Apotheonic Resource Collection

GnuPG Quickstart/Howto

Eventually, this will be fixed so that it will be wired into the rest of the site. Until then, it'll be ugly.

The following is a quickstart guide for users of the basic GnuPG tools on open source Unix/Linux systems. It assumes that readers know how to install GnuPG for their own operating systems, and understand the basics of shell use. It also assumes the readers know what GnuPG is, and why they should use it.

I have an article at the TechRepublic IT Security weblog titled **Using OpenPGP on Unix/Linux systems with GnuPG** that explains some things about what GnuPG is, and why you might want to use it, as well as how to install GnuPG on FreeBSD and Debian GNU/Linux systems. Other than that, you are on your own for that information. Google and Wikipedia can help.

In the following, the dollar sign "$" is used to indicate the shell prompt for your normal, unprivileged user account.

Generate Private Key

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

Prior to creating the private key, the routine initiated by this command will ask some questions. The following is an explanation of what those questions are as of this writing (14 February 2008) and some notes about how you should answer them:

```
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (3) RSA (sign only)
Your selection?
```

Simply pressing the `Enter` key will choose the default. This is almost certainly what you want. Assuming you chose option (1), you will be asked the following:

```
DSA keypair will have 1024 bits.
ELG-E kyes may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

The default here is the parenthetical 2048. That should be fine for the next few years at least, considering the difficulty of cracking strong encryption using a 2048 bit key in any reasonable period using current technology. Next, it will ask how long you want your key to be valid:

```
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)
```

For most purposes, either accepting the default (0) by pressing the `Enter` key without making any changes, or specifying `1y` for one year, should be appropriate. Assuming you entered `1y`, it then asks you to verify the expiration date. Here, `[date]` is a stand-in for the actual time and date information indicated by the gpg utility:

```
Key expires at [date]
Is this correct? (y/N)
```

The default answer is N, for "no". If you want to change your answer, just press `Enter`. If you wish to accept it, enter `y`.

Next it asks you to identify yourself:

```
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:
```

Enter your name here. The next two questions are similarly self-explanatory:

```
Email address:
Comment:
```

After all this information is entered, you should see something like the following. Before answering this question, however, **make sure you read the following notes**.

```
You selected this USER-ID:
    "Sparky Woolworth (wookie) <sparky@woolworth-dudely.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

If Sparky Woolworth is your name, wookie is the comment you entered, and sparky@woolworth-dudely.com is your email address, you should enter `o` if you see the above. Whatever name, comment, and email address is presented, make sure they are accurate and enter `o` to select "Okay" -- or, if they don't match the reality, choose appropriately to change your answers.

The important thing to know before you answer this question is that, after answering this question, you are going to be prompted to create some "randomness" to help GnuPG generate a key that is as difficult to crack as possible. Before answering this question, you should have some other applications open to help in that regard. Compiling software, playing video or music, entering a bunch of semi-random text into a text document, moving the mouse around, and heavy network activity can all help contribute to that needed "randomness".

Finally, when it's done, it will give you some statistical data about your new key, then return you to the standard shell prompt.

Generate Revocation Certificate

This creates a "revocation certificate", which is intended to be published tonotify users of your public key that it has been revoked if your passphrase or private key is compromised by a malicious security cracker.

```
$ gpg --output revoke.asc --gen-revoke 'name'
```

It is generally recommended that a revocation key should be stored on durable physical media in a secure location so that it cannot be compromised by the same person that compromised your private key and/or passphrase.

Generate Public Key

This produces a plain text file called "pubkey.txt" that will contain your public key, where `name` is replaced with the Real Name you used to create the key:

```
$ gpg --armor --output pubkey.txt --export 'name'
```

Register Public Key

This uploads your public key to a keyserver on the PGP Corporation's pgp.net domain so that others can search a central location by name for your public key:

```
$ gpg --send-keys 'name' --keyserver hkp://subkeys.pgp.net
```

Import Key Directly

Assuming you have a plain text file called "pubkey.txt" with the public key of someone to whom you may later want to send encrypted files, this is how you can import the public key contained in that file into your gpg keyring:

```
$ gpg --import pubkey.txt
```

From Keyserver

This command retrieves a public key from the pgp.net keyserver, using the key owner's email address to identify it, where `email` is replaced with that person's email address:

```
$ gpg --recv-keys email --keyserver hkp://subkeys.pgp.net
```

Encrypt A File

To encrypt a file when you have the public key for the intended recipient in your gpg keyring, you can use this command, where `ID` is replaced with that key's ID and `filename.ext` is replaced with the name of the file you wish to encrypt:

```
$ gpg --encrypt --recipient ID filename.ext
```

The short version of the above command is:

```
$ gpg -e -r ID filename.ext
```

In either case, a file is created with the same name, plus an additional `.gpg` file extension added to the end. Thus, if your file is `filename.ext`, you will create an encrypted copy of the file named `filename.ext.gpg`.

If you simply wish to encrypt a file so that nobody but you can read it, you can specify your own key ID as the intended recipient.

Decrypt A File

Decrypting a file is simple:

```
$ gpg --output filename.ext --decrypt filename.ext.gpg
```