Loaf

About Loaf

A technology called Loaf (List of all friends) was presented on Slashdot (article and comments).
The idea is that users publish the e-mail addresses from their contact list in a cryptographically secure manner (SHA-1/bloom filter) and collect these databases from their trusted friends.
When an e-mail from a friend arrives, the client automatically saves the LOAF database for that user.
When an e-mail from an un-trusted party arrives, the client searches through the saved databases to see whether one of the user's "friends" communicates with the un-trusted sender. If a match is found, then the message is accepted.

Security problems

The authors discuss a few attacks on this method, but ignore some other vulnerabilities:

1. Forged headers: If an attacker sends a message forged to be from a friend containing a LOAF database, the legitimate database will be overwritten. At the cost of runtime and database size, a history of friends' databases can be kept to protect against this attack. Since addresses are hashed using the bloom filter, there is no way to easily decipher the contents to check for authenticity or unauthorised alterations.
   2. Virus attack: An e-mail worm that collects e-mail addresses could use something similar to a Chinese menu attack (rotating the salts) to generate multiple bloom filters to send out with itself. As soon as the unlucky recipient of the virus gets one that is tagged as from a friend, they will have a database full of all of the other addresses where the virus is being sent. Then, all further virus e-mails will be accepted by Loaf.

Implementation issues and ideas

Loaf could be greatly improved through the use of trust, both of users and of the individual databases received. Among other things, it could be used to solve the "Marc Canter attack" discussed by the Loaf authors. It would be useful to tie in the expected false positive rate based on the density of the bloom filter. Trust can also be assigned based on how many users' databases the address exists in and the trusts of those databases. This could be tied in with something like AutoWhitelist.

As the authors suggest, outgoing mail would be run through a filter on the mail server where the address book would be maintained. This is a good idea, since you could also do other things like train your personalised text classifier (BayesInSpamAssassin).

Performance might degrade if you're using a large number of trusted users. For every e-mail address checked, SHA-1 hashes would need to be computed using every salt present in the database. I can't think of any algorithms off the top of my head that would be helpful for speeding up this search.

Discussion

My biggest issue with it is the concept of multi-kilobyte attachments on all (or most) of my outgoing mail. I also pointed out to the authors when it first came up 😉 that it needs to avoid the from-me-to-me attack, whereby a spammer simply uses the desired recipient address as the From address. Avoiding that is as simple as ensuring that the user's addresses aren't given any "whitelisted" points. --JustinMason.