pEp Python Remailer Documentation

juga

Jan 29, 2021

Contents:

CHAPTER 1

pEp MixMailer prototype in Python

This project is a prototype of the pEp MixMailer project described in [Design] . It is composed by two subpackages:

1. `client`

It encrypts an Email message several times as described in and send it to the first remailer.

2. `remailer`

It decrypts an OpenPGP encrypted Email and sends it to the recipient(s) ‘in the decrypted Email. It is intented to run as Postfix pipe.

1.1 Documentation

See the `docs` directory.

1.2 Issues

Please, report any bug or feature request at the **‘issue tracker‘_**.

1.3 Contributing

See `CONTRIBUTING.md`

1.4 License

GPLv3

1.5 Authors

juga at riseup dot net

[Design] https://mixmailer_docs.codeberg.page/Design/Proposal1.html. .. _issue tracker: https://gitea.pep.foundation/pEp.foundation/mixmailer/issues

CHAPTER 2

Integration tests

To run the integration test, you will need docker installed.

2.1 Mixnet test

Then run `docker-compose up -d`

The test represent the following scenario:

Alice (A) want to send a message to Bob (B) routing it via N1, N2, N3.

For this test, the following docker containers are needed:

2.1.1 Alice

- has a keyring with the following keys: As, Ap, Bp, N3p, N2p, N1
- compose a message from Alice to Bob
- launch the mixnet client, the client will:
    1. encrypt the message for Bob
    2. encrypt the message for N3
    3. encrypt the message for N2
    4. encrypt the message for N1
    5. the final message is from A to N1
    6. send the message to N1

2.1.2 N1

- has a kerying with the keys: N1s, N1p, N2p, N3p
- receives the message from A to N1
- the mixnet remailer is triggered, which: 1. decrypts the message and obtain a message from A to N2 2. change the from header to be from N1? 3. send the message to N2

2.1.3 N2

- has a kerying with the keys: N2s, N2p, N1p, N3p
- receives the message from A (or N1?) to N2
- the mixnet remailer is triggered, which: 1. decrypts the message and obtain a message from A to N3 2. change the from header to be from N2? 3. send the message to N3

2.1.4 N3

- has a kerying with the keys: N3s, N3p, N1p, N2p
- receives the message from A (or N2?) to N3
- the mixnet remailer is triggered, which: 1. decrypts the message and obtain a message from A to B 2. change the from header to be from N3? 3. send the message to B

2.1.5 Bob

- has a kerying with the keys: Bs, Bp, Ap, N1p, N2p, N3p
- receives the message from A (or N3?) to B
- launch pep client, which: 1. decrypts the message and obtain a message from A to B

2.2 GNUnet GNS tests

To run the tests that use the GNUnet cli, run `docker/gnunet/test.sh`. To run the tests that use the GNUnet REST API, run `docker/gnunet/test_rest.sh`.

CHAPTER 3

Mixnet nodes registration in GNS

See some documentation about GNS [gnunet-gns]

V. proposed to use GNUnet GNS to register and retrieve mixnet nodes in the mixnet network:

```
Nodes should register, with an email address and a key, what is
already in pEp identity. The key should be in ASCII armor Clients
obtain the nodes from GNS.
```

3.1 GNUnet GNS limitations

3.1.1 No TLS

in the Web REST interface [rest-gns]

Because it's not possible to send/receive encrypted queries, every node would need to run their own GNUnet node locally and send/receive queries to it.

3.1.2 No authentication

in the Web REST interface [rest-gns]

3.1.3 Not a global system

With a global name system like DNS, all the client would have the same view of the network. An authority would still be needed for other reasons.

Because GNS is not global, authority(s) are needed.

3.1.4 Delegating GNS records

The authority(s) would neeed to add (to their `gnunet-namestore`) the keys of the mixnet GNS nodes to be able to resolve their records, ie. the authority(s) delegates the resolution of the node records to the nodes.

Likewise, each mixnet GNS nodes would need to add the key of the authority too, to be able to solve other GNS nodes, ie. each mixnet GNS node delegates the resolution of other node records to the authority.

See [delegation] for more details.

In the following diagram, there's an authority and two nodes, showing which records they'd store and which records they can query. Note that 0000, 1111, 2222 would be the GNS nodes keys, while AAAA and BBBB would be the mixnet nodes OpenPGP keys.

3.1.5 Node registration

The key(s) of the authority(s) would be hard-coded, so a mixnet GNS node can easily add the authority(s) key(s) to its zone, to delegate to the authority the resolution of other nodes.

But the authority needs to get the (new) node key too. How this can be done?

- The node could add its own key querying the Web REST API of the authority, but the request would not be encrypted
- nkls is investigating using GNUnet cadet of file sharing

3.1.6 Node discovery by the client

A mixnet client would need to know which are the mixnet nodes in the network. It would need to also have a local GNS node to query records to the authority.

Since it is only possible to ask a record to the authority, but it is not possible to ask all the records the authority knows about, the authority would need to store in a record (called eg. mixnet) a list of all the mixnet (GNS) nodes. After retrieving that list, the client could ask the authority about a node record.

A TXT record with the list of nodes could look like:

```
n1,n2
```

Which means we can then ask the authority about `n1` and `n2` records.

3.1.7 GNS records

And which would be the records that the nodes should register?

Each node should register their Email address and their OpenPGP key. There is no need to register the OpenPGP fingerprint, since it can be obtained from the key, and it does not add any extra security to transmit the fingerprint in the same "channel" the key is transmited. They probably should also register the mixnet "layer" in which they'll operate.

CG proposed to use the CERT record type [cert], but it would only allow to register the key, not the Email address and the layer.

We think TXT records are more suitable to store the triple. It'd have the form:

```
email=root@n1.pep.example;layer=1;opengpg=AAAA
```

TXT records are limited to 255 characters, and a OpenPGP key can be way longer than that. But it's possible to add several records with the same name, and they key can be splitted in several. When querying the TXT record, the key can be reconstructed concatening all the query results.

References:

CHAPTER 4

mixnet

4.1 mixnet package

4.1.1 Subpackages

mixnet.client package

Submodules

mixnet.client.cli module

mixnet.client.client module

mixnet.client.constants module

Module contents

mixnet.remailer package

Submodules

mixnet.remailer.cli module

mixnet.remailer.remailer module

Module contents

4.1.2 Submodules

4.1.3 mixnet.common module

4.1.4 mixnet.defaults module

4.1.5 mixnet.exceptions module

4.1.6 mixnet.settings module

4.1.7 Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Bibliography

[cert] https://git.gnunet.org/gnunet.git/tree/src/gnsrecord/plugin_gnsrecord_dns.c#n130

[gnunet-gns] https://gnunet.org/en/use.html#gns_cli

[rest-gns] https://rest.gnunet.org/

[delegation] https://docs.gnunet.org/handbook/gnunet.html#Adding-Links-to-Other-Zones