An encrypted mailing list with Mailman, Exim and Gnupg on Debian GNU/Linux

July 16, 2001

1 Concept

The mailing list user (list) has a GPG key pair. Mails sent to the mailing list have to be encrypted with the list's public key. The mail is then decrypted on the server, and reencypted for sending to each member of the mailing list. To do that, the public key of each member has to be present in the keyring of the mailing list user (list).

2 Configuration

2.1 Mailman

No tweaking needed

2.2 GnuPG

Generate the key pair of the mail list user (see MAILMAN_UID in exim.conf) with gpg –gen-key. Share the public key of the list with members (gpg –armor –export)

Import the key of each member and sign it. This is necessary to use gpg in batch mode.

2.3 Exim

Modify the list_transport so that it looks like this:

```
list_transport:
driver = pipe
command = "/usr/bin/decrypt.py | MAILMAN_WRAP post ${lc:$local_part}"
current_directory = MAILMAN_HOME
home_directory = MAILMAN_HOME
user = MAILMAN_UID
group = MAILMAN_GID
use_shell
```

add a crypt_transport:

```
crypt_transport:
driver = pipe
command = "/usr/bin/recrypt.py $local_part@$domain" #command =
"/bin/cat > /tmp/mailoutput"
current_directory = /var/lib/mailman
home_directory = /var/lib/mailman
user = list
group = daemon
```

Dont touch the directors, but add a crypt_router before the smarthoste one:

```
crypt_router:
driver = domainlist
senders = list-admin@smailman
transport = crypt_transport
route_list = "* mailadm bydns_a"
```

As senders, put the mail of the list admin of the lists that has to be encrypted (I didn't test it with several lists yet).

2.4 Scripts

I wrote two Python scripts (these are my first python script ever, thanks your forgiveness if the code could have been better). The first is used for decryption of the mail received:

```
#!/usr/bin/python2

#Importing modules
#-----------------
import os
import sys
import tempfile


#Temporary files (decrypted mail body)
#---------------
tempfilename = tempfile.mktemp()
tempfile = open(tempfilename, 'w')

po=os.popen("".join( "%s %s" % ("/usr/bin/gpg --batch --armor
--decrypt > ", tempfilename )),'w')

firstline = sys.stdin.readline()        #Remove first line because
it's not relevant for SMTP
data = sys.stdin.readline()

#Extraction of the headers
#-------------------------
headers=""
while data !='\n':
```

```
        headers+=data
        data = sys.stdin.readline()
headers+="\n"

#The rest of the input is the message body
#-------------------------
data = sys.stdin.read()
po.write(data)
po.close()

#Print the decrypted mail to the standard output mail and remove
tempfiles
#-------------------------
body=open(tempfilename,'r')
body_var=body.read()
body.close()
print firstline + headers + body_var

os.remove(tempfilename)
```

and the second is used to encrypt the mail to send it to each member of the list:

```
#!/usr/bin/python2

#Importing modules
#-----------------
import os
import sys
import time
import tempfile


#Temporary files (file1=headers, file2=mail body)
#---------------
tempfilename = tempfile.mktemp()
tempfilename2 = tempfile.mktemp()
tempfile = open(tempfilename, 'w')
tempfile2 = open(tempfilename2, 'w')

po=os.popen("".join( "%s %s %s %s" % ("/usr/bin/gpg --batch --armor
--encrypt --recipient", sys.argv[1], ">> ", tempfilename2 )),'w')


data = sys.stdin.readline()     #Remove first line because it's
not relevant for SMTP
data = sys.stdin.readline()

#Extraction of the headers
#-------------------------
headers=""
```

```
while data !='\n':
        headers+=data
        data = sys.stdin.readline()
headers+="\n"
tempfile.writelines(headers)
tempfile.close()

#The rest of the input is the message body
#-------------------------
data = sys.stdin.readline()
po.write(data)
po.close()

#Send the reencrypted mail and remove tempfiles
#-------------------------
os.system("".join( "%s %s %s %s %s %s" % ( "/bin/cat ", tempfilename,
" ", tempfilename2, "| /usr/sbin/exim", sys.argv[1])))
os.remove(tempfilename)
os.remove(tempfilename2)
```

3 Problems

I use temporary files a lot. I didn't find a way to take the ouput of the popen and put it in a variable. This is especially a problem for the stage where the mail is not encrypted. I also use os.system to call gpg in the decrypt file. Maybe it would be better to use a python interface to gnupg?

If the mail received is not encrypted, problem.

Behaviour for mails with Attachments is not garanteed.