File encryption using OpenSSL and GnuPG (GPG/PGP) - part 2

Systems and Network Security

Contents

1 Notes

- Commands preceded with "$" imply that you should execute the command as a general user - not as root.
- Commands preceded with "#" imply that you should be working as root.
- Commands with more specific command lines (e.g. "RTR-GW>" or "mysql>") imply that you are executing commands on remote equipment, or within another program.

2 Exercises

We're going to communicate encrypted files to other participants in the class. For this to happen, we need exchange keys: unlike symmetric encryption where one needs to communicate the passphrase securely, we'll be using public keys provided by those we want to send files to.

2.1 Exporting your public key with GPG

The first step is to export your public key in a form that can be copied and imported by your colleagues.

Remember, to see which keys you have in your key ring, use the following command:

```
$ gpg --list-keys
```

The output will be a list of the keys contained in your keyring:

```
/home/sysadmY/.gnupg/pubring.gpg
-------------------------------
pub   2048R/C9FBE546 2013-07-30
uid                  Bob Bobson (Work address) <bob@bob.com>
sub   2048R/3BE8FE75 2013-07-30
```

Let's make a copy of our public key, and place it in a text file, ready to be sent to our friends and colleagues.

Note: a key can be addressed in one of several ways:

- using the fingerprint (here, C9FBE546) - this is the preferred method as it's garanteed to be unique
- using an email address, for example, bob@bob.com
- using part of the name ("bob") - but there can be many people called "bob" in your keyring!

We'll use the email address:

```
$ gpg --export -a --output myname-key.asc my@email.address
```

... of course, replace myname with your name (no spaces!) and replace me@email.address with the email address you entered when you created your key in the previous exercise. That is the email address you see when running "gpg –list-keys".

This will produce a file "myname-key.asc". You can view its contents using the "less" or "more" command:

```
$ less myname-key.asc
```

You will see something similar:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.11 (GNU/Linux)

mQENBFH3yPkBCAC2DHRIk6FXiovejBXlNgZdnapHqq7OwascfluD+qX7wDk93etX
4Y+GfSLC2vlC4tNlB9VEYgMAY61sQC31ZoY9vr5MfJnZPcN+3Byzx2G0d8lwnH0g
[...]
t1CdT+UawL0dWu4bkNHjC8qwBg0PedS/VBJqlJl4TWg832CXRYI=
-----END PGP PUBLIC KEY BLOCK-----
```

2.2 Exchanging keys

Let's stop for a second and think:

- To be able to *encrypt* files that only a certain person can decrypt, you will need a copy of THEIR public key
- Therefore, if someone else in the class wants to send you an encrypted file or a message, then will need a copy of YOUR public key

This is a three step approach:

1. Import the key of the person you wish to send encrypted files/messages to
2. Encrypt the message/file using the public key of that person (recipient)
3. Communicate (copy) the message to the recipient

So first, you will need to find another user in another group you want to send messages to, and agree with them that THEY should send YOU their public key by copying the file created in the previous step (myname-key.asc) to your server:

- "Hello sysadmX in groupY, I want to send you encrypted files - please copy your public key to MY account on MY server, I am sysadmA in groupZ"
- And inversely, if you want a group to send YOU messages, you will need to copy YOUR public key to their system (they will ask you)

NOTE NOTE NOTE: You don't have to send your key to the same person/group you received a key from. You can pick another group.

Once you've agreed who you will send messages to, and which other person you will receive messages from, proceed with the steps below:

2.2.1 Sending your key to another group (so they can encrypt files for you)

After you've exported your public key to a file using the "gpg –export" command from earlier, copy that file over to the server of the group who should encrypt files for you:

```
$ scp myname-key.asc sysadmY@10.10.x.10:
```

. . . where x is the number of the group you've agreed to send your key to, and sysadmX is the user in the group you're going to be conversing with.

If you are prompted to accept the SSH key ("Are you sure you want to continue. . . "), say yes.

2.2.2 Receiving (and importing) the key of another group (so you can encrypt

files for them)

Look in your homedir and check if a file "theirname-key.asc" (whatever their name is) exists - assuming you have asked them to copy their public key to your host!

If it is there, you can now import it into your keyring:

```
$ gpg --import theirname-key.asc
```

You will see:

```
gpg: key E24ACC69: public key "Alice (Alice) <alice@eve.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

At this point, the key of your correspondent is now imported into your keyring.

Verify this:

```
$ gpg --list-keys
```

The output should be similar to this:

```
/home/sysadmY/.gnupg/pubring.gpg
-------------------------------
pub   2048R/C9FBE546 2013-07-30
uid                  Bob Bobson (Work address) <bob@bob.com>
sub   2048R/3BE8FE75 2013-07-30

pub   2048R/E24ACC69 2013-07-31
uid                  Alice (Alice) <alice@eve.com>
sub   2048R/438E172B 2013-07-31
```

. . . note that your public keyring now contains two keys!

Note: you can verify that you still only have your own SECRET key in your SECRET keyring - verify this with the command:

```
$ gpg --list-secret-keys
```

. . . you should only see your own key. This is expected: you only imported the PUBLIC key of your colleague.

2.3 Encrypting using the imported key

You should still have a file “my-secrets-myname.txt” from earlier labs. If not, quickly create such a file, replacing “myname” with your own name, and save it.

Now, we will encrypt the file “my-secrets-myname.txt” with the PUBLIC key we have just imported:

```
$ gpg -a -e -r alice@eve.com my-secrets-myname.txt
```

You will see output similar to this:

```
gpg: 438E172B: There is no assurance this key belongs to the named user

pub  2048R/438E172B 2013-07-31 Alice (Alice) <alice@eve.com>
 Primary key fingerprint: 23F5 A6B0 98CC C571 B8DE  9B29 9EDB 8FBE E24A CC69
      Subkey fingerprint: 7062 E046 C0B6 993A 6C62  5E57 657D 4930 438E 172B

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

Notice how you are being informed that you have no proof that this key really belongs to this person…

Think about the implications!

Note: you might be told the file already exists and you should overwrite it:

```
File ‘my-secrets-myname.txt.asc’ exists. Overwrite? (y/N)
```

Answer ‘y’ if this is the case.

You now need to copy the encrypted file over to the account (home directory on their server) of the person (correspondent) who sent you their key.

To do this:

```
scp my-secrets-myname.txt.asc sysadmY@10.10.x.10:
```

… where sysadmY is the login of the user and 10.10.x.10 is the IP of the machine of your correspondent.

Once you have done this, inform them of this, and tell them to try and decrypt it, as described below.

You should soon be receiving a file as well in your home directory, if you haven't already - but only, of course, if you have sent your key to someone and asked them to encrypt a file with your key!

2.4 Decrypting a received file

Check in your home directory to see if you have received a file:

```
$ ls -l my-secrets-*
```

If you see an encrypted file with the name of the colleague (that you had sent your public key to), then it's time to decrypt it!

```
$ gpg my-secrets-theirname.txt.asc
```

If the file is indeed intended for you, you will see something like this (with your own name of course instead of Alice :)

```
You need a passphrase to unlock the secret key for
user: "Alice (Alice) <alice@eve.com>"
2048-bit RSA key, ID 438E172B, created 2013-07-31 (main key ID E24ACC69)

gpg: gpg-agent is not available in this session
Enter passphrase:
```

Here you will enter the passphrase for *your* secret key so you can decrypt the file encrypted for you by your correspondent (using your public key).

You will then see this:

```
gpg: encrypted with 2048-bit RSA key, ID 438E172B, created 2013-07-31
      "Alice (Alice) <alice@eve.com>"
```

. . . if you already have a file with the same name in the current directory, you will see:

```
File ‘my-secrets-theirname.txt’ exists. Overwrite? (y/N)
```

Say ‘y’

If the correspondent sent you a file encrypted for someone else, you would see:

```
gpg: encrypted with RSA key, ID 3BE8FE75
gpg: decryption failed: secret key not available
```

If you compare the keyid with that of your own key (gpg –list-keys), you would see that it doesn’t match: this file was encrypted with someone else’s public key!

If it DID work, congratulations!

You can look at the contents of the file now. . .

```
$ more my-secrets-theirname.txt
```