Secure List Server: Mailman, PGP and S/MIME

Support for encryption and authentication for the GNU mailing list software

by

Joost van Baal

Januari 12, 2009

About this document

This document is published on
http://non-gnu.uvt.nl/mailman-pgp-smime/pgp-smime/talk/.

Copyright and license

Copyright © 2009 Joost van Baal, ad 1810 <joostvb-mailman-pgp-smime/a/mdcc.cx>

This document is free; you can redistribute it and/or modify it under the terms of the GNU GPL, either version 3 or any later one, see http://www.gnu.org/copyleft/gpl.html . There is NO WARRANTY.

Revision control

Maintained at http://bazaar.launchpad.net/~joostvb/mailman/2.1-pgp-smime/files

About the author

Joost van Baal is hacking on Mailman since 2005. Debian developer since 2000. Working on Lire, LogReport's log analyzer, since 2000. Free software advocate since 2001. Pugilist since 2007 and active as a DJ since 1995. Owner of ad 1810 since 2008. Joost works and lives in Eindhoven, The Netherlands. See http://mdcc.cx/.

Motto

Users and other ``soft'' factors (economy, psychology, sociology) are the reasons why security often fails; not errors in logic or in the application of mathematics. --Adam Shostack and Andrew Stewart, ``The New School of Information Security'', Pearson USA 2008

His dismissal notice stated that he was being removed from production on account of a situation of [...] thoughtfulness amid the general tempo of labour. "If we all would start thinking, who would get the work done?" --Андрей Платоно́в, ``The Foundation Pit'', 1930
(See also http://www.litencyc.com/php/sworks.php?rec=true&UID=14435)

Introduction

The Secure List Server, mailman-pgp-smime, is an effort to add support for encryption and authentication to Mailman, enabling groups of people to safely cooperate and communicate using email. The project currently is made possible by the NLnet foundation.

This article will start with a very short overview of the history of Mailman and the mailman-pgp-smime project. Some remarks will be made on how to install and configure the software, so that one can try it. Currently supported features

will be mentioned, as well as an overview of development plans. One will learn how to contribute to the project; an overview of the revision control system used will be given. Some remarks on the future of the patch will be made: will it be shipped with Mailman itself?

The reader is assumed to have some knowledge of Mailman, e.g. by being subscribed to a Mailman managed list and by administrating such a list. Furthermore, some knowledge of PGP and/or S/MIME is assumed.

GNU Mailman and other mailing list software

GNU Mailman is mailing list management software. It allows you to create and manage electronic mail mailing lists. It provides a web front-end for easy administration, both for list owners and list members. It supports digests, archiving, spam protection, bounce detection, Usenet gateways, and many more features. Mailman is licensed under the GNU GPL and is written in Python. Is is likely the most popular Open Source mailing list manager.

Other popular mailing list managers are (names of packages available with Debian GNU/Linux):

- sympa (written in Perl)
- mlmmj (relatively new, styled after the ezmlm mailing list manager)
- smartlist (based upon the procmail MDA)

Other alternatives are:

- minimalist (small and easy, no web ui)
- enemies-of-carlotta (another ezmlm-like one, new)
- ecartis (the free listserv)
- courier-mlm (part of Courier mail framework)

See http://popcon.debian.org/ for a comparison of the popularity of these packages withing Debian.

See the pictures popcon.png, retreived from http://qa.debian.org/popcon-png.php?packages=minimalist+mailman+smartlist+sympa+courier-mlm+enemies-of-carlotta+ecartis+ezmlm-idx+mlmmj&show_installed=on , as well as the picture popcon-non-mailman.png.

Popular mailing list managers not shipped with Debian are

- ezmlm-idx (a fork of the original ezmlm)
- phplist

Other ones worth mentioning are:

- listserv (not Open Source)
- majordomo (popular in early 1990ies, development stalled since 2000, the first popular mailing list software)

Mailman development was started in the late 1990ies by John Viega. The first release was in 1996. Barry Warsaw, who joined late 1990ies, currently leads the development. Mark Sapiro currently maintains the stable branches of the code; Tokio Kikuchi is another one of the main contributors. In total about 20 people have contributed substantially to the code. (And in total about 200 get explicitly thanked for their contributions to the project.)

Secure List Server

The Secure List Server, mailman-pgp-smime, is an addition to Mailman, enabling

groups of people to safely cooperate and communicate using email. The patch includes support for both RFC 2633 (S/MIME) and RFC 2440 (OpenPGP) email messages.

A post to a secure list will be distributed only if the PGP (or S/MIME) signature on the post is from one of the list members. For sending encrypted email, a list member encrypts to the public key of the list. The post will be decrypted and re-encrypted to the public keys of all list members.

The mailman-pgp-smime project has its roots in work by Stefan Schlott, probably from 2004. In 2005, this project was known as the SURFnet Secure List Server (mailman-ssls). SURFnet and Tilburg University made the project possible. Since 2008, the project is known as Secure List Server (mailman-pgp-smime) and made possible by the NLnet foundation.

Installing mailman-pgp-smime

As of 2009-01, the mailman-pgp-smime software is offered as a patch only. (Shipping a Debian and RPM package is planned.)

Patch and install

For installation, one has to download
both the original GNU Mailman source tarball as well as the mailman-pgp-smime patch. Once that's done, apply the patch:

```
% tar zxf mailman-2.1.11.tgz
% cd mailman-2.1.11
% zcat ../mailman-2.1.11-pgp-smime_2009-01-02.patch.gz | patch -p1
```

Now that the Mailman software is patched, continue following the instructions in the GNU Mailman Installation Manual. (Including something like:

```
# aptitude install python-dev apache2

# mkdir /opt/mailman
# chgrp list /opt/mailman
# chmod a+rx,g+ws /opt/mailman

# su - list
% ./confire --prefix=/opt/mailman --with-groupname=list --with-username=list
% make
% make install
% /opt/mailman/bin/check_perms -f
```

Configure webserver, MTA and Mailman

Configure webserver. This will include e.g.

```
# echo 'ScriptAlias /mailman/ /opt/mailman/cgi-bin/' > /etc/apache2/conf.d/mailman
```

Add FollowSymLinks to the Options-line for Directory "/usr/share/apache2/icons" in /etc/apache2/mods-enabled/alias.conf, and create symlinks from /usr/share/apache2/icons/ to the icons in /opt/mailman/icons/.

Configure your MTA, see the GNU Mailman Installation Manual.

Set up list "mailman":

```
% /opt/mailman/bin/newlist mailman
```

and configure it. Set up cronjobs:

```
# crontab -u list /opt/mailman/cron/crontab.in

Debian and SLS specific stuff
-----------------------------
Adjust the cronjobs: strip the python-option "-S".

% crontab -e

For pgp-smime:

# aptitude install python-gnupginterface

Work around python path issues:

% ln -s /var/lib/python-support/python2.5/GnuPGInterface.py \
   /opt/mailman/GnuPGInterface.py

Start Mailman
-------------
% /opt/mailman/bin/mailmanctl start


Create a PGP-list
-----------------
After creating a normal list called test-gpg, and having subscribed to it, run

$ gpg --gen-key
$ gpg --armor --export DEADBEEF
$ gpg --export-secret-keys --armor DEADBEEF

Upload public and secret listkeys using
https://your.web.server/mailman/admin/test-gpg/privacy/gpg

Upload your member key using
https://your.web.server/mailman/options/test-gpg/you@your.dom.ain


Check results
-------------
You can peek at current settings running:

% /opt/mailman/bin/config_list -o - test-gpg | grep \^gpg
% GNUPGHOME=/opt/mailman/lists/test-gpg/gpg gpg --list-keys

The PGP listkey is stored in /opt/mailman/lists/test-gpg/gpg/*.gpg (and in
config.pck as well).

If you'd like to quickly change some (gpg, smime) settings, run

% config_list -o - testlist >/tmp/dump
% vi /tmp/dump
% config_list -i /tmp/dump testlist


Create an S/MIME-list
---------------------
First create your own SSL CA:

$ /usr/lib/ssl/misc/CA.pl -newca

Create your member S/MIME keypair:

$ openssl genrsa -out test-member.key 2048

Create a Certificate Signing Request:
```

```
$ cat <<EOT >test-member.cfg
[ req ]
default_bits            = 2048
default_keyfile         = you-testlist-member.key
distinguished_name      = req_distinguished_name
attributes              = req_attributes
prompt                  = no

[ req_distinguished_name ]
C                       = NL
O                       = Yoyodyne
OU                      = Secure List Server project
CN                      = Joe Random Hacker (testlist member)
emailAddress            = you-testlist-member@your.dom.ain

[ req_attributes ]
EOT

$ openssl req -new -newhdr -config test-member.cfg -key test-member.key \
   -days 1000 -sha1 -verify -out newreq.pem

Sign our member key with our CA:

$ /usr/lib/ssl/misc/CA.pl -signreq

Configure our emailclient (mutt) to work with this CA and keypair:

$ smime_keys init
$ smime_keys add_root ~/.smime/cacert.pem
$ smime_keys add_chain ~/.smime/test-member.key ~/.smime/newcert.pem \
   ~/.smime/cacert.pem

Now create an S/MIME list called test-smime, and subscribe
you-testlist-member@your.dom.ain to it.

Create a keypair for the S/MIME list (use e.g.:

[ req ]
default_bits            = 2048
default_keyfile         = key.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
prompt                  = no

[ req_distinguished_name ]
C                       = NL
O                       = ad 1810
CN                      = Testlist SMIME
emailAddress            = test-smime@your.dom.ain

[ req_attributes ]

as list.cfg):

% openssl genrsa -out key.pem 2048
% openssl req -new -newhdr -config list.cfg -key key.pem -days 365 -sha1 \
   -verify -out list.csr

Sign this listkey.

Store this key:

% mkdir /opt/mailman/lists/test-smime/smime
```

Make sure permissions and ownership are:

drwxrwx--- 2 www-data list 138 okt 29 15:59 smime/

Move (or copy) key.pem, cert.pem and ca.pem (and optionally list.cfg and list.csr) to this directory: install the signed certificate as smime/list.crt, and install the root CA certificate as smime/cert.pem.

Make the lists' public key known to our emailclient:

$ smime_keys add_cert cert.pem

Upload the member .pem using
https://your.web.server/mailman/options/test-smime/you-testlist-member@your.dom.ain .

NB: for S/MIME lists, the listkey is not kept in config.pck; there's no interface (yet) for uploading the listkeypair via the webserver.

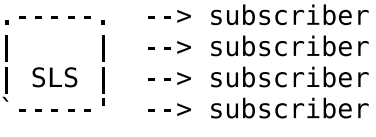
What can you do with it?

PGP and S/MIME offer Integrity and Authenticity (by signing messages) and Confidentiality (by encrypting messages). These are nice features, also when working with Mailing lists. However, traditionally achieving such functionality for lists means each subscriber would have to know and trust (or setup some trustpath to) each other subscriber. This means lots of work, and requires quite some clue for each subscriber.

The Secure List Server, mailman-pgp-smime, makes this easier. When using this software, each subscriber (optionally) has a personal keypair, and (optionally) there's a public key for each list. Managing trustpaths is fully delegated to the list administrator.

SLS caters lots of different security requirements: for each list, there are more than 100 ways to configure it (2 * 3 * 3 * 3 * 2), using 8 configuration settings {gpg,smime}_{distrib,post}_{encrypt,sign}. (For each list either all gpg_-settings should be set to No (such a list is called an S/MIME list) or all smime_-settings should be No (a PGP list).)

We'll give some example use cases. It's useful to realise the server works like this:



Example: Mailman vanilla

You can run the software without any PGP or S/MIME configuration. This way, the software works just like stock Mailman. Care has been taken in most circumstances the SLS-code even won't get executed.

Example: SLS light: gpg_distrib_encrypt, subscriber keys

Suppose you want to set up a list where some subscribers want some confidentiality: they want to receive posts while being protected from eavesdroppers within their own network. Suppose these subscribers know how to decrypt messages encrypted to their personal PGP public key. In such a setup, these subscribers should upload their public key. The list will be a PGP-list, option gpg_distrib_encrypt will be set to Yes; all other gpg_ and smime_-options will be set to No. No listkey is needed.

Example: SLS as anti-spam tool: smime_post_sign, subscriber keys
---
Suppose you want to make sure no spam ever gets posted to your list. (Allowing posting only for subscribers can get circumvented by spammers.) Suppose the subscribers know how to sent an S/MIME signed message. In such a setup, all subscribers should upload their public key. The list will be an S/MIME-list, option smime_post_sign will be set to Force; all other gpg_ and smime_-options will be set to No. No listkey is needed.

(One could also choose to use the listkey as anti-spam measure; allowing only encrypted posts. This way, subscribers don't need to fiddle with personal keys. It'll likely take a while before spammers start running encryption software. It would be possible though.)

Example: SLS for dissidents: smime_post_encrypt, list key

Suppose you want to set up a list where some people want some confidentiality: they want to be able to post while being protected (by S/MIME) from eavesdroppers within their own network. (Like a "Chinese dissident" scenario, a non-member posting anonymously.) This poster would need to have a copy of the lists S/MIME public key. The list will be an S/MIME list, option smime_post_encrypt will be set to Yes; all other gpg_ and smime_-options will be set to No. No subscriber keys are needed.

Example: full SLS

Suppose you want to set up a list where full integrity, authenticity and confidentiality is needed. Suppose the audience knows how to use PGP. In such a setup the list administrator should generate a PGP keypair for the list, and configure the list to use it. Each subscriber should get a copy of the lists' public key and import it to their keyring. Furthermore, all subscribers should upload their personal PGP public key.

The lists settings will be:

gpg_post_encrypt Force: only posts encrypted to the list key will get distributed.

gpg_post_sign Force: only posts with a valid subscriber key will get distributed.

gpg_distrib_encrypt Force: all posts will get encrypted to the subscribers key bfore being distributed.

gpg_distrib_sign Yes: all posts will get signed with the listkey before being distributed.

This way, the post is encrypted (so kept confidential) both when in transit from the poster to the server, as well as while in transit from the server to the various subscribers. Integrity and authenticity are guaranteed also by keeping the message signed while on the network.

Overview of settings

- gpg_post_encrypt (No, Yes, Force): Are postings which are encrypted with the GPG list key decrypted? Are subcribers forced to encrypt their posts? Such messages will get decrypted and (possibly) re-encrypted. A header "X-Mailman-SLS-decrypted: Yes" will get added to the messages.

- gpg_distrib_encrypt (No, Yes, Force): Are posts encrypted to the subscribers GPG public key before being distributed? Is such encryption (and uploading of a public key) mandatory?

- gpg_post_sign (No, Yes, Force): Should posts be GPG signed with an acknowledged subscriber key before being distributed? (Yes means: hold for

  approval, Force means: discard unsigned messages.)

- gpg_distrib_sign (No, Yes): Should the server sign messages before
  distributing?

Latest Changes

We give a summary of the changes since the patch was published by Stefan Schlott (2005-02).

Security was improved, thanks to suggestions made by Security Auditor Guus Sliepen:
- No longer allow a member to change an already set public key using the password authenticated web UI.
- In case a message was decrypted and should be held or discarded, forward only the headers to the listmaster, not the decrypted content.
- Emails with a valid signature of a known subscriber are now accepted only if the address in the From header matches one of the email addresses associated with the key. Since the original signature is removed before the mail is sent to the other subscribers, this did allow one subscriber to impersonate another subscriber or even an outsider.

Thanks to Stefan Schlott, a mailinglist is available for discussing development of the patch. The patch now is maintained using a public Version Control system (first darcs, now bzr at Launchpad). Some documentation got added, in README.PGP-SMIME.html, TODO.PGP-SMIME and NEWS.PGP-SMIME.

The patch got stepwise ported from upstream 2.1.5 to 2.1.11.

Support for PGP subkeys got added (contributed by Tonnerre Lombard). The patch now deals with both inline signatures and detached signatures. Signature-verification support (via new options {gpg,smime}_post_sign) as a moderation criterium got added.

The patch now supports S/MIME (next to PGP).

Development Plans

NLnet agreed to support the following future work:

Write and publish documentation	2009-01-15
Create a package of SLS	2009-03-01
Disseminate results	2009-03-01
Act upon auditors final report	2009-04-01
Try get SLS shipped w/ distros	2009-04-15

That is:

1) Writing documentation for users, for list admins, for site admins, as well as for developers.

2) Building and publishing both a Debian and an RPM package for SLS.

3) Disseminate the results by giving presentations: at CCC Ulm, Mon January 12th, and a lightning talk at Fosdem, Sun February 8th, 10h20, ULB Campus Solbosch, Brussels (http://fosdem.org/2009/node/164).

Guus Sliepen has performed a security audit; results are online at http://non-gnu.uvt.nl/mailman-pgp-smime/pgp-smime/audit.pdf. Guus is now performing a second and final audit, on the latest SLS release (mailman-2.1.11-pgp-smime_2009-01-02.patch.gz).

Next to the 3 mentioned jobs, NLnet agreed to support:

4) Act upon finding of Security Auditors final report.

5) Try to get Secure List Server shipped with Free Software distributions.

The last job consists of: Ask and help maintainers of Mailman packages for e.g. GNU/Linux distributions to include the patch. Work with the Debian Mailman package maintainer to try to get the patched Mailman shipped with Debian and Ubuntu. Next to Debian/Ubuntu, people within the Sabayon (http://www.sabayonlinux.org/) and Small Sister (http://smallsister.org/) [1] projects will get asked (and offered help) to include the patched Mailman system. NB: The decision on wether or not to include this patch is under control of the package maintainer (not the patch author).

Contributing to the project

If you'd like to contribute patches, check out the code using Bazaar:

```
 $ bzr branch lp:~joostvb/mailman/2.1-pgp-smime
 $ vi Mailman/GPGUtils.py
 $ bzr commit -m 'fixed all bugs'
 $ vi Mailman/Handlers/Moderate.py
 $ bzr commit -m 'added the missing feature'
 $ bzr send --output=/tmp/merge
 $ mutt -a /tmp/merge -s '[patch] bugfix, feature' \
    joostvb-mailman-pgp-smime/a/mdcc.cx </dev/null
```

See https://code.launchpad.net/~joostvb/mailman/2.1-pgp-smime for instructions. A fancy webinterface to this version control system is available at Launchpad's Bazaar page.

There is a (huge) TODO-list, at http://non-gnu.uvt.nl/mailman-pgp-smime/TODO.PGP-SMIME .

The future

Currently (2009-01-12) there are 3 Mailman branches: 2.1, 2.2 and 3. Up to now, only for 2.1 there have been stable releases. No fancy new stuff will be introduced in 2.1 or 2.2. All exciting new development will take place in the 3-branch.

Latest news for 2.1 and 2.2:

On Sunday January 11, Mark Sapiro released 2.1.12rc1: a bugfix and python 2.6 compatibility release.

On Jan 3, 2009, at 2:51 PM, Mark Sapiro wrote:
> I expect to ship the final 2.1.12 release by the end of January.
[...]
> After January, my focus will be on Mailman 2.2. I hope to be able to
> release a 2.2 beta before the end of March, 2009.

Latest news for 3:

On Sat, 3 Jan 2009 Barry Warsaw wrote:
> Released 3.0 alpha 2
[...]
> still an alpha snapshot and not suitable for production systems, functional
> enough to create mailing lists, add and remove members, send email from and to

> lists. [...]
> The web interface is still not functional, so for now you have to interact
> with Mailman via the command line.

It is not known when a stable Mailman 3 will get released. (As Free Software hackers say: that's up to _you_ !)

I've requested support for the project on 4 Mar 2008. At that time it seemed wisest to focus on patching 2.1. I would like to port the patch to 2.2, once there's a stable 2.2 release. However, I don't think the Mailman developers would like to ship a 2.2 including the patch. I haven't yet investigated about the feasability of porting the patch to 3. It for sure would be a very useful project! (And it might be easy: being pluggable is one of the design decisions for Mailman 3.)

Contact, questions

If you're interested in helping with the work, you might like to subscribe to the developer list for SLS: ssls-dev /a/ ulm.ccc.de. (Yes, that's sSls. Backwards compatibility :). (Thanks a lot to Stefan Schlott for hosting this list.)

If you'd like to contact the author directly, mail Joost van Baal on <joostvb-mailman-pgp-smime /a/ mdcc.cx>. I'm on IRC too: joostvb@{OFTC,freenode}.

Thanks

Jeroen Hoppenbrouwers (http://hoppie.nl/) for helping translating the Platonov quote. The Mailman community, for giving valuable feedback and making Mailman possible. Guus Sliepen, for guarding the patch's security. The NLnet foundation, for making the work on this patch possible.

Notes

[1]: The Small Sister Project aims to increase your privacy by delivering SmallMailServer and SmallMailClient, creating a privacy-friendly system where personal data is properly secured. Using Tor and GnupG, it enables (optionally anonymous) e-mail without the burden of data retention and eavesdropping.