Table of Contents

Package signing as implemented in certification.

Introduction

A general overview of package signing could be found here and in the references therein. This document is to describe how package signing is implemented in certification. According to the Release Workflow, Signing is implemented after the PatchVerification stage.

Procedure for the certification team

Follow the steps described below:

Setting up the infrastruucture:

- Generate a key like this
- Generate a revocation certificate like this. Probably this will not be used, but always good to have one.
- Back up the key in this way
- Nominate a place where publick key will be published, and circulate this information using the relevant lists and tools only after the whole prcedure has been tested.
- Since package managers cannot handle public (remote/netwok) keyrings we do not upload the public key to any keyring.
- Nominate 2 people who is allowed to own the key and who know the password, they will be the signing responsible persons (SRPs), they should have a common and unique contact point (email, jabber, whatever), i.e. a certifier should not know in advance which SRP is on duty a given time.

Operation

- When a certifier certified a patch triggers the SRP contact asking for signing the relevant rpm. This should be a formal message/mail having a commonly agreed format. For example by moving the patch to 'Certified' in Savannah, Savannah itself sends a message.
- The SRP on duty acknowledges the request in the shortest possible time and signs the rpms and sends an other notif when the signing is done. (May these notif could be attached to the patch later on.) While the sending of message can be scripted the signing itself should not be integrated together with a script performing other functionalities (like move patch). The signing script should only sign and should be executed manually. The time scale between the trigger and the acknowledgement should be in the order of some 10 minutes. If this turns out to put a too high load on the SRPs we can reviese this policy.
- After signing the rpm the SRP sets the patch state ('Signed') and moves the patch (physically) to the appropriate place. It can be Savannah itself sending the ack message about the signing. Being Savannah the tool sending the message we will have the timing information between the various steps.

Transition

- Only the packages went through on the above procedure should be signed. Old packages in the repository should not be signed, because **a.)** you cannot be sure anymore that the packages in the repo are indeed the ones have been certified some month/year ago, **b.)** it would introduce a (not too serious) problem of versioning/rebuilding, etc...
- Since there are a lot of package which are changing very slowly, they will be signed only in the far future, which questions the seriousity of the whole process, since signing is really meaningful only if *every pacakge* in a repository is signed. This fact points toward the whole re-certification of the gLite

stack :) :(An alternative approach could be to do this re-certification metapackage by metapackage.
- Thus, in this transition period one (ex. YAIM) should maintain a (signed) list about the signed rpms, and check only that ones.
- Since the duration of this transition period can easily extend a year or so, the validity of the generated key should be greater than that in order to avoid extra complications.

For site administrators

This section describes various procedure for site administrators for several package manager.

The RPM package manager

- Importing the public key to keystore
    1. Download the key: =wget http://certif-website/certpub.gpg=
    2. Import the public key to the keyring: `rpm --import certpub.gpg`, you can directly import the key using =rpm --import http://certif-website/certpub.gpg=
    3. Check that the key has been imported with : `rpm -q gpg-pubkey --qf '%{name}-%{version}-%{release} --> %{summary}\n'`
    4. Check the fingerprint og the key with `rpm -q gpg-pubkey`. If it is different then the one published, do not use the key.
- Verifying packages
    1. Check that the key is imported.
    2. Execute: `rpm -K package.rpm`
- Removing the key from the keystore
    1. rpm -e gpg-pubkey-version-release
    2. Sometime a key has been imported multiple time, then you can use the `-allmatches` switch to remove all the instance.

The YUM package manager

Yum is a fronted which uses the rpm package manager. In order to allow yum checking the package signatures

- repeat the procedure described for the RPM package manager and
- edit the `/etc/yum.conf` file and replace all occurances of `gpgcheck=0` to `gpgcheck=1`.

This is all you have to do for YUM.

The APT package manager for RedHat

- Importing the key
    1. You have to import your key to the GPG keyring as described here
    2. Set `GPG-check=true` in your `/etc/apt/apt.conf`
    3. Configure the file `/etc/apt/apt.conf.d/gpg-checker.conf` to point to your gpg checker.
- Verifying packages
    1. The default setting is that APT verifies package signature. If you want to disable it uncomment `// GPG-Check "false";` in `/etc/apt/apt.conf`.
- Removing the key
    1. You have to remove the key from your GPG keyring as described here

The APT package manager for Debian/Ubuntu

The `apt-get`, `aptitude` and `synaptic` package managers are aware of the following method.

- Importing the public key to keystore
  1. Download the key: =wget http://certif-website/certpub.gpg=
  2. Import the public key: `apt-key add certpub.gpg`
  3. Check that the public key has been imported: `apt-key list`
- Verifying packages
  1. The `apt-get install` command will verify packages automatically. If you want to install something without signature, you can use `--allow-unauthenticated` switch with the command.
  2. For verifying the signature of standalon packages you can use the `debsig-verify` utility.
- Removing the key from keystore
  1. Execute: `apt-key del KEYID`

-- GergelyDebreczeni - 04 Feb 2009

This topic: EGEE > CertificationSigningImplementation
Topic revision: r5 - 2010-02-12 - unknown