Autoload Script by project

fast, secure, easy autoload

Yann Esposito

[2019-08-18 Sun]

tl;dr: A script that use projectile and GPG to securely load an emacs lisp script when opening a new project.

Check the #solution section to get the code.

Problem

When providing a repository containing only org files of my blog. I also wanted to provide everything necessary for users to be able to publish my website. Emacs, org-publish mostly assume you should put all those details in a centralised place in your `~/.emacs.d/.init.el` file.

The main principle is quite simple.

1. When opening a new file in a project, check for the presence of a `project.el` and `project.el.sig` file at the root directory of the project.
2. Check the file was signed with by a trusted fingerprint.
3. Load the file.

Other solutions I found on the internet asked you each time you enter in a project if you trust the file or not. This was both quite annoying and insecure as it is kind of easy to type 'y' instead of 'n' and to load 3rd party script.

Note that checking who signed a file with an external signature is not as straightforward as it should be:

```
(defun auto-load-project/get-sign-key (file)
  "Return the fingerprint of they key that signed FILE.

To sign a file you should used

`gpg --local-user my@email --output project.el.sig --detach-sign project.el`"
  (string-trim-right
   (shell-command-to-string
    (concat
```

```
(format "gpg --status-fd 1 --verify %s.sig %s 2>/dev/null " file file)
"|grep VALIDSIG"
"|awk '{print $3}'"))))
```

- The '–status-fd' should provide more script friendly output. GPG provide localized output by default which are therefore hard to use in script (for grep for example).

We use `projectile` to detect the project-root and when we are in a new project. Unfortunately the `projectile-after-switch-project-hooks` doesn't work as I expected. So I use the hooks `find-file-hook` and `dired-mode-hook` to try to load the file. In order not to load the code each time, I need to keep a local state of project already loaded.

So now, each time I modify the `project.el` I sign it with the following command line:

```
gpg --local-user my@email --output project.el.sig --detach-sign project.el
```

Solution

The project is hosted here: https://gitlab.esy.fun/yogsototh/auto-load-project-el

You can setup the emacs package in spacemacs with:

```
;; ...
dotspacemacs-additional-packages
'((auto-load-project :location
                     (recipe
                      :fetcher git
                      :url "https://gitlab.esy.fun/yogsototh/auto-load-project-el"
                      :files ("auto-load-project.el"))))
;; ...
(defun dotspacemacs/user-config ()
  ;; ...
  (require 'auto-load-project)
  (setq auto-load-project/trusted-gpg-key-fingerprints
        '("0000000000000000000000000000000000000000"
          "1111111111111111111111111111111111111111"
          "2222222222222222222222222222222222222222"
          )))
;; ...
```

The full current code should be easy to follow if you have basic notions of eLISP:

```
(require 'projectile)

(defvar auto-load-project/trusted-gpg-key-fingerprints
```

```
  '()
  "The list of GPG fingerprint you trust when decrypting a gpg file.
You can retrieve the fingerprints of your own private keys
with: `gpg --list-secret-keys' (take care of removing the
spaces when copy/pasting here)")

(defun auto-load-project/get-sign-key (file)
  "Return the fingerprint of they key that signed FILE.

To sign a file you should used

`gpg --local-user my@email --output project.el.sig --detach-sign project.el`"
  (string-trim-right
   (shell-command-to-string
    (concat
     (format "gpg --status-fd 1 --verify %s.sig %s 2>/dev/null " file file)
     "|grep VALIDSIG"
     "|awk '{print $3}'"))))

(defun auto-load-project/trusted-gpg-origin-p (file)
  "Return non-nil if the FILE is encrypted with a trusted key."
  (member (auto-load-project/get-sign-key file)
          auto-load-project/trusted-gpg-key-fingerprints))

(defconst auto-load-project/project-file "project.el"
  "Project configuration file name.")

(defvar auto-load-project/loaded-projects (list)
  "Projects that have been loaded by `auto-load-project/load'.")

(defun auto-load-project/load ()
  "Loads the `auto-load-project/project-file' for a project.
This is run once the project is loaded signifying project setup."
  (interactive)
  (when (projectile-project-p)
    (lexical-let* ((current-project-root (projectile-project-root))
                   (project-init-file (expand-file-name auto-load-project/project-
file current-project-root))
                   (project-sign-file (concat project-init-file ".sig")))
         (when (and (not (member current-project-root auto-load-project/loaded-
projects))
                 (file-exists-p project-init-file)
                 (file-exists-p project-sign-file)
                 (auto-load-project/trusted-gpg-origin-p project-init-file))
        (message "Loading project init file for %s" (projectile-project-name))
        (condition-case ex
```

```
            (progn (load project-init-file)
              (add-to-list 'auto-load-project/loaded-projects current-project-root)
                    (message "%s loaded successfully" project-init-file))
          ('error
           (message
            "There was an error loading %s: %s"
            project-init-file
            (error-message-string ex))))))))

(add-hook 'find-file-hook #'auto-load-project/load t)
(add-hook 'dired-mode-hook #'auto-load-project/load t)

(provide 'auto-load-project)
```