NAME

apt-secure − Archive authentication support for APT

DESCRIPTION

Starting with version 0.6, **APT** contains code that does signature checking of the Release file for all repositories. This ensures that data like packages in the archive can't be modified by people who have no access to the Release file signing key. Starting with version 1.1 **APT** requires repositories to provide recent authentication information for unimpeded usage of the repository. Since version 1.5 changes in the information contained in the Release file about the repository need to be confirmed before APT continues to apply updates from this repository.

Note: All APT−based package management front−ends like **apt-get**(8), **aptitude**(8) and **synaptic**(8) support this authentication feature, so this manpage uses APT to refer to them all for simplicity only.

UNSIGNED REPOSITORIES

If an archive has an unsigned Release file or no Release file at all current APT versions will refuse to download data from them by default in **update** operations and even if forced to download front−ends like **apt-get**(8) will require explicit confirmation if an installation request includes a package from such an unauthenticated archive.

You can force all APT clients to raise only warnings by setting the configuration option **Acquire::AllowInsecureRepositories** to true. Individual repositories can also be allowed to be insecure via the **sources.list**(5) option allow−insecure=yes. Note that insecure repositories are strongly discouraged and all options to force apt to continue supporting them will eventually be removed. Users also have the **Trusted** option available to disable even the warnings, but be sure to understand the implications as detailed in **sources.list**(5).

A repository which previously was authenticated but would loose this state in an **update** operation raises an error in all APT clients irrespective of the option to allow or forbid usage of insecure repositories. The error can be overcome by additionally setting **Acquire::AllowDowngradeToInsecureRepositories** to true or for Individual repositories with the **sources.list**(5) option allow−downgrade−to−insecure=yes.

SIGNED REPOSITORIES

The chain of trust from an APT archive to the end user is made up of several steps. **apt−secure** is the last step in this chain; trusting an archive does not mean that you trust its packages not to contain malicious code, but means that you trust the archive maintainer. It's the archive maintainer's responsibility to ensure that the archive's integrity is preserved.

apt−secure does not review signatures at a package level. If you require tools to do this you should look at **debsig−verify** and **debsign** (provided in the debsig−verify and devscripts packages respectively).

The chain of trust in Debian starts (e.g.) when a maintainer uploads a new package or a new version of a package to the Debian archive. In order to become effective, this upload needs to be signed by a key contained in one of the Debian package maintainer keyrings (available in the debian−keyring package). Maintainers' keys are signed by other maintainers following pre−established procedures to ensure the identity of the key holder. Similar procedures exist in all Debian−based distributions.

Once the uploaded package is verified and included in the archive, the maintainer signature is stripped off, and checksums of the package are computed and put in the Packages file. The checksums of all of the Packages files are then computed and put into the Release file. The Release file is then signed by the archive key for this Ubuntu release, and distributed alongside the packages and the Packages files on Ubuntu mirrors. The keys are in the Ubuntu archive keyring available in the ubuntu−keyring package.

End users can check the signature of the Release file, extract a checksum of a package from it and compare it with the checksum of the package they downloaded by hand − or rely on APT doing this automatically.

Notice that this is distinct from checking signatures on a per package basis. It is designed to prevent two possible attacks:

• Network "man in the middle" attacks. Without signature checking, malicious agents can introduce themselves into the package download process and provide malicious software either by controlling a network element (router, switch, etc.) or by redirecting traffic to a rogue server (through ARP or

DNS spoofing attacks).

- Mirror network compromise. Without signature checking, a malicious agent can compromise a mirror host and modify the files in it to propagate malicious software to all users downloading packages from that host.

However, it does not defend against a compromise of the master server itself (which signs the packages) or against a compromise of the key used to sign the Release files. In any case, this mechanism can complement a per−package signature.

INFORMATION CHANGES

A Release file contains beside the checksums for the files in the repository also general information about the repository like the origin, codename or version number of the release.

This information is shown in various places so a repository owner should always ensure correctness. Further more user configuration like **apt_preferences**(5) can depend and make use of this information. Since version 1.5 the user must therefore explicitly confirm changes to signal that the user is sufficiently prepared e.g. for the new major release of the distribution shipped in the repository (as e.g. indicated by the codename).

USER CONFIGURATION

apt−key is the program that manages the list of keys used by APT to trust repositories. It can be used to add or remove keys as well as list the trusted keys. Limiting which key(s) are able to sign which archive is possible via the **Signed−By** in **sources.list**(5).

Note that a default installation already contains all keys to securely acquire packages from the default repositories, so fiddling with **apt−key** is only needed if third−party repositories are added.

In order to add a new key you need to first download it (you should make sure you are using a trusted communication channel when retrieving it), add it with **apt−key** and then run **apt−get update** so that apt can download and verify the InRelease or Release.gpg files from the archives you have configured.

REPOSITORY CONFIGURATION

If you want to provide archive signatures in an archive under your maintenance you have to:

- *Create a toplevel Release file*, if it does not exist already. You can do this by running **apt−ftparchive release** (provided in apt−utils).
- *Sign it*. You can do this by running **gpg −−clearsign −o InRelease Release** and **gpg −abs −o Release.gpg Release**.
- *Publish the key fingerprint*, so that your users will know what key they need to import in order to authenticate the files in the archive. It is best to ship your key in its own keyring package like Ubuntu does with ubuntu−keyring to be able to distribute updates and key transitions automatically later.
- *Provide instructions on how to add your archive and key*. If your users can't acquire your key securely the chain of trust described above is broken. How you can help users add your key depends on your archive and target audience ranging from having your keyring package included in another archive users already have configured (like the default repositories of their distribution) to leveraging the web of trust.

Whenever the contents of the archive change (new packages are added or removed) the archive maintainer has to follow the first two steps outlined above.

SEE ALSO

apt.conf(5), **apt-get**(8), **sources.list**(5), **apt-key**(8), **apt-ftparchive**(1), **debsign**(1), **debsig-verify**(1), **gpg**(1)

For more background information you might want to review the **Debian Security Infrastructure**[1] chapter of the Securing Debian Manual (also available in the harden−doc package) and the **Strong Distribution HOWTO**[2] by V. Alex Brennen.

BUGS

APT bug page[3]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug−reporting.txt or the **reportbug**(1) command.

AUTHOR

APT was written by the APT team <apt@packages.debian.org>.

MANPAGE AUTHORS

This man−page is based on the work of Javier Fernández−Sanguino Peña, Isaac Jones, Colin Walters, Florian Weimer and Michael Vogt.

AUTHORS

Jason Gunthorpe

APT team

NOTES

1. Debian Security Infrastructure
   https://www.debian.org/doc/manuals/securing-debian-howto/ch7
2. Strong Distribution HOWTO
   http://www.cryptnet.net/fdp/crypto/strong_distro.html
3. APT bug page
   http://bugs.debian.org/src:apt