, and the other one is the token expected to be found in messages from that address. Correspondents not requiring consent tokens may be marked as such in the address book. Whenever a message is sent to an address in the address book, the appropriate token must be added to the message in an X-Consent-token header[22]. If a message is sent to a new address, then the user can manually add a token if available, send the message without token (the receiver won't receive the message if the address is consent-enabled), or send a consent request. A consent request is a normal message respecting the above defined limits in size and format, and carrying a X-Consent-request header. In this header, the MUA will insert a (pseudo)random generated token[23] that will be used by the receiver for its answers, and that will be added in the respective address books. This token will be also added to the sender's MTA database, so that answer messages carrying it can be properly delivered.

When receiving a consent request, the MUA will emphasize this fact, e.g. with a colored bar[24], so that the receiver will not confuse it with other messages and can evaluate if a contact is actually desired. If so, then the receiver can send an answer using the token provided in the X-Consent-request header, and adding a X-Consent-new-token header with a new random token to be used by the correspondent for further communications. This new token will be added to address books and to the MTA database, so that future messages carrying it in a X-Consent-token header can be properly delivered. The receiver can also answer without adding any token if he/she feels that the

20 In many cases, a SMTP interface may not be appropriate, since the user may only have access to a submission MTA, which is not its receiver's MTA.

21 This is what the ASRG draft would call “Consent Policy Exchange Protocol”

22 Add-ons already exist for some MUAs that permit to add custom headers; such as Mnenhy for Thunderbird , http://mnenhy.mozdev.org/

23 Effective database management, as will be noted later, requires that for each address, some valid but unused tokens be inserted beforehand, so that whenever a new token is needed, it is in fact selected among those already registered in the database. The newly generated token will then be added to the database as unused token.

24 The same way e.g. Enigmail (http://enigmail.mozdev.org/) displays a colored bar in order to inform the user about signed messages

message exchange will end with this message (e.g. to politely reject a contact).

Finally, receiving a message with a X-Consent-new-token means that the second token in the header must replace an existing token, optionally specified as fist token in the header. This header is meant to be used when a receiver has invalidated a token by removing it from the MTA database, and needs to send a new one to a correspondent. Proper handling of this token is critical, as detailed in the “Security Considerations” section.

Tokens dissemination and management

While consent requests and responses are a way tokens can be created and disseminated, most user are expected to disseminate tokens through other means. As said, the expected model is similar to the way cell phone numbers are handled. People usually tell their cell phone numbers to persons they want to be called from. Also, some write their cell phone numbers on their business cards and vcards. Moreover, phone numbers are communicated from person to person following personal evaluations on privacy and appropriateness. This model is usually enough to avoid excessive misuse of cell numbers. The proposed framework allows for the same model, with some additional benefits. In order to understand how the model works, some considerations are needed.

The first one is that tokens can be pre-authorized, e.g. by associating them to fake addresses in the address book and forcing as a consequence their insertion in the MTA database. A pre-authorized token can be written (or even printed) on a business card, written in registration forms for conferences or mailing lists, sent through other channels or given by hand. This kind of tokens will be used by the person receiving it for its first communication. After this first contact, a personal token will be provided through a X-Consent-New-Token header..

Another important point is that while tokens are usually expected to be random, any character string can be a valid token. Some easy-to-remember tokens can be pre-authorized so that they can be communicated by voice to other people. As an example, the string *ThisIsAValidToken* is a valid token, yet it's easy to remember and communicate[25]. These tokens may have the same use of pgp key hashes[26], which are communicated in order to validate subsequent messages where the actual key is sent.

These token have two advantages over cell pone numbers, since they are meant to be different for every correspondent. The first one is that token dissemination can be traced. Whenever someone new contacts an address using a valid token, the token can be linked to the person or address it was first communicated to, which can be reprimanded if the dissemination has been inappropriate. The second one is that tokens can be easily revoked and replaced. If a token has been communicated to an undesired correspondent, the token can be revoked, and new token can be sent to any other correspondent currently using the same token, if the receiver is still willing to communicate with them[27]

An example can be a token written with the email address in a conference registration form: if too much advertisement arrives carrying that token, or the address is subscribed to an undesired mailing list, the token can be easily revoked without impacting other legitimate communications. This example shows how consent can be denied in order to block messages that are not spam in the common definition of “undesired bulk messages”, but are nevertheless undesired from the receiver's perspective.

Additional basic issues

Some important issues are to be dealt with in order to complete the description of the framework.

25 This kind of tokens is easily guessable; this problem will be dealt with in the “Security considerations” section

26 http://en.wikipedia.org/wiki/Key_signing_party

27 In fact, it could be preferable to replace a token that has been shared by two correspondents anyway; otherwise, should the token be abused, it would be unclear which of the two correspondents is responsible of the abuse.

Multiple receivers in the envelope

Many messages have more than one receiver in the envelope. Some of these addresses may be consent-enabled and thus require consent tokens/headers, while others may not. In order to be delivered in all the appropriate mailboxes, the message needs to carry all the required consent tokens, which must be delivered only to the appropriate receivers. Tokens meant to be used in communications with one address must never be delivered to other addresses.

When composed, the message must include all the relevant consent tokens, each in a separate X-consent-token header. Since there are multiple receivers, each X-Consent-token header must include a destination address, so that it can be linked to the appropriate receiver[28]. It's up to the sender MUA or to the submission agent [12] to ensure, as the message is submitted, that addresses in the headers and in the envelope match.

Whenever a message is forwarded to a SMTP server, the SMTP client must ensure that only the tokens related to the receivers of that message are included. If a message includes many receivers and many tokens, and copies of the message are forwarded to different MTAs dealing each one with a subset of the receivers, then the X-Consent headers not pertaining the appropriate subset of receivers must be deleted before any delivery attempt.

Whenever a SMTP client supporting the consent framework (including the sender MUA) starts a connection to a SMTP server in order to send or relay a message, it checks the EHLO response for the presence of the X-Consent extension. If the SMTP server is not supporting the consent framework, then a separate message must be delivered by the SMTP client to the SMTP server for every receiver associated to a consent token[29]. This way, even if the following MTAs are not aware of the consent framework, the tokens will still be delivered only to the appropriate receivers. Note that removing the X-Consent-token headers from the message may violate the requirement of RFC 5321, that relay MTAs don't modify the headers of the message. However, the user, the MUA and all involved MTA are implementing the consent framework, and are aware of these possible changes in the message[30]. Should there be any reason not to accept these changes, an MTA or submission agent can reject the message with a temporary error, asking to resubmit it with separated recipients[31].

Multiple recipients must also be handled with care by SMTP servers receiving messages from non-consent-aware SMTP clients[32].

In this case, when an SMTP server receives the first RCPT TO: command for a message, it checks if it is the receiver's MTA for that address, and if the address is consent-enabled. It will then accept subsequent recipients in the same state (consent enabled or not) while rejecting other recipients with a 4xx temporary error. This ensures that the receivers of a message will all be in the same state with respect to the consent framework, even if the SMTP client is not consent-aware. A message with the recipients in the opposite state will be resubmitted later on by the SMTP client[33]. This way, messages to be delivered in mailboxes will only have either enabled or non enabled recipients. This

28 An X-Consent-token header without address carried in a message with multiple receivers would ease dictionary attacks (see the “Security Considerations” section) since a token would be tried against all the recipient addresses. Also, it would require a more complex search for a match on the receiver's MTA

29 A single message can be delivered for all recipients for which no consent token is present in the message

30 The message cannot have multiple consent-enabled receivers if relayed to an MTA not supporting the consent framework.

31 This shouldn't violate the requirements of section 4.5.3.1.8 of RFC 5321, since it isn't just the number of recipients that causes the error. However, the use of error 452 as described in section 4.5.3.1.10 may be inappropriate; a specific error code should be probably defined

32 Note that this case should only be related to messages with no consent tokens, since non-consent-aware MTAs should never handle messages with multiple receivers and carrying consent tokens. Consent-aware SMTP clients can be recognized by their use of consent extensions in the envelope.

33 This behaviour is consistent with greylisting. If greylisting is adopted, then recipients with the consent framework disabled are at first always rejected with a temporary error, while recipients with the consent framework enabled are always accepted at the first try, since spam check are replaced by checks on X-Consent headers and tokens.

is required because the SMTP server will need to reject messages to consent-enabled addresses, whenever no consent header is present, with a 5xx error informing that an appropriate token is needed. However, the sender of the message may be unaware of the consent framework and may be unable to deal with the error. With the described workaround, messages destined to non consent enabled addresses would be unaffected. This behaviour is needed, since otherwise messages to lists of recipients couldn't be delivered whenever one of the receivers enables the consent framework.

Forwarding messages

A consent-enabled address may be configured so that whenever a message is to be delivered to that address, it is forwarded to a different address instead. The tokens in the message may not be valid for the address the messages will be forwarded to. It's up to the forwarding agent to add an appropriate token and, depending on its configuration, to delete the original tokens. Users enabling message forwarding on one of their accounts will need to provide an appropriate consent token for the agent. This may require the receiver's MUA to manage many different databases, which the user may also decide not to keep aligned (e.g. a correspondent may be enabled to communicate with one address, but not with another). The option to only consent-enable the final destination address may seem easier to manage, but it would produce backscatter whenever a message is accepted by the first address' MTA and then discarded by the final one. In fact, consent should be enable as early as possible in a chain of forwarding addresses.

Unless the address owner decides to do otherwise, X-Consent headers should be removed from messages forwarded to other addresses, since the associated tokens could then be disclosed to unexpected recipients. However, users should decide to keep tokens in forwarded messages whenever possible, since otherwise the original token, and so the related information in case of token compromise, would be lost. However, real-world testing of the framework will be needed in order to sort-out all the different cases, and to determine the best policies for token management in case of message forwarding.

Mailing lists, address expansion and address rewriting

The same issues must be dealt with by consent aware MTAs performing address expansion and rewriting. The MTA will need to store appropriate tokens along with the addresses that will be inserted in the message, and may need to delete some X-Consent-token headers from the message[34]. The same considerations made in previous sections on RFC 5321 compliance of headers modification apply here with respect to section 3.9 of the RFC.

Error notifications delivery

Whenever a delivery error notification message needs to be delivered to a consent-enabled address, an appropriate token would need to be included. This section describes how to handle this problem. However, specific proposals for bounce messages validation exist, which may be more appropriate and efficient.

An X-Consent-MTA-token header must be included in every message whose return-path points to a consent enabled address, but only if the receiver is also consent-enabled: if the receiver is not, then it could mishandle the token, e.g. publish it. A single token will be adopted for every message sent in a given time frame (e.g. four hours); subsequent messages will carry a new token. This token will stay enabled for the time required for proper error delivery (e.g. four days) and then be revoked. With the given values (a new token every four hours and revocation after four days) only 24 different MTA tokens will be enabled at a given time. Also, if a token is not used for a temporary

34 In order not to modify every mailing list management tool, this could be performed by a proxy implementing the consent framework and acting as a submission agent.

delivery failure in a given time frame, it can be invalidated before the original timeout[35].

An MTA token may be marked as such in the sender's MTA token database, so that received messages using that token may be checked as being syntactically conforming to the structure of an error message.

If for any reason adelivery error notification needs to be delivered, and no MTA token is available to the involved consent-aware MTA, but the return-path is, then the MTA can still send an error message conforming to the X-Consent-request constraints.

Also, MTAs should remove MTA tokens whenever the Return Path is modified.

Note however that non consent aware MTAs could be unable to deliver their error notifications. This means that error messages without consent tokens may need to be accepted anyway, as discussed in the section on the effectiveness of the framework. While this could suggest that MTA-tokens are useless, it must be considered that closed groups of addresses and overlay networks implementing the consent framework would not produce error messages without appropriate tokens, so it would by a pity if they were forced to accept error messages without valid tokens anyway. Also, error messages carrying a valid token wouldn't need to be checked by antispam tools, so it's in the interest of any organization implementing the consent framework for its users, to use the MTA tokens whenever possible. Should an organization or a user experience too many inconveniences due to MTA tokens, they could safely disable them.

Additional fields in the MTA's tokens database

A record in the token database of the receiver's MTA may include some additional fields, besides the couple {address, token} which is used as search key. One is the above mentioned “MTA token” bit. Also, since MTA tokens need to be revoked after a fixed time, and this may be for many reasons impractical to do through the MUA, records may also include a “valid until” time field. This may be used to automate the revocation of MTA tokens, or for any other token that the address owner is not willing to give unlimited validity to. Also, it can be used to postpone the revocation of tokens, in order to allow the delivery of already-sent messages. This last problem may also need some real-world testing in order to be properly evaluated.

It can be also useful to limit the number of messages that can be received with the same token. This is especially true if the token is sent as a response to a consent request from an address that the receiver suspects to be a trap to collect addresses and tokens. Every time a message carrying that token is received, a counter in the record is decreased. Note that this would require the MTA to be able to write in the database, which is otherwise not required ans may decrease the efficency of database access and management. Such a counter could be useful also for MTA tokens, for the same reason.

Effectiveness of the framework

The main goal of the framework is that only people/addresses a token has been provided to can send messages to a consent enabled address. Tokens can be obtained from the owner of the address, from someone who decides to share its token, or can be illegitimately collected. Tokens used in a way that is not conforming to the owner's desires can be revoked as soon as a single undesired message is received. Information about which token and correspondent are involved in the compromise is available in a format that can be understood by the end user, which is not the case if e.g. useful information on a compromised address can be inferred from trace headers.

An additional goal of the whole framework is to provide an extension to the current SMTP e-mail framework that allows for consent expression for the address owners willing to enable it, while

35 Considerations on timeouts are subjective for every user/MTA, since no timeouts are set in the SMTP standard for the first temporary failure notification or for permanent delivery failure.

leaving “as is” the usage and implementation of the SMTP e-mail framework whenever the consent framework in not needed[36] or implemented.

In this respect, this proposal partially fails, since error messages generated by non consent aware MTAs must be either discarded, so that senders may become unaware of some delivery failures, or accepted, opening a door to undesired messages. Since not every MTA can be expected to implement the consent framework, and loosing error messages is probably not an option for most address owners, error messages will need to be accepted: fake error messages will be dealt with by other protections.

While this seem to severely limit the effectiveness of the protocol, it must be considered that the problem with error messages is limited to some forms of malware and spam, which are not the only target of this protocol (and maybe not even the most relevant one). Spam and malware presenting themselves as an error message are a well known problem, and other tools may help well with it. Also, forged error messages won't produce backscatter. Besides this limit, only influencing the protection of consent enabled addresses, the framework can be implemented and deployed as an optional extension to the current framework, without hindering the usual delivery of messages.

However, this proposal may violate the requirement if RFC 5321 not to modify message headers, since X-Consent headers may need to be deleted by relay MTAs.

Messages from unknown addresses or from forged addresses are limited to consent requests and error messages: the framework provides actual control on consent for other kinds of messages. For additional protection, the receiver's MUA can highlight when a legitimate token is used by a sender that is not the one the token was intended for.

This mechanism provides a much more powerful control than opt-in/opt-out mechanisms for commercial mailing. In fact, in countries where privacy and spam regulations enforce opt-in, some companies already send a “consent request message” before sending commercial messages. These consent requests often almost already fit the constraints of this framework, since they are short messages just explaining the reason of the contact and asking for permission to send further commercial messages.

Also, the framework provides a way to revoke consent that don't need the cooperation of the counterpart.

The framework avoids any form of “trust”, besides trust in proper operation mainly of the receiver's MTA: no information sharing or third party evaluation on messages or senders is needed, and no centralization points are required, thus actually giving to the receiver the control on whose mail to accept[37].

Also, while enforcement occurs at the MTA, tokens are actually under the full control of the receiver, which can move to another provider carrying its tokens with him. Tokens aren't even bound to a specific address: the receiver may accept messages with the same tokens on a different email address, and thus can change email address and just communicate the new address to his/her correspondents, with no need to renew the tokens. Consent management is fully end-to-end, with no dependance from specific services or service providers.

The effectiveness of the framework with respect to specific kinds of messages varies depending on the kind of unwanted message that is sent to the address.

Malware propagates also through legitimate channels, e.g. from a compromised MUA. As a consequence, anti-malware protections cannot be disabled even for messages that properly passed

36 Some addresses may need to be freely contacted by anyone, e.g. the contact address of a company. These addresses just shouldn't enable the consent framework. This is also true for the “postmaster” address.

37 As said, anyone owning a valid token for the receiver's address can share the token with whoever he/she decides to be proper, but the framework enables the receiver to quickly revoke tokens, so that the dissemination of token can be considered more an advantage than a problem.

consent verification. However, since messages without tokens can only be consent requests or error messages, where e.g. executable attachments are not expected to be found, these legitimate channels should be the only ones available for email malware propagation, thus reducing the impact of some worm infections.

Unsolicited Bulk Email must be sent as consent request or error message. Only in few cases, UBE can be sent with appropriate tokens; an example may be a message sent to all the participants to an exposition, which provided token in the registration form; however, this kind of situation can be blocked by the participants after a few messages by revoking the token.

UBE sent as consent request or delivery failure notification message should be much easier to recognize by spam filters, and with less computational effort. Moreover, only UBE which is deliberately sent with the aim to get around protections will be sent as error message. This is a kind of UBE for which the current protections may be very effective. Also, hosts that start sending a high number of consent requests or error messages could be detected both by the sender ISP and by cooperative spam fighting tools.

Consent requests have syntax constraints that help in detecting spam, but also a semantic that can be expected for this kind of messages. An example may be that the name of the receiver can be expected to be in the message. Spam that passes the MTA controls could then be caught by more specific MUA controls.

Other UBE which is just sent with “little care” in the receiver's interest in receiving it (e.g. newsletters, commercial mailing lists etc.) is properly blocked by the framework.

So while this framework is no replacement for antivirus and antispam tools and techniques, it can ease their work and significantly reduce their workload, both by reducing the number of messages they must work on (antiviruses don't need to check consent requests, antispam tools don't need to check messages carrying proper tokens, and many malicious messages should be dropped or rejected anyway), and by setting more constraints on the format of spam messages.

Also, it can reduce the effectiveness of spam messages that still manage to be delivered to mailboxes. Many address owners, and especially home users which are a major target for spam messages, may prefer to directly trash consent request messages instead of dealing with spam, the same way many people don't answer to cell phone calls when the calling number is unknown. So, the framework could reduce both the volume and the effectiveness of spam messages delivered to mailboxes[38].

Forging a sender address becomes useless unless a valid token is available, which is usually not the case for phishing attempts. Note that someone owning a valid token could try to forge the address of another correspondent. The message would be delivered, but could be recognized by the MUA as carrying an unusual couple {sender, token}, and be highlighted as suspicious.

However, this framework should be mostly effective for undesired messages that cannot be classified as malware or spam in the sense of “bulk unsolicited mail”, but are nevertheless undesired. These messages won't usually try any kind of deception, are not dealt with by most current protection tools and will be properly blocked by the framework based on the subjective choices of the receiver. These include e.g. mailing lists to which the user subscribed or has been subscribed, and ignoring any unsubscribing attempt.

Finally, controls related to this framework are very lightweight, especially when compared to cryptographic authentication schemes.

38 Spammers could increase the number of spam messages sent as consent requests or error messages, but since these should be easier to detect, at the end less spam messages could still be delivered to the mailbox. However, the ability of spammers to adapt to new situations must not be underestimated.

Security considerations

The framework is not expected to increase the security of email messages or delivery, besides what is the actual goal of the framework itself, that is to be able to reject undesired messages. As an example, the SMTP protocol will still be vulnerable to e.g. man-in-the-middle attacks, where the compromise of a (relay) MTA allows the attacker to collect tokens and thus send forged messages until the MTA is secured and tokens are invalidated. No content security is expected from this framework. Malware protection is mostly unaffected, or slightly increased at best. Spam controls are reduced for messages carrying proper tokens, as discussed later on. However, since the main effect of the framework is to discard some messages, caution must be put in possible denial of service effects/attacks. This will be also discussed.

First, it must be noted that this is not an authentication scheme. Tokens can be freely exchanged and disseminated by whoever has access to them, be it legitimately or not. The effectiveness in identifying the sender must not be overestimated. Tokens will every now and then be obtained spammers and other miscreants. It could be actually dangerous if too much trust were put in the effectiveness of the framework with respect to authenticity, since people could get used to trust the authenticity of messages carrying a proper token, falling prey e.g. of phishing attempts, should the phisher manage to obtain a valid token. Whenever message authentication is needed, an actual authentication scheme should be adopted in addition.

There is actually a subtle but relevant difference between an authentication scheme, e.g. based on electronic signatures, and this consent framework: while a signature proves that a message is authentic, with this scheme the only correct statement is that a message without the appropriate token is forged[39].

Tokens could end up in the hands of the wrong person in three ways:

- through legitimate channels, e.g. when the owner of a token is convinced to share it with the miscreant, when the token in collected through some registration form, or when a user answers to a consent request which was just sent in order to collect tokens[40];

- through a direct attack on the communication channel; an example may be pharming[41] attacks, where the attacker could manage to collect a large number of tokens. These tokens will be valid for addresses in the domain whose information has been faked, and for some sender addresses in domains that used the fake information for message delivery (e.g. because a X-Consent-new-token header is present in the message); the compromise of a relay MTA is another example of attack on the communication channel

- through accounts compromise, e.g. when a user's computer if successfully attacked.

Collected tokens can only be used to send messages to the addresses that created them. Whenever an account is compromised, it can only be used to send messages to addresses for which the account owns appropriate tokens (the account can still be used to send unlimited message to non-consent-enabled addresses). After a few undesired messages, the token will be revoked, severely limiting the consequences of account compromise, at least from a spam fighting perspective. When an account is compromised, tokens used by correspondents in order to communicate with that address are also compromised and need to be revoked and replaced too. Also, the use of a token in spam can help in detecting which account has been compromised.

---

39 If an address is associated with a sender token in the address book, then every message with that address as sender is expected to carry that sender token. However, other messages could carry that token too, either because the sender gave it away, or because it was compromised. Private keys of signatures can also be compromised, but the owner is expected to put an appropriate effort in avoiding the disclosure, while tokens are sent unprotected through the Internet and are supposed to be every now and then given away by correspondents

40 in this case, an MTA token is also compromised; when answering to consent request, it could be considered to use a different MTA token from other messages

41 An attack aiming at redirecting connections through DNS information manipulation, http://en.wikipedia.org/wiki/Pharming

A compromised account can be used to modify the related entries in the token database of its receiver's MTA, but this just permits to block or send messages to the already compromised account, so it doesn't add much from a security perspective (proper backups may as usual help in recovering the damaged database)

Should an MTA be compromised[42], the attacker would have access to its token database, where couples {address, token} are available. However, only addresses managed by the MTA are available, e.g. addresses whose messages would be already accessible to the attacker. No information is available on the addresses that can use these tokens to send messages, nor would such information be useful to the attacker in order to send messages to them. The attacker could monitor outgoing messages and collect tokens, but this doesn't add anything to what could be achieved by compromising the same MTA in absence of the consent framework; also, usage of these tokens would easily be linked to the compromised address and to the compromised MTA. As usual, revocation of tokens would avoid any additional consequence on addresses, besides the messages already received.

From a phishing perspective, access to a single token may permit to send a few messages to the receiver's address, and this may be enough for the attack to succeed. This is one of the many reasons, and a major one, to use different tokens for communications between a user and a correspondent, and for communications in the opposite direction. The reason why will be explained with reference to tokens collected through pharming.

Currently, pharming is performed mostly by forging DNS information about e-commerce or online banking sites, and providing this information to end users, e.g. home users[43]. The opposite, that is providing to the banks fake information on their clients, is less common and probably much more difficult, since it would require to provide forged information to online banking sites, which are much more careful than home users from a security perspective. With the current pharming practices, miscreants could only collect tokens that are used for communications between the end user and e.g. its bank, and not tokens used for communications from the bank to the end user, which are much more useful for phishing attempts. This protection would be lost if the same token were used for communications in both directions.

While tokens should be pseudo-random numbers, it can be expected that they won't be random in many situations, e.g. when a user needs to remember some. This opens to the opportunity of dictionary attacks by trying to send messages to the receiver's MTA. While this is feasible, a successful attack would provide a single token, that can be used to send a limited number of messages to a single address. Provided that the framework is not improperly used as an authentication mechanism, this result is not worth the effort of a dictionary attack.

Since tokens don't carry any information on the sender, they don't need any special treatment e.g. when crossing firewalls, differently from other headers (e.g. Received: headers) that may disclose information on the internal infrastructure.

Let's now consider how to deal with messages that carry a valid consent token but are found to carry some malware, e.g. a virus. The use of a valid consent token would suggest that the sender address in the envelope is not forged, and thus an error message could be safely sent to it. However, since accounts can be compromised, the token could have been stolen and be used my the malware itself. The malware could then try to use the receiver as reflector in order to spread itself or spam messages to other addresses[44]. As a consequence, proper management of the message would be to inform the receiver that the token has been compromised and used to spread malware: the receiver knows the proper association between token and address[45]. Also, the token could be automatically

42 Note that this would be a severe problem for the organization owning that MTA, so token compromise would be probably a marginal problem.

43 Usually by compromising the user's name resolution tools.

44 This is the behaviour that should actually be expected by consent-aware malware.

45 Note that the receiver's MTA has no information on associations between tokens and receiver's correspondents. The

revoked[46], so that no other infected messages will be accepted. It will be up to the receiver to send a new token to the compromised correspondent, or even to re-enable the original token, if and when communication is still desired. A message informing the sender about the problem can still be sent by the receiver, since it uses a different token. Since the sender in the envelope could have been forged by the malware, the decision by the MTA to send an delivery notification failure message is not influenced by the consent framework[47].

An MTA token will be available to every MTA and user receiving the message; these MTAs may abuse of the token by inserting it e.g. in a fake or spam message. However, these messages may be easily recognized by the address owner, and once such a message is received, the token can be promptly revoked, with the limited damage of not being able to receive some error messages. Note that a compromised MTA can cause much more damage than stealing consent tokens, and could do so even without the consent framework, so this issue is probably marginal in the real world.

The X-Consent-New-Token header must be handled with care. The optional token must be always present if available (usually, unless the header is sent in the answer to a consent request). Also, the receiver must check its presence and correctness before replacing the token for the address.

Also, a message carrying a X-Consent-New-Token could get lost, so that the sender may think that the receiver switched to the new token, while this in fact didn't happen. In this case, sender and receiver must resynchronize with a new consent request[48] or by using the offline channel they used for the first contact. Note that intercepting a message, e.g. with a pharming attack, doesn't open to this kind of denial-of-service, since the attack requires tokens both from A to B (to deliver the message) and from B to A (to be included in the X-Consent-New-token header). Intercepting messages in both directions requires a much more powerful attack than pharming; should this happen, denial-of-service problems are probably marginal. Also, these problems show that tokens should only be replaced if the framework requires it or if a relevant problem has been detected. Unneeded token replacements expose tokens and open up to problems without any real benefit.

Privacy and anonymity

The framework should score well against most privacy and anonymity issues. A token doesn't carry any information neither on the sender nor on the receiver, and can be provided without disclosing information on whoever else can use it. No useful information can be collected from a token database per se, e.g. on an MTA. Tokens are only associated to correspondent identities in the user's address book and in the message, where they are associated to the message receiver. If messages are forwarded, or addresses are translated or expanded, the sender doesn't need to have any information on the final recipient. Two users can communicate with different identities using different tokens, so that neither knows that different identities match to a single person. This only requires that tokens are selected based on both the sender and the receiver identities[49]. A similar mechanism is already implemented by some MUAs, which select a proper sender address when answering to a message, selecting the same address the message was sent to. Tokens are not protected at the same level of the body of the message (e.g. they are not encrypted by S/MIME) but are protected as other message headers, which should be enough whenever tools are used in order to protect the messages from traffic analysis, e.g. in onion routing[50].

---

owner of the token could have given it to somebody else, who is the one with the infected host. However, the above considerations are still valid.

46 Note that this would require the antivirus component of the MTA to be able to write in the token database. Automated invalidation of tokens in general increases the risk of denial-of-service due to false positives.

47 There seem to be some discussion about how to deal with messages rejected because of carrying malware, and this framework shouldn't add anything to that discussion.

48 Consent requests from an address that is in the address book of the receiver could be highlighted or given a special treatment from the MUA, the same way some MUAs can currently whitelist these addresses w.r.t. spam controls.

49 This may require additional support on the MUA and in the address book.

50 http://en.wikipedia.org/wiki/Onion_routing

Implementation issues

A key point in the framework is that it enables the receiver to preemptively set up a consent policy on the receiver's MTA, so that undesired messages can be rejected instead of bounced (or accepted and then deleted).

For MTAs, the most relevant implementation requirement is the per-address token database, and the ability to manage messages where just some of the receiver addresses are consent enabled.

When looking at the receiver's MTA as a whole, the token database shouldn't be a big problem. Currently, most users have at most some hundreds of correspondents, which means the same number of entries in the token database. This won't make a big database when compared e.g. to mailboxes or the message queue. Also, searches in the database should be fast and efficient, given that there is a just a primary key, and that no complex searches are required. Workload for the MTA should be small when compared to the workload required by many common antispam controls.

Many current antispam tools, designed to be integrated in the MTA software, could be easily adapted so that messages are searched for X-Consent tokens, and lookups in the token database are performed. As a consequence, there would be no need to modify the MTA code in order to support the token database. The same tools are already able to start controls and reject messages based on the message envelope.

Also, database updates should be very limited, given the experience with tools with similar problems [4]. As said, token invalidation and renewal should be limited to what is strictly required. For many users, most activity should be actually related to the MTA-token updates.

A more complex activity may be the management of messages with multiple receivers, especially when the message need to be passed to an MTA not implementing the consent framework. While no change seems to be required in the overall behaviour of an MTA, changes in the code may be needed in many parts of the MTA. An MTA should declare to implement the consent framework, by answering to an EHLO with the X-Consent extension, only if the consent framework is properly implemented in every involved component. Fortunately, not every component handling messages needs to be aware of the consent framework, since unless the list of recipient addresses in the envelope is modified, the consent framework headers are just part of the message and are not supposed to influence the message management in any way.

For many end-user stand-alone MUAs, as already discussed, the consent framework could be implemented as a plug-in. The most relevant issue may be the interface with the MTA token database(s), unless some sort of standard is adopted, be it formal or de-facto. However, there are many MUAs, including webmail interfaces, and many tools which can generate and submit messages. All these tools may need to implement at least partially the consent framework (some just for message submission) in order to be able to send messages to a consent enabled address. In some cases, a local proxy could adopted in order to deal with the consent framework for tools not implementing it (e.g. mailing lists software): the proxy would be configured as outgoing SMTP server for the MUA, and would provide the token management interface.

Attaching tokens to addresses (e.g. *user@domain.x.token.x*.token, where the second “token” is a fixed string and *x* is a delimiter string not appearing in the token) could be a way to deal with tokens in existing infrastructures handling email addresses but not prepared to deal with the consent framework. This way, tokens could be provided through forms, stored and communicated through the infrastructure. Only components that actually create messages would need to separate the token from the address and put the token in a X-Consent header, and component receiving messages should be able to change addresses in address database in order to deal with new tokens.

Mailing list management tools may present some more issues. As a user subscribes to a mailing list, the management tool must be able to send a consent request and manage tokens. A consent request is actually what most mailing lists management software currently send in response to a

subscription request, in order to verify the consent of the subscribed address, so the framework is not adding anything new in this perspective, besides some format constraints. However, when sending messages, mailing list tools may be required to add a lot of X-Consent-token headers, depending on the number of receivers each message is sent to. This may cause some workload to MTAs that need to relay messages to subsets of the receivers.

All in all, a widespread implementation of the framework may need a much more pervasive implementation than most current antispam tools, whose implementation can be limited to SMTP servers and maybe DNS records. However, standards have already been proposed that require to be implemented in many components, e.g. RFC 4405 experimental series [8][13][14][15]

Another relevant issue are secondary/backup mailservers. They may be unaware of the consent framework, thus expecting the primary MTA to deal with the issue whenever messages are delivered to it. Or, they may try to early discard undesired messages or even have access to user mailboxes instead. In these cases they will need a properly synchronized token database[51]. These servers may be hosted by a different organization, and use a different technology for the MTA. As a consequence, proper synchronization between the token databases may be difficult. Again, standardization of the management interface may greatly reduce the difficulties.

Synchronization between databases may be temporarily lost. This means that the token database of the secondary mailserver may miss some tokens or include some tokens that have been removed from the original database. The first problem can be solved if address owners are required to insert an adequate number of pre-authorized tokens in the database beforehand[52], and then just start using them when a new token is needed. This way, the database of the secondary mailserver will hold all the tokens actually used, since during the (hopefully) short time the database cannot be synchronized, address owners will just enable existing tokens even while adding new ones. Messages carrying tokens that should have been revoked will still be accepted, since they have not been removed from the secondary database. Messages can be rejected later if they are then queued through the primary mailserver, or can be delivered to the user mailboxes, until the database is updated. Note that in this case, errors for bounced messages could be actually backscatter to forged addresses, since the token could have been revoked because it was compromised and used for spam. This doesn't add much to other synchronization problems that may occur between the receiver's MTA and its secondaries, e.g. on active addresses. Also, synchronization problems between servers could hardly be anticipated or caused by spammers, so the actual impact would be probably very limited.

A consent-aware SMTP client should try to recognize the consent token in the message and use a SMTP command (e.g. X-CONSENT-RCPT-TOKEN) to notify the SMTP server that it will supply the tokens as an option in the RCPT TO: command, or will notify in the same manner that the message is a consent request. The SMTP server could then check the validity of the token before the body of the message starts being delivered, accepting it and thus avoiding to even start inspecting the body for spam or headers, or rejecting it immediately selectively based on the recipient. This would greatly increase the effectiveness of the framework, especially in the management of messages with many recipients. Should the SMTP client be unable to detect the X-Consent-token header in the message, e.g. because non being consent-aware, the message could just be handled as usual (that is, the SMTP server searches for the headers).

Deployment of the framework

This framework cannot be expected to be implemented at once and on the whole Internet. While its effectiveness is best with a widespread implementation, two common issues are to be considered:

51 Access to a single token database in probably a bad idea, since whenever the backup mailserver is requested to receive messages, there may be some communication problems that may also deny access to the token database

52 Unused MTA tokens should also be added, which will be used in the future and will already have a proper validity set.

- the effectiveness of the framework while deployed on a limited, or even minimal subset of the MTAs and MUAs
- the interest of users and service providers in such a limited deployment

While a minimal subset of the Internet implements the framework, address owners cannot expect to properly take advantage of the framework with all their correspondents: rejecting messages that are not carrying a consent token would turn out in rejection of most legitimate messages. As a result, users won't see much benefit in enabling the framework. Also, support from most widespread MUAs would be required. This support could be implemented in the form of a plug-in, or by a local proxy acting as a submission agent, but should be implemented by every MUA involved in the framework anyway.

This could suggest that there is no way for the framework to get adopted. However, the framework may provide an opportunity for new services for closed user groups.

Mailboxes protected against harassment could be created even with a limited adoption of the framework, and with no need for a separate infrastructure. Consent requests would not be delivered to these mailboxes. Interested organizations could set up an overlay network of complying MTAs that protect children mailboxes, and provide the appropriate plug-ins for MUAs to both children and parents[53]. Tokens would be only exchanged by hand with trusted people (the interface with the MTA database could even be managed by parents). Also in this case, the ability to track any misuse of tokens and the ability to revoke them would be of great benefit[54].

People could also choose to adopt two different addresses; a consent-enabled address could be reserved for “clean” communications with trusted parties. These could include banks and partners, which could be interested in an additional protection layer. Generic phishing messages wouldn't reach that address, even if the address is disclosed.

While receivers would be the ones that could have the greatest benefit from a widely deployed consent framework, some senders could be the ones that would benefit from early deployment. A key mechanism that could help is whitelisting of domains implementing the consent framework. Messages received from a domain that is known to implement the consent framework should be handled consequently: consent enabled addresses should expect to receive messages with proper X-consent headers from these domains. Information on consent framework implementation could be put in a DNS TXT record[55] for the domain, since domains would have no benefit by stating to implement the consent framework while they didn't[56]. Consider now a domain that has been targeted by phishing, e.g. an online banking or e-commerce site. By enabling the consent framework[57], it would provide to its users a mean to reduce the risks of this kind of attacks. Its users in turn would be encouraged in the adoption of the framework, in order to benefit from this opportunity. The same could hold true for a company that is willing to reduce the misuse of its domain in address spoofing[58].

---

53 Parents with consent enabled MUAs could in turn be interested in consent also in other contexts.

54 In this case, messages with a valid token but with a sender address that differs from the one the token was given to, could also be considered anomalies and could recorded for parent inspection.

55 Note that the only information provided is whether the consent framework is enabled or not, so a descriptive record like the TXT should suffice.

56 Domain managers could only damage their own domain by putting fake information in a DNS record, since their messages could be discarded, so this kind of information should be published with due care. However, DNS information is not secure: records pertaining the implementation of the consent framework could be forged. Adding a record stating that consent is implemented when it's not would cause a denial of service, as messages would be discarded by MTAs using the forged information. However, should someone be able to forge DNS information for a domain, more dangerous and hard-to-detect attacks were possible.

57 Initial token exchange could be part of the confirmation and password management messages that are commonly exchanged at the registration.

58 Note that signed messages would provide a stronger protection against phishing. However, end users would need to deal with validation of senders and handle invalid signatures, and may fail to detect forged messages anyway. With the consent framework, messages without proper tokens are not delivered at all, and all in all it seem easier to

So, while a widespread deployment of the framework would provide some additional protection against spam, a limited adoption could mostly help against phishing and similar threats.

For whitelisted domains, consent aware MTAs could start reducing their controls when delivering messages to consent enabled addresses (e.g. avoid antispam controls on messages with valid tokens), and this in turn could increase the interest for ISP in the adoption of the framework in their MTAs.

In the early stages of deployment, implementation could be limited to MUAs, without involving the receiver's MTA. Messages would be delivered to the user mailbox regardless of X-Consent headers, but the MUA could still detect the presence or absence of valid tokens, use this information to rank messages as malware or phishing, and send error messages to request the use of consent request messages and of the framework in general[59]. While this could be feasible, receivers adopting this policy could actually see little benefit for the effort, and it could be hard to convince their correspondents to adopt the framework.

Also, in the beginning, consent-enabled addresses could accept messages which conform to consent request requirements (short text only messages) even if not carrying any consent header. This would increase the interest of receivers in the framework, especially those users that already have more that one email address in order to cope with spam and privacy issues: a consent enabled address would provide a very “clean” address to share with close friends. Note however that correspondents would need to be aware of this behavior, since it could hardly be explained in an error message to an unaware correspondent.

While these considerations may help the adoption of the framework, the critical factor will be the actual interest of users in the ability to explicitly manage consent to communication.

Usability

The whole framework is meant to be almost fully automated. The only part that cannot be automated is token dissemination and revocation, and acceptance of consent requests. However, these are exactly the activities on which a receiver wants to keep control, at least in the assumptions of this scheme. Also, MUAs can help in a lot of activities, e.g. by presenting a consent request message format whenever a user tries to contact a new address, and by highlighting anomalies.

Since tokens dissemination should be managed by the user, this may imply some burden in many situations: tokens need to be exchanged whenever a user gives its address or receives an address for a correspondent, including subscription to services and mailing lists.

Also, since these tokens are hard to remember, the address book becomes a requirement. Whenever a user uses more that one MUA, it needs to keep the address books synchronized. Even if no new address has been added, some tokens may have changed[60]. Also, the address book must be personal: corporate directories may suffice for addresses, but tokens must be handled separately for each user.

With respect to internal email in big organizations, users may require to be able to send messages to their colleagues without any prior consent request, even to consent enabled addresses. Organizations should be able to whitelist their own domains and to deal with forged internal addresses through other means. Otherwise, this kind of problems can be dealt with by MUAs which may interact with an internal directory service (e.g. an LDAP server) which may provide a valid token for each internal address, to be used only by internal addresses for the first communication. The same token would also be added to databases of internal MTAs by some agent on the directory service itself.

---

understand by users than most email authentication schemes (and their failures).

59 This can only be a temporary solution and error messages cannot be sent automatically, since this would produce a lot of backscatter.

60 Note however that it is suggested that tokens don't change very often. Also, an updated token database for the MUA is mostly needed for sending messages, since messages are accepted based on the MTA database.

Conclusions

An extension for the SMTP protocol has been presented, and a framework for it to enable address owners to define a policy for messages rejection based on senders. The use for the ability to express consent is clear when looking at how other communication tools are used, and at current legislation about privacy and spam in many countries. Most recent communication protocols and frameworks provide some way for consent expression. This feature is missing in the current email delivery framework.

This proposed framework is designed so that it can take advantage of the existing protocols and practices. Current mail management is not influenced by the framework for users and domains not willing to implement or deploy it, unless they need to send messages to consent-enabled addresses.

The proposed framework can coexist and cooperate with existing tools in order to increase the control on messages delivered in protected mailboxes.

While the framework could provide the greatest benefits to participants if widely deployed, even with a limited deployment there can be benefits in specific cases and for some sets of users. The adoption by these groups could ease the deployment also in other cases. However, such a framework would reach a widespread deployment only if end users will show to be interested in the ability to explicitly manage consent to communication.

References

[0] *You might be an Anti-Spam Kook If...* http://www.rhyolite.com/anti-spam/you-might-be.html

[1] *DIRECTIVE 2002/58/EC* of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)

[2] *RFC 5321: Simple Mail Transfer Protocol*, October 2008

[3] *RFC 5322: Internet Message Format*, October 2008

[4] *The Effectiveness of Whitelisting: a User-Study* David Erickson, Martin Casado and Nick McKeown, Fourth Conference on Email and Anti-Spam, CEAS 2007

[5] *Countering Spam by Using Ham Passwords (Email Passwords)*, David A. Wheeler July 8, 2004 (Revised November 18, 2004), http://www.dwheeler.com/essays/spam-email-password.html

[6] *Shades of grey: On the effectiveness of reputation-based "blacklists"* Sinha, Sushant; Bailey, Michael; Jahanian, Farnam - MALWARE 2008. 3rd International Conference on Malicious and Unwanted Software

[7] *Email Forwarding Best Practices*, Messaging Anti Abuse Working Group Recommendation, 2008, http://www.maawg.org/about/publishedDocuments/MAAWG_Email_Forwarding_BP.pdf

[8] *RFC 4406: Sender ID: Authenticating E-Mail*, April 2006

[9] *RFC 4871: DomainKeys Identified Mail (DKIM) Signatures*, May 2007

[10] *RFC 2505: Anti-Spam Recommendations for SMTP MTAs*, February 1999

[11] *Consent Framework for Fighting Spam v0.3*, October 2003, http://www.shaftek.org/publications/asrg-consent-framework.html

[12] *RFC 4409: Message Submission for Mail*, April 2006

[13] *RFC 4405: SMTP Service Extension for Indicating the Responsible Submitter of an E-Mail*

Message, April 2006

[14] *RFC 4407: Purported Responsible Address in E-Mail Messages*, April 2006

[15] *RFC 4408: Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*, April 2006

[16] *RFC 4870: DomainKeys Identified Mail (DKIM) Signatures*, May 2007