SigningReleases

Signing a release version

Releases must be signed prior to release, but the procedure for how to sign releases hasn't been formalized so far. This document is a scratchboard to gather links and resources so that we can end up with a document with adequate info for the novice ReleaseManager, to be transcribed and put here.

This procedure has been formalized and well documented for httpd, available at http://httpd.apache.org/dev/release.html. Java based projects may find the commons release document useful.

Note that for some good commentary on how users can verify signatures (with some useful background), see http://httpd.apache.org/dev/verification.html

---

Important

Signing requires a private key. Private keys should not be stored on our public servers, baring some policy statement to the contrary.

Release Managers typically build and sign the release on their own secured systems, and then push them to the ASF infrastructure, where other Committers and the Community can check them prior to a release vote.

Some discussion about writing this page...

Not true. There are so many assumptions in this document that it is virtually unusable for other projects. Some examples:

- Assumes that the RM even knows what PGP and GPG are
- Assumes that the RM has a public key and/or knows how to get one
- Assumes the RM knows how to publish a key and even what that means

Well, frankly I'd suggest that performing signing operations is mostly pointless unless you understand the basic ideas of cryptographic signatures, and that an RM should be capable of educating himself to the extent that he *is* aware of the issues.

If you're looking for a good place to start, http://www.gnupg.org/gph/en/manual.html defines the concepts quite nicely.

Understanding digital signatures and using the specific tools are two different things. For example, I know a fair amount about encryption, signatures, certs, CRLs, etc., because I work for an internet security company, and the product I work on uses these technologies extensively. However, as an individual user, I had not had to use the PGP or GnuPG tools until I recently needed to sign an Apache release. My purpose in suggesting a page such as this was to help people with the mechanics of signing a release, not to educate them on what cryptography is or why releases should be signed.

The value of this page is that it shows what a Release Manager needs to learn. Step-By-Step instructions have the fault that they seems to imply that so long as the steps are followed, everything will be okay. Whereas, they are best thought of as aide memoirs detailing the minimum actions that are required to cut a release.

```
  }}}
== Why we need to sign releases ==

Signing releases assures users that the software package they just download has not been tampered with since it
was built.  Users can get the cryptographic signature .asc/.sig file with the release and use their own copy of
PGP/GPG/whatever to 'Verify' the signature, which proves that the software package was not changed since it was
signed.  It is part of the ASF's duty to ensure that we take reasonable precautions that the software we
deliver to users is good and has not been tampered with.

The ASF protects releases in two ways. Most users don't need to know who cut the release just that the release
was not tampered it left apache. MD5 sums are ideally suited for this. Developers and the ASF also need to know
that the distribution was cut by the offical release manager and that the release has not been tampered with
since it was built. Digital signatures allow this to happen. That's why MD5 sums and signatures are needed for
all releases.  (use "openssl md5" or equivalent to generate an MD5 - i.e. openssl md5 filename > filename.md5)
{{{
```

(Can someone confirm where we state this policy officially? Also: I *think* we should restrict signing to only be committers, since the ASF has a specific relationship with committers via CLA, etc. -sc)

What you need to do

- create your own PGP or GPG key
- publish your key to the KEYS file
- sign the release to create a detached signature file
- post the release and its signature to the distribution directory
- (optional) adding a checksum file to the dist directory
- point to instructions on how to verify signatures

Tools you can use

PGP[pgp]

Free:

- **PGP 8.0 Freeware** - http://www.pgpi.org/products/pgp/versions/freeware/
  - Apparently, different versions of PGP require different command line options, or even different commands. For example, some Apache KEYS files say to use 'pgpk -ll', but PGP 6.5.8 has only pgp, not pgpk. At this point, I don't know which version uses what, but I'll see what I can find. Note that most modern PGP versions have handy GUIs as well.
- **The GNU Privacy Guard** - http://www.gnupg.org/ Many add-ins and GUIs for GPG exist now as well.

Commercial:

- **PGP 8.0** - http://www.pgp.com/

Signing files with PGP 8.0 Freeware

To sign a file with PGP 8.0, you need to run PGPmail, despite the fact that signing the release itself is not related to e-mail.

With PGPmail running, select the 'Sign' option, and then select the file(s) you want to sign. If you select multiple files, a separate detached signature file will be created for each file. When the Enter Passphrase dialog appears, make sure you select the correct signing key, and check the 'Detached Signature' and 'Text Output' checkboxes. Then enter your passphrase, and click OK. (Note that if you have already signed a file since starting PGPmail, your passphrase may be cached, so you will not need to enter it again.)

PGPmail will create a detached signature file for each selected file, in the same directory as the original files. The files are created with a .sig extension, so you will need to rename them to have a .asc extension, per Apache convention.
(Note: many projects still use .sig, so either extension should be OK for now -sc)

NOTE: The above instructions may also work for the commercial version of PGP 8.0, but since I don't have that available, I can't check.

=== Step by Step for PGP 8.0 Windows ==

For PGP 8.0 Windows, you have to

Onetime setup

- Unzip the install program
- Run the install program (and restart)
- PressLater on the registration screen. (We can preview indefinitely for non-profit use.)
- Enter your personal name and email (@apache.org), and a passphrase. (8 characters or more.)
- After the keys are generated, open the PGPKeys applet.
- Run Server/Send To/Domain Server to register your public key with pgp.com.
- Run Keys/Export to save your public key as a text file. Add the contents to your project's KEYS FILE.
- Close PGPKeys

Signing ritual

- Open PGPMail
- The document button brings up a standard file dialog. Select the *.zip and *.gz files to sign. Check "Detached Signature" and "Text Output".
- PGP generates plain-text *.sig files (which you can rename to *.asc).
- Close PGPMail
- Upload the *.asc files to the distribution directory, along with your public key.

Note that if PGPMail skips the dialog where you choose "Detached Signature" and so forth, clear its cache.

To learn more aobut PGP 8.0, see the bundled documentation, which is quite good.

Publish your key to the KEYS file

You need to publish just the **public** half of your PGP/GPG key so that users can download it to verify the signatures later. You must publish it to the KEYS file that should be checked into CVS - either in your subproject's area, or perhaps in a global KEYS file somewhere on Apache. You may also wish to publish it to public keyservers as well, although this is optional. Note that the ASF does not have a specific public keyserver. To publish your key to the KEYS file, just export the public half of your key into a plain text file, and then just copy and paste it into the KEYS file. You can optionally add a line above your key with your name on it, but this is not required. Be sure to check in the KEYS file before uploading the release!

Some public servers you might consider:

ldap://keyserver.pgp.com

- note that many builds of at least GnuPG aren't LDAP-enabled

x-hkp://the.earth.li

ldap://europe.keys.pgp.com:11370

http://pgpkeys.mit.edu x-hkp://pgp.mit.edu

are both on the pgp-keys network.

Using GPG

This does **not** replace the official documentation. It is simply a list of steps that work.

Create your key

- gpg --gen-key

Export your key, appending to the project's KEYS file

- (gpg --list-sigs **your name** && gpg --armor --export **your name**) >> KEYS

Signing files

- for i in *.zip; do gpg --output $i.asc --detach-sig --armor $i; done
- for i in *.gz; do gpg --output $i.asc --detach-sig --armor $i; done

Adjust the list as necessary.

Good luck! 🙂

The Apache Web of Trust

If all possible, please get your key cross-signed with some people in the apache web-of-trust; see

```
http://www.apache.org/~henkp/trust/apache.html
```

Step by Step

```
EXPORT

 create an 'ascii armored' (plain text) version of your public key:

   gpg --armor --export 'your name'

 then you point your browser to 'pgp.mit.edu' and paste your public
 key in the box under 'Submit a key', click 'Submit this key ..'.

 It gets sent to a lot of other keyservers too. Now you're public!

 SIGNING

 There is protocol for this, but basically it comes down to this.

 -- Find someone (we'll call him/her Joe) who is in the apache
    web-of-trust ; someone you can meet face-to-face.

 -- go and meet Joe, be sure to bring the output of

      gpg -a -v --fingerprint --list-key 'your name'

    and some photo ID. Joe will do the same.

 -- Verify that that Joe is who he says he is (by photo ID)

 -- make Joe say that the key on the paper is his ; let him sign it
    Ask Joe to identify the uid's (email addresses) you are
     supposed to sign.

 -- take home the paper with Joe's key and finger print

 -- At home, get Joe's key from pgp.mit.edu (say the ID of
    Joe's key is 0xABABABA

      gpg --keyserver pgp.mit.edu --recv-key 0xABABABA

 -- verify that the fingerprint on the key

      gpg -v --with-fingerprint --list-key 0xABABABA

    is the same as the one on the paper

 -- To each of the uid's you are supposed to sign, send a random message,
    crypted with Joe's key, Joe should be able to read and decrypt each
    random message. Of course you only sign the uid's that check out.
    To crypt a file MESAGE with Joe's key :

      gpg -v -e -r 0xABABABA MESAGE

 -- sign Joe's key ; horse around with

      gpg --edit 0xABABABA

 -- after you signed the key, extract

      gpg -a --export 0xABABABA

    and send it to Joe or submit it to pgp.mit.edu :

      gpg -v --key-server pgp.mit.edu --send-key 0xABABABA

 -- done.

 Some more :

   http://www.lysator.liu.se/~jc/signing-policy.html

 If you don't know any apache people in your area, look for others
 you can cross-sign with.
```