Security with Public Keys GnuPG

Guilherme S. Mazzariol
Bachelor of Science in Computer Science
Campinas/SP - Brazil
g138466@g.unicamp.br

I. Introduction

When we talk about security in communications, it's difficult to guarantee that all messages are safe from an attacker. The big question is: how to guarantee data integrity in messages and files? This question can be solved using GnuPG - GNU Privacy Guard encryption and signing tool - to protect all messages shared between two people. In this paper, we talk about the GnuPGP, how to use it, and which are the vulnerabilities of that tool.

II. The use of GnuPG

GnuPGP guarantees integrity in messages and files using digital signatures, encryption, compression and Radix-64 conversion.

The GnuPGP uses a private and a public key, that allow to encrypt/decrypt messages (which can be a text or a file) with a signature. To read the message you need the public key pair of the private key that encrypts the message. It's also possible to verify if the identity of the person who wrote the message is true, avoiding fakes.

The first action is to create your public key, using the following command:

```
$ gpg --gen-key
gpg (GnuPG) 1.4.16; Copyright (C) 2013 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
      0 = key does not expire
    <n>  = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
                "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
Real name: Guilherme Mazzariol
Email address: g138466@g.unicamp.br
Comment: SEGC - EXP04
You selected this USER-ID:
            "Guilherme Mazzariol (SEGC - EXP04) <g138466@g.unicamp.br>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.
can't connect to server: ec=255.16777215
gpg: problem with the agent - disabling agent use
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++
gpg: key 6E91697A marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid:  1 signed:  0 trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/6E91697A 2017-10-21
```

```
    Key fingerprint = DB46 4994 ACEE F85B 1D3D
51DA 0F2A FC20 6E91 697A
uid                  Guilherme Mazzariol (SEGC - EXP04)
<g138466@g.unicamp.br>
sub   2048R/E61BF562 2017-10-21
```

You should specify what kind of key you want (RSA and RSA, DSA, etc) and how long the key should be valid for, and you have to put some information about you.

The public-key algorithms used by GnuPGP are:

- RSA [2]
- Elgamal [3]
- DSA (Digital Signature Algorithm) [4]
- ECDSA (Elliptic Curve Digital Signature Algorithm) [4]
- ECDH (Eliptic Curve Diffie-Hellmann) [5]

To sign a message you have to use this command:

```
$ gpg –clearsign -o msg138466.txt.asc
msg138466.txt
You need a passphrase to unlock the secret key for
user: "Guilherme Mazzariol (SEGC - EXP04)
<g138466@g.unicamp.br>"
2048-bit RSA key, ID 6E91697A, created 2017-10-21
$ cat msg138466.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Guilherme Sbrolini Mazzariol
ra138466
Ciência da Computação
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAEBAgAGBQJZ7OuAAAoJEA8q/CBukWl6re4IA
Jbj+IJIKuTMHaioFpcAtHLA
09CJAjRpWLlR9PQJv4VtNDMWF2Dh4GQwv2LBmoS
wHVIDzVNR7JK+4wVWrsPnN3Yj
gzgWLCjEMF5z6V9GKNamhFkmtjUWL6DBv/0pUqAn
hgvsACZuhofAVILHHLiGc2vZ
jvfkPtol8DijO7YXfk5/sFtdtWSiqg1Us500DT3KSx1hZ
MHXKuD1vhABvd0Ro1te
6rVf1rpTaXHtVTwEcOZiZSbV4SWicjrn0sVPVN6OyeK
4jkWqqVkKT9HJhkiRpaqy
bO6Un/NAdLQ9mbUR6VJE6TCS+qiS0mEA9ynSeRisZ
dkrP6dw0Mox11en97sx/VA=
=H0fc
-----END PGP SIGNATURE-----
```

The most important thing is to create a "*Web of Trust*" with the other people that send/receive messages to and from you. To make that you need to exchange your Public-Key with your partners, sign their Public-Keys, export these Public-Keys-Signed, and send these for your partners. After that, you need to import from them your Public-Key again. For example:

```
$ gpg –import <public_key_of_a_partner>
$ gpg –edit-key <uid_of_a_key>
…..
Really sign? (y/N) y
…..
gpg> trust
…..
gpg> quit
Save changes? (y/N) y
$ gpg –armor –export
<uid_of_a_partner> > <pubring_Key>.gpg
$ gpg –import
<your_public_key_assign_by_partner>
```

This process guarantee that all messages sent to/received from your partners will be read only by authorized people.

III. Vulnerabilities and Possible Attacks

The main vulnerability is the storage location of your private key. To guarantee your communication is safe, protecting your private key is the most important task. If someone gets your private key, all data encrypted by it can be decrypted and signatures can be made in your name. If you lose your private key, you will no longer be able to decrypt documents encrypted for you in the future or in the past, and you will not be able to make signatures.

Another way to stay protected is to use a strong password (passphrase) when you are creating your keys (public and private) for encrypting/decrypting messages that will be sent to your partners.

Some programs can make a dictionary attack on your secret keyring directory. These dictionary attacks are very easy to write and create. so you should protect your "~/.gnupg/" directory very well.

Other possible attacks are:

A. A side-channel attack on GnuPG [6]

The attack extracts the secret decryption key within seconds, from a target computer located in an adjacent room, across a wall. The attack utilizes a single carefully chosen ciphertext, and tailored time frequency.

B. Implementation of Chosen-Cipher-text Attacks against PGP and GnuPG [7]

The lack of compression by PGP before encryption in a plaintext message can allow a single-ciphertext or two-ciphertext attack to determine the entire content of the message. The same can happen if the plaintext is already compressed file and the PGP doesn't further compress it.

C. Brute Force attacks [8]

A brute force attack involves the systematic searching through all possible keys, meaning the attacker tries every possible combination of key parameters. If a sufficient key length is used, trying all possible keys until the message can finally be decoded will most likely be doomed to failure, especially if the attacker has a relatively limited amount of time and/or resources.

IV. Conclusion

The GnuPG is a strong tool for sending and receiving messages with many partners, guaranteeing the integrity and authenticity of the messages, but trusting only in one tool is not a good way to stay protected. It is necessary to protect the private keys directory to avoid attacks in which they can be stolen. To create a "*Web of Trust*" with your partner's, some servers [9] can be used to check if a public key is from a specific person. Although it is necessary to use a strong algorithm in the private key to encrypt and protected all messages, taking some measures can mitigate a potential attack.

References

[1] OpenPGP Message Format https://tools.ietf.org/html/rfc4880

[2] Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 https://tools.ietf.org/html/rfc3447

[3] ElGamal, T., "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, Vol. 30, No. 4, pp. 469-472, 1985.

[4] Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC https://tools.ietf.org/html/rfc6605

[5] CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE) https://tools.ietf.org/html/rfc8037

[6] Genkin, Daniel, et al. "ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs." Cryptographers' Track at the RSA Conference. Springer, Cham, 2016.

[7] Jallad, Kahil, Jonathan Katz, and Bruce Schneier. "Implementation of chosen-ciphertext attacks against PGP and GnuPG." ISC. Vol. 2433. 2002.

[8] Attacks on PGP: A Users Perspective, https://www.sans.org/reading-room/whitepapers/vpns/attacks-pgp-users-perspective-1092
[9] MIT PGP Public Key Server, https://pgp.mit.edu