Vimproved, bizarrerie ou non ?

mercredi 20 février 2013, par Olivier Duquesne (DaffyDuke)

Contexte

Un fichier.

Deux user distincts, dans le même groupe Unix.

Des permissions propre à un seul utilisateur.

Le fichier est dans un répertoire modifiable par le groupe.

```
user@host:/tmp # ls -l file
-rwxr-xr-x 1 user staff 209 Feb 15 10:40 file

hacker@host:/tmp # chown hacker file
chown: changing ownership of `file': Operation not permitted

hacker@host:/tmp # echo coin >> file
-ksh: file: cannot create [Permission denied]

hacker@host:/tmp # sed -i -e "s/coin/pan/g" file

hacker@host:/tmp # ls -l file
-rwxr-xr-x 1 hacker staff 209 Feb 15 10:41 file
```

Sed

Sed fait les choses proprement. L'utilisateur qui n'a a priori pas le droit de modifier le fichier si on s'en tient aux permissions Unix le modifie, et en plus se l'approprie. Certes l'inode change.

En réalité, c'est un fonctionnement normal de sed via l'option -i . Lisons le man :

```
-c, --copy

use copy instead of rename when shuffling files in -i mode
(avoids change of input file ownership)
```

Vim

Vim en fait autant. Le phénomène est dit naturel, et apparaît dans le man, tout simplement .... Et tant pis si cela dépasse la compréhension des permissions unix :

```
:w[rite]! [++opt] Like ":write", but forcefully write when 'readonly' is
set or there is another reason why writing was
refused.
Note: This may change the permission and ownership of
```

```
the file and break (symbolic) links. Add the 'W' flag
to 'cpoptions' to avoid this.
```

Source : http://vimdoc.sourceforge.net/htmld...

Mais en réalité, l'inode n'est pas modifié avec vim, contrairement à sed. D'autres se sont posés la question, voir ce sujet sur http://unix.stackexchange.com/quest...

```
You, glen, are the owner of the directory (see the . file in your listing). A
directory is just a list of files and you have the permission to alter this list
(e.g. add files, remove files, change ownerships to make it yours again, etc.).
You may not be able to alter the contents of the file directly, but you can read
and unlink (remove) the file as a whole and add new files subsequently.1 Only
witnessing the before and after, this may look like the file has been altered.

Vim uses swap files and moves files around under water, so that explains why it
seems to write to the same file as you do in your shell, but it's not the same
thing.2

So, what Vim does, comes down to this:

cat temp > .temp.swp # copy file by contents into a new glen-owned file
echo nope >> .temp.swp # or other command to alter the new file
rm temp && mv .temp.swp temp # move temporary swap file back

1This is an important difference in file permission handling between Windows and
Unices. In Windows, one is usually not able to remove files you don't have write
permission for.

2 update: as noted in the comments, Vim does not actually do it this way for
changing the ownership, as the inode number on the temp file does not change
(comaring ls -li before and after). Using strace we can see exactly what vim does.
The interesting part is here:

open("temp", O_WRONLY|O_CREAT|O_TRUNC, 0664) = -1 EACCES (Permission denied)
unlink("temp") = 0
open("temp", O_WRONLY|O_CREAT|O_TRUNC, 0664) = 4
write(4, "more text bla\n", 14) = 14
close(4) = 0
chmod("temp", 0664) = 0

This shows that it only unlinks, but does not close the file descriptor to temp.
It rather just overwrites its whole contents (more text bla\n in my case). I
guess this explains why the inode number does not change.
```

Autrement dit, vim ne ferme juste pas le filedescriptor que l'OS lui autorise à ouvrir.

Trop gros pour être vrai ?

Essayez .....

P.-S.

Note le user "hacker" dans l'exemple n'est que carricatural, les résultats sont tout à fait normaux.