Computer Security
Lab Exercise 1
Assymmetric Encryption with GPG

October 16, 2016

In this lab you will learn to use GPG for day-to-day usage, most importantly including signing and verifying signatures. GPG (sometimes written as GnuPG) is the GNU Privacy Guard. GPG is an open source implementation of the OpenPGP standard for asymmetric encryption.

1 Verifying Signatures

Verifying the integrity of software you download is important to ensure that your software hasn't been tampered with. This section will show you how to verify signatures if they are available. Your task is to download the GPG stable source code and verify its signature, which you can find under the following url:

```
https://gnupg.org/download/index.html
```

Before you can verify the signature however, you need to import the public key which was used to make it. These are also available from GnuPGs website[1]. Copy the data from the final section, and save it as a plain text file `gnupg.keys`. Then, run the following command:

```
gpg --import gnupg.keys
```

You should see a report of the six keys which were imported. Next, to verify the file itself, run:

```
gpg --verify Downloads/gnupg-2.0.30.tar.bz2.sig
```

You should see a line stating "`Good signature from <person> [unknown]`". This indicates that the signature is valid, and that you have the signers public key. The `[unknown]` indicates that you haven't assigned the public key a trust level. For most purposes, this is not something you should worry about.

[1] `https://gnupg.org/signature_key.html`

2 Cryptographic Checksums

Many programs are not distributed with signature files. It is more common to offer a cryptographic checksum. While this is not a replacement for signatures, and provides far less security, it is helpful to know how to verify them.

There are multiple hash functions which are used for cryptographic checksums, the most common of which are **MD5** and **SHA1**. To compute the checksum of a file, the commands `md5sum <file>` and `sha1sum <file>` can be used respectively. Calculate the SHA1 checksum of the gpg source code, and verify that it matches the one listed on the GnuPG website[2].

3 Generating a Keypair

In order to sign, or receive encrypted messages, you will need your own key pair. To generate one, run:

```
gpg --gen-key
```

Complete the command line dialogue, and wait for the key to be generated. Note that it is **highly** recommended to secure the key with a strong passphrase. Your private key is your digital identity, do not treat it lightly.

4 Key IDs

Many commands in GPG need to identify the key to use. The public keys available can be listed with the command ‘`gpg -k`’, and the private keys with ‘`gpg -K`’. Each key is associated with a long hexadecimal ID, which can be used to refer to it. More conveniently, keys can also be referred to by their email address.

5 Key Management

Once you’ve generated a key, there are a few maintenance operations you may need to do from time to time.

1. Upload your public key to the keyserver at ‘`hkp://keys.gnupg.net`’. You will have to set the ‘`keyserver`’ option in ‘ `/.gnupg/gpg.conf`’, and then run ‘`gpg --send-keys <Key ID>`’.
2. Make sure you can receive a coursemate’s public key. After they have uploaded theirs, run ‘`gpg --recv-keys <Key ID>`’. Note that in this situation, the key ID *must* be the full hexadecimal ID, and an email address does not suffice.

[2]`https://gnupg.org/download/integrity_check.html`

3. Generate a revocation certificate for your key, using the command '`gpg --gen-revoke <Key ID>`'. A revocation certificate can be used to invalidate your key pair. This is not something you want to do right now, however it is helpful to know what to do. The revocation certificate can be imported with '`gpg --import`', similarly to keys. The (now revoked) public key can then be pushed to a keyserver.

4. You can export your keys with the command '`gpg --export > gpg.keys`'. This will create a binary file 'gpg.keys', containing all public keys in your database. It is also possible to export private keys, using the command '`gpg --export-secret-keys > gpg_private.keys`'. When exported in this way, the keys are still encrypted with your passphrase.

6 Signing Messages

GPG signatures operate on files. The most basic way to sign a file is to execute '`gpg -b <file>`'. This will create a new file, called '`<file>.sig`', which contains the signature of the file with your private key. Adding the `-a` option will force the signature to be generated in an ASCII format, making it more convenient for embedding.

It is also possible to package the data together with the signature, by running '`gpg -s <file>`'. This is typically used in conjunction with encryption.

7 Encrypting and Decrypting Messages

To encrypt a message, double check that you have a coursemate's public key. Create a plain text file containing your message, and then encrypt it with '`gpg -e <file>`'. Send the newly created file to your coursemate. The same command can also be run with the `-s` option, to also sign the message, and the `-a` option to create an ascii-formatted message.

Hopefully you will have received an encrypted message from one of your coursemates. If not, ask someone to send you one. To decrypt the message, simply run '`gpg -d <file>`'.