Saturday, June 25. 2011

Digital Signatures and Encryption with GPG/PGP

In this day and age, I think it is wise that people use digital signatures with methods such as GPG/PGP to prove authenticity and using its encryption capabilities for privacy when storing mail on untrusted networks, such as those hosting mail in other countries, especially those countries who have questionable laws regardng privacy.

GNU Privacy Guard (GPG) is a free-software drop-in replacement for Symantec's proprietory PGP cryptographic software suite.

It is useful in many ways, from saying "Yes, I really sent that message", to using it to encrypt a message or files for privacy, to something as important as signing a checksum file, after all, what's the point of creating a checksum for a file, since if your machine is compromised, all they need to do is to recreate a new checksum and you're none the wiser, but this is harder to get around when it is also expected to be digitally signed by someone.

GPG is available for Linux, Mac, and Windows.

Windows users should install GPG4Win

Apple users should install GPGMail

GPG with Linux/Unix
This is not designed to be an indepth guide, it's a quickstarter. Most distros by default include GNU Privacy Guard (gpg) in the base install so you should not need install anything, we will be using command line, so if you're in X, open a terminal window...

Some things to remember, when creating keys, it is important to remember to create a revocation key, and to backup not only your public key, but your private and revocation keys as well. It is also important to upload your public key to a key server and make it publicly available via your website so your signature can be confirmed for authenticity and recipients can decode encrypted files you send them.

Create your key

gpg --gen-key and follow the instructions, basically, you want to use the default type (1) RSA and RSA (default)
What keysize do you want? (2048) hit enter, 2048 is very strong.
The key lifetime is up to you, on work keys I would use no more than 2y , if its personal, I use 0 for indefinite, we are going to create a revoke authority key later.

Next enter your First and Last name, the email address this key is associated with, any comment (you don't need one, but if you worked for say TPG, you could enter TPG.
Now you'd end up with something like....
Real name: Noel Butler
E-mail address: deletethis@ausics.net
Comment:
You selected this USER-ID:
  "Noel Butler (TPG) "
Then hit o for OK
Then you will have to enter a pass phrase, think of a good one, do NOT use names or people/pets, or dates, phone numbers etc and make sure you use numbers and mixed upper and lower case characters.

You will need to move your mouse or hit gibberish on the KB so it can generate enough random goodness.

When its done, you will end up with your Key details

gpg: key xxxx94E 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: 2 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2012-07-02
pub 2048R/xxxx94E 2011-07-03 [expires: 2012-07-02]
Key fingerprint = xxxx xxxx xxx x
uid Noel Butler
sub 2048R/ xxxxhy2011-07-03 [expires: 2012-07-02]

Congratulations, most the hard work is done, well, kinda.... copy the characters on the pub line, after the 2048R/ in our example it's xxxx94E

Submit your Key

Now you have your keys, you should send your public key to a keyserver so others can find it.
There are several key servers, they do sync up with each other so you don't have to upload your public key to all of them, just one is sufficent.

Warning: Do not ever send to a key server, or give out to anyone, your private keys! Only your public key.

From our earlier example your key is xxxx94E, so to submit this key you would issue the command
gpg --send-keys --keyserver hkp://subkeys.pgp.net xxxx94E

Next, we are going to export our public key to a text file so we can put it on our website for those who can't access it via key servers for whatever reason...
gpg --export -a -o .gnupg/pubkey.asc xxxx94E

You should now enter your keyID (xxxx94E) in your email client's security section if you want to sign or en/decrypt mail.

Backup your Keys

Now the all important backup stages, first backup your private key...
gpg -ao .gnupg/backup-secretkey.asc --export-secret-keys

then backup your public key...
gpg -ao .gnupg/backup-pubkey.asc --export KEYID

To restore your keys to your keyring in case it gets corrupted or deleted...
gpg --import backup-secretkey.asc
gpg --import backup-pubkey.asc

Revocation Keys

Next step is to create a revocation key in case our key has been compromised...
gpg --output revoke.asc --gen-revoke xxxx94E

If you need to revoke your key, use (and this is how you add someone elses public key to your keyring)
gpg --import revoke.asc, and to complete the revocation
gpg --keyserver subkeys.pgp.net --send xxxx94E but lets hope you never need to do this.

Warning: It is very important that your revocation key, and the backups of your private keys are kept very secure. I usually have a spare directory where all of my gpg backup files are kept, but they are all also encrypted by use of ccrypt, so anyone who gets a hold of them has to know my rather long pass phrase before they can decode them.

Note: You cannot, at least at this point in time, delete your keys from the keyserver network, once there, they are there forever. This is why it is important to create a revocation key, this amends your key status to revoked with, in most cases, a reason, but again, it does not delete them, I hope that one day this will be possible, but as things stand today, it's not.

Expiration of Key

There are merits in setting an expiration in your keys lifetime, it is not essential, but if you are using your ISP's or Work Emails, then it's likely an OK idea.

So, you've done all that, and here it is a year or so down the track and the keys about to expire, the Email account is still valid, so you want to extend the life for another two years, you can do this via the gpg shell using --edit-keys, so, if your email is deletethis@ausics.net you would run

gpg --list-keys deletethis@ausics.net

The resulting information would be

pub 2048R/xxxx94E 2011-07-03 [expires: 2012-07-02]
Key fingerprint = xxxx xxxx xxx x
uid Noel Butler
sub 2048R/ xxxxhy 2011-07-03 [expires: 2012-07-02]

We will need both the primary and sub key ID's in our example,
key 0 xxxx94E
key 1 xxxxhy

Now, run gpg--edit-key xxxx94E
You're now in the gpg shell, by default the primary key is key 0, so just type
expire
You'll see something like

Changing expiration time for the primary key.
Please specify how long the key should be valid.
0 = key does not expire
= key expires in n days
w = key expires in n weeks
m = key expires in n months
y = key expires in n years

Type is, for example 2 years : 2y
You'll then be presented with the new expiration date, enter in y to confirm, and your password when prompted, next we need to update the sub key by typing

key 1
expire
2y
y

and your password when prompted, now, we are almost done, before quitting you need to type
save

Congratulations your keys are now updated, don't forget to save your new key to a text file if you make your public key available via a website etc, and, importantly, you need to submit your updated key to a key server, eg:

gpg --send-keys xxxx94E

Signing Messages and Files

As mentioned earlier I often create checksum files for configs and some files, but, to ensure they are not also tampered with, I also sign them, being text files I can add the signature to it by way of
gpg --clearsign filename
I'll be asked for my pass phrase, once entered, a new signed file will appear, for example, I've just made a checksum file called files.md5, now I gpg --clearsign files.md5 afterward I'll also have files.md5.asc (the signed version) so I can delete files.md5 (the unsigned version).

For obvious reasons you can not really sign a binary file, like a zip file, so you would sign as...
gpg --sign file this results in a sig file being created of same name plus .asc, eg:

gpg --sign CAM results in the file CAM untouched, but the verification file is now created as CAM.gpg which must stay with the file CAM.

Verifying Messages and Files

To verify a file (or message) is from someone, you would typically use gpg --verify filename
The file may be a clear signed file with gpg information embedded or a .gpg file.

In some cases, the file will be unmodified, but will be accompanied by a .sig file, generated by gpg -b filename, in this case you verify it using gpg --verify filename.sig filename

Most email clients do this automatically so you don't need to bother too much there, but if it is an attached message or one sent via other means you may need to issue this, especially so for files.

If you need to import their key (necessary for encryption) use gpg --recv-keys THEIRKEYID

In some cases, if a key is not found, you can try searching key servers by:
gpg --search-keys 'senders email'
eg: gpg --keyserver hkp://subkeys.pgp.net --search-keys 'deletethis@ausics.net'

You'll then be presented with a list of their keys, select the most current key matching the address you are sending to, it should then automatically import that key for you to your keyring

If you have problems, in particular with mail, you may see in some messages GPG/PGP Fingerprint, you can determine their key by using the last eight digits in their fingerprint then try gpg --fingerprint THEIR_KEYID

Verifying Keys

You will from time to time see messages like Good signature but untrusted or unverified or something similar, all this means is the message is likely valid, but untrusted as you are not 100% sure it is truly that person and their right key. The following key signings is not mandatory to effectively use GPG, it is only added as an extra security level, for instance most senior Government officials or corporate directors sending confidential information to each other or their lawyers, accountants etc, should take this extra step.

For organisations and clubs, they sometimes have key signing ceremonies, this is for the web of trust that needs to be created. If you're in an urgent situation and you need to verify immediately and you know the person concerned, you can ring them (or ask them in person) to verify their key, you type in gpg --fingerprint email-address/or Key ID, example: gpg --fingerprint deletethis@ausics.net and look for the Key fingerprint = line, read that line and ask them to verify if it is really them and they will confirm or deny. If you are going to set up trust for someone you do not know, then you should follow the basics, meet the person, in person, they should provide you a printed copy of their key, and produce photo ID, such as a drivers licence so you can know it's really them and you should so same.

The way you do this is issue a gpg --fingerprint your.email > fingerprint.txt and give a copy to each person you want to set up a trust with.

You can print out a page full of these using gpg-key2ps for mass handouts, eg, to print out 2 columns of a full sheet of

A4 size paper, if your keyID is xxxx94E you would use:
gpg-key2ps -p a4 xxxx94E > gpg_fingerprints.ps

Where xxxx94E is your key, if you have multiple keys, just add the extra keyID after the first, when done, simply print gpg_fingerprints.ps

Once you get home, take the key ID and import their key, like gpg --recv-keys THEIRKEYID (from their fingerprint slip) then type gpg --sign-key THEIRKEYID you'll be asked if you're sure, hit Y, then enter your gpg pass phrase, then it'll be signed by you.

Optionally, you can then export the signed key by gpg -a -o THEIRKEYID.sasc --export THEIRKEYID
Send this file by email to the person using the email address in their key, then submit this key to a keyserver, gpg --send-keys THEIRKEYID

When they get it (this applies to when you get one from others also) import that file as mentioned earlier (gpg --import THEIRKEYID.sasc) then send your keyID to keyserver gpg --send-keys MYKEYID

Encryption
To encrypt a file, you must have access to the recipients public key, this allows me to sign it and them to decrypt it...
If you do not have the recipients public key, import it as suggested earlier.

Then to encrypt a file, for example called builds.txt and send it to someone at deletethis@ausics.net, you would use
gpg -e -r deletethis@ausics.net builds.txt

Now you have your encrypted builds.txt.gpg which you can put on a USB key or email to your friend who then decrypts it using your key.

To decrypt a file, all you need to do is use...
gpg --decrypt file

Encrypting with only Password or Pass Phrase
If you want to encrypt a file for general distribution, you of course may not know all of the recipients, to get around this, you can use the symmetric option which takes a password instead of using keys, eg:
gpg -c builds.txt
-or-
gpg --symmetric builds.txt

This will result in builds.txt.gpg the encrypted version of builds.txt (which will remain untouched)

Tips

You might also want to add into ~/.gnupg/gpg.conf
keyserver hkp://keys.gnupg.net
keyserver-options auto-key-retrieve

To list all keys on your keyring with fingerprints use gpg --fingerprint --list-keys

That's it, I hope your not so confused, Have fun....

Posted by NoelB at 19:49