„Spam Salt“ aka „Message Salt“

An invention against email abuse (Spam), introducing an email sender authentication mechanism.

Author: Kai Engert, kaie at redhat dot com or kaie at kuix dot de
For updates to this document, please check http://kuix.de/spamsalt

Invention conceived 2010-04-29.
Document version 3, first public version, 2010-09-27.

Foreword:

I'm publishing this invention in the hope that it makes sense and can contribute to solve the „Spam“ problem, i.e. the problem of receiving lots of unsolicited email. Hopefully there are no fundamental flaws in the invention that make it pointless. If you do find a flaw, please let me know.

If the ideas in this invention receive positive feedback, the next steps are to state the protocols in this invention more precisely, implement a server software that provides the new functionality and enhance some email client applications to support it. However, this may happen in any pace, because the approach presented here can coexist with any existing email infrastructure.

It will still be possible to use current email transport servers and client applications. For server side support, it will be sufficient to deploy server software that runs in parallel to existing email server infrastructure.

Description of the invention:

For each email user account, in addition to the traditional password, an additional secret value is introduced, called salt key.
Each user's salt key is managed by the email provider.
The salt key is known to both the user and the email provider.
The salt key is usually stable over time, but will be changed after it get's abused and the abuse is detected.

Each time a message is sent, the sending email agent calculates a message salt hash value, also known as the outer hash value.
For calculation the value an hash function is used, such as SHA-256.
The input to the hash function is the combination of:

- sender's FROM email address
- REPLY-TO email address
- all TO and CC email addresses (BCC is ignored)
- the complete DATE added by the user agent
- the sender's secret salt key
- an inner hashsum based on message subject and message body

The inner hash value is used to represent the message subject and the message body. The same hash function is used to calculate both outer hash value and inner hash value.

When calculating the inner hash value, it must be asserted that both sender representation and receipient representation of an email message will produce the identical hash values. While in

transport, email transport agents may change portions of the MIME encoding, e.g. add whitespace or add quoted-printable encoding for text portions. The input to the inner hash value should be normalized, such as undoing quoted-printable encoding and potentially ignoring any white space, or normalizing each sequence of multiple whitespace characters to a single space character.
Before sending the email, the sending agent adds one more email header line(s) to the message:'

```
X-MessageSalt-Hash-SHA-256: \
      12991aca22970ac05f2e8a27de16afdcd41780c3be66f9cc09911604e7b420a6
```

This new header will be ignored by all E-Mail transport and processing software that doesn't support the message salt system, and is therefore not causing any problems for older software.

In order to offer verification of the correctness of message salt hashes, each email provider may operate a Message-Salt-Server that responds to verification queries.

Similar to 'MX' (mail exchange) records in the DNS (domain name system), that point to the mail transport agents for a domain, a new type of DNS record (e.g. 'MSALT') should be introduced, that lists the verification server for each domain, thereby making the message salt verification servers discoverable.

When an email is received by an email agent (server or client application), the agent can choose to verify the correctness of the hash. It can send a verification request to a verification server, using a client-server protocol (not delayed).

The contents of the verification request are:

- sender's FROM email address
- REPLY-TO email address
- all TO and CC email addresses
- the complete DATE added by the user agent
- an inner hash value of the message subject and message body (see above)
- the message salt (outer) hash value found in the `X-MessageSalt-Hash-SHA-256` header

The message verification server that is responsible for the email sender's domain will lookup the account data of the local email user, lookup the sender's secret salt key, combine it with the data in the verification request and perform the calculation of the message salt hash value. Then it will compare the calculated value with the value received in the verification request. If the values are identical, the verification server will then return success result to the client that requested the verification. If they are different, it will return a failure result.

The connection to the verification server should use an SSL connection (or another secure transport), and both parties should require that the other party presents a a valid X.509 server certificate (or use some other form of secure authentication). This helps to ensure that an attacker can not use MITM / DNS attacks to pretend false positive verification results.

If the verification request produced a successful result, the recipient (or transporting agent) of an email knows that the combination of {sender, recipients, date, subject, body} has really been produced by the owner of the sender's email account, unless that email account has been compromised, and the compromise has not yet been detected. (This is based on the reasonal assumption that administrators of an email provider will not abuse the secret salt keys of their local users.)

An email receiving client can use the verification result for filtering. All emails that can't be verified

can be sorted as potential spam (it may be spam, or from compromised accounts, or from senders who use legacy software). As the Message Salt mechanism gets adopted more and more, the percentage of email without message salt protection will decrease.

(Why is it necessary to include the message subject and body in the hash value calculation? If the message subject or body were excluded, an attacker would be able read the contents of a compromised mailbox, read all messages, find pairs of {sender, recipients, date}, and could send spam messages to each group of recipients. The spam messages could contain an arbitrary message body, with a forged sender and date, the stolen hash value, and recipients would accept it as authenticated.)

In order to avoid denial of service attacks against message verification servers, the following restriction is proposed:

- email clients shouldn't talk directly to foreign verification servers. Instead, an email client talks to the verification server of the user's own email provider, and authenticates using the regular email account credentials.
- this makes it possible to avoid SSL (or the secure transport) on a recipient client system
- the recipient provider's verification server then talks to the sender provider's verification server using SSL (or another secure transport). When doing so, both verification servers identify each other using SSL client authentication and SSL server authentication (or another secure bidirectional authentication mechanism).

(Regarding the use of Certificates: It shouldn't be necessary to purchase server certificates for message servers from commercial Certificate Authorities. Each NIC (Network Information Centre for Domain Names) could issue its own root CA certificate and issue free verification server certificates (restricted to e.g. msaltverify.domainname.tld). The verification server software could ship with those root CA certificates embedded and trust them. This detail of the proposal is meant to simplify worldwide adoption of this new Message Salt mechanism, as it would eliminate the need to have domain operators invest money for purchasing certificates.)

The above restriction has additional benefits:

- Proxying message verification servers (that operate as a client talking to another verification server) may cache requests and responses
- Verification servers may use verification requests that result in failure results as indication for abuse. If several failures are seen for a sender address, that account might have been compromised (although an attacker will most likely omit hash salt values in spam messages). However, it may be more appropriate to rely on spam reports from humans for more reliable compromise detection.
- If an attacker is able to read the secret salt key of an email account, the attacker can send spam that will have correct hash values and the spam messages will get accepted by recipients. However, upgraded email software may have a feature that allows a user to report spam messages to the original sender's verification server. If the number of spam reports for an account is higher than a configurable threshold value, the account can automatically be marked as compromised. As a reaction to the detected compromise, the email provider (i.e. the verification server) can automatically assign a new secret key salt, disable the user account, and inform the email account holder (who will not be able to send further salted messages until the situation is clarified and the account gets secured, passwords are changed, email client software updated to use the new salt key, etc.).
- Increased privacy for recipients. In many scenarios the sender is unable to spy the IP address of arbitrary recipients by sending an email, because the verification requests will originate from the receipient's verification server.

Also, the Message Salt system will make it possible to use blacklists and/or whitelists more reliably, because for messages having a valid salt hash value the sender is known to be correct (unless an account has been compromised).

The hope is, when using this invention widely, compromised (or new) emails accounts can be used for sending spam only for a very short amount of time, and it will affect only a few recipients. Once a spam message has been reported several times and the sender account has been marked as compromised, additional recipients will have the benefit that their email agent will automatically filter all messages failing the verification as known spam.

The Message Salt system can be deployed incrementally. Email domains that don't yet support the system simply don't get the benefits yet, but they will still be able to send email to, and receive email from any other email domain, as long as the other domains don't chose to require the presence of valid message salt hash values.

Using the message salt system, it's still possible to send email to any recipient, even if there has been no contact between sender and recipient previously.

The message salt system doesn't make automatic decisions which messages are spam and which messages not, it ratherenables sender authentication, which is an important first step towards spam protection. Past attempts to introduce sender authentication include the use of digital certificates, but require users to make personal keys on their own, and despite having been available for a long time, such applications are not being used by the mass of Internet users. The advantage of the message salt system is that the user doesn't need to manage the secret salt key, other than using it as a configuration parameter when configuring an email account in an email application.

As soon as a recipient receives a spam message, either from a compromised account, or from an account that was created with the sole intention to send spam, the recipient can report it. There may be automatic mechanisms installed on verification servers that automatically put an email account into the „potentially compromised“ state. This may limit sending of spam to just a few successfully delivered spam messages per compromised/spam account.

Once an account has been put into the „potentially compromised“ state, and the salt has been changed, future verification requests may return „failure“, despite the fact that the salt hash was correctly created at sender time, and email recipients agents can automatically filter such messages into a „potentially spam“ folder.

Each party involved in email transport may query their own message verification server for the status of email messages. An email transport agent, routing email on the way from the sender to the recipient, may verify emails and chose to route only emails messages that can be verified, and reject all messages that can't be verified. However, such a configuration probably wouldn't make a lot of sense until the message salt system has been deployed widely.

Notes on generalization:

The use of SSL in this document is an example for a secure transport. SSL is simply a helper protocol for implementing this invention. Other secure transport mechanisms, that also provide authentication, could be used instead of SSL.

The use of SHA-256 in this document is an example of a hash algorithm. An implementation may use other hash algorithms, and use header used in emails may be of the general form X-MessageSalt-Hash-{name-of-hash-algorithm}. This may be necessary in the future when SHA-256 is no longer secure and a different one must be used.

Elements of the Message Salt system:

Verification server: Software must be written that implements the Message salt verification server functionality. This may be straightforward to do. The software must have access to email accounts, optionally being able to lock accounts out of sending. It must store a „good“ or „potentially compromised“ state for each email account. It must implement the Message-Salt-Verification-Server-Protocol (yet to be defined in detail) and respond to verification requests. It must allow connections from email users, that are able to authenticate using one of the local accounts, and implement the client side of verification requests to other verification servers. The verification server must implement the algorithm to calculate the hash values.

Message-Salt-Verification-Server-Protocol: The protocol that uses a secure transport, does bidirectional authentication, receives all data necessary for calculating a hash salt value, and returns a result. As an example, this could be implemented by using a simple HTTP style request and response dialog, running over SSL with bidirectionaly SSL authentication.

E-Mail clients (either webmail services or desktop applications) should be enhanced to accept configuration of the secret salt key, calculate the inner and outer hash values, and add the necessary email header.

Salt-Key or Message-Salt-Key: A secret value known only by email account owner and the provider of the specific email account.

Hash-Value or Message-Salt-Hash-Value: A checksum, which was calculated by using the contents of a particular email message and the secret Hash-Key.