su, sudo

New Identity

Better safe than sorry – even if you have root privileges for a system, it makes sense to use them only temporarily to prevent accidental damage. su and sudo allow you to switch ID quickly on the command line. BY HEIKE JURZIK

Root privileges are required for administrative tasks, but it does not makes sense to be the superuser all the time. It is preferable to become *root* for an administrative task and then revert to being a "normal" user. Two commands, *su* and *sudo*, allow you to switch your ID.

Su

The *su* ("substitute user") command allows you to switch your ID on the command line. The command launches a new shell in the background using new user (**UID**) and group (**GID**) IDs. When you type *su* to become the superuser, or another privileged user, you are expected to know the password for that user's account.

The basic syntax of the command is *su [-] [username]* – but there is a subtle difference, depending on whether you type the minus sign or not. The minus sign (or alternatively the parameter *-l*, or its long form *--login*) ensures that you actually log in, thus setting the appropriate environment variables and shell, and changing to the appropriate home directory. The environment variables are unchanged if you omit the minus sign, and this could mean that the new user has no privileges for the current directory (see Figure 1).

If you do not supply a name, the superuser account, *root* is assumed. This is also what leads to the misconception that *su* is actually an abbreviation of "superuser".

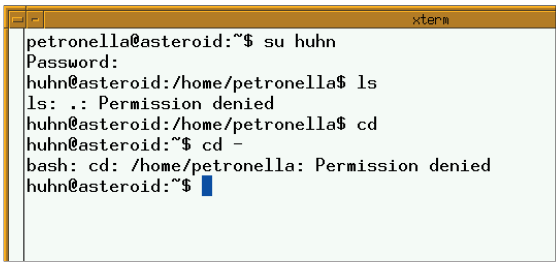
By default the *su* command does not allow the new user to launch X applications. "External" users must first be granted permission to use the X server for output, and this means editing the *.Xauthority* file in the appropriate home directory (see also *man xauth*). To allow the *root* user to launch an X program in an Xterm that belongs to the user *petronella*, you need to extract a key from *.Xauthority*, add it to the administrator's *.Xauthority*, and then redefine the *DISPLAY* variable (see Listing 1).

su also allows you to use another account to launch a single command. To do so, stipulate the *-c* (*--command*) option:

```
huhn@asteroid:~$ su -c "less ↩
/var/log/messages"
Password:
```

The use of the *su* command is logged. Depending on the distro you use, these log entries will be located in */var/log/auth.log* (e.g. Debian) or */var/log/messages* (e.g. Suse Linux). Invalid attempts are easily located, allowing the admin user to see quickly who has tried to misappropriate root privileges:

```
Dec 22 14:50:50 asteroid ↩
PAM_unix[2108]: authentication ↩
failure;
(uid=500) -> root for su service
Dec 22 14:50:52 asteroid ↩
su[2108]: pam_authenticate: ↩
Authentication failure
```



```
petronella@asteroid:~$ su huhn
Password:
huhn@asteroid:/home/petronella$ ls
ls: .: Permission denied
huhn@asteroid:/home/petronella$ cd
huhn@asteroid:~$ cd -
bash: cd: /home/petronella: Permission denied
huhn@asteroid:~$
```

Figure 1: Without a proper login, you might not have any privileges

Listing 1: Display

```
petronella@asteroid:~$ xauth
extract key $DISPLAY
huhn@asteroid:~$ su -
Password:
asteroid:~# xauth merge
/home/huhn/key
asteroid:~# export DISPLAY=:0.0
```

```
Dec 22 14:50:52 asteroid ↩
su[2108]: - pts/8 huhn-root
```

If you are the admin user, you do not need to enter a password after typing the *su* command. You can assume any identity to quickly test a modification from another user's perspective.

Do the sudo

The *sudo* command allows you to avoid disclosing the root password for a machine, which is understandable for security reasons. The command does what the name suggests: "sudo" stands for "substitute user, do", and provides individual users or groups with administrative privileges for a limited period, and limited to a specific task. A user can then simply type her own password to launch the privileged command.

The admin user needs to create a list of users permitted to run specific privileged commands in the */etc/sudoers* file. While working as *root*, edit the file with the *visudo* command. This program offers the usual features of the *vi* editor, plus a few additional functions. *visudo* "locks" the */etc/sudoers* file to prevent it being edited by multiple users at the same time. Also, *visudo* checks the syntax of the file on quitting, and informs you of any errors it finds:

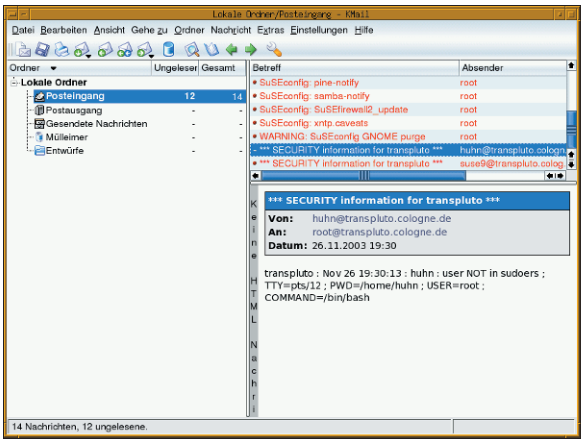
```
>>> sudoers file: syntax ↩
error, line 20 <<<
What now?
```

You have three choices: press *e* to edit the file again, *x* to cancel the changes and quit the editor, or *Q* to save the changes anyway.

There is a default entry for *root ALL=(ALL) ALL* in */etc/sudoers*. This allows the root user to do everything, but of course root can do that anyway without *sudo*. If you need to grant another user on a machine unrestricted root privileges, simply copy this line, and replace *root* with the name of that user. After you save the file, this user will be able to run administrative commands by prepending *sudo*, e.g.:

```
huhn@asteroid:~$ sudo ↩
/sbin/shutdown
Password:
```

If the user is not permitted to use *sudo*, a message such as "*sudo*: *huhn is not in the sudoers file. This incident will be reported.*" will occur. The default behavior, which can be changed in */etc/sudoers*, is to mail a warning to the admin with the details of the user who attempted to launch *sudo* (see Figure 2). To be on the safe side, non-privileged users can enter *sudo -l* to display a list of the permitted commands.


Lokale Ordner/Posteingang - KMail
Datei Bearbeiten Ansicht Gehe zu Ordner Nachricht Extras Einstellungen Hilfe
Ordner
Ungeleser
Gesamt
Betreff
Absender
Keine HTML Nachricht

```
Lokale Ordner
  Posteingang              12   14
  Postausgang              -    -
  Gesendete Nachrichten    -    -
  Mülleimer                -    -
  Entwürfe                 -    -

• SuSEconfig: pine-notify                     root
• SuSEconfig: samba-notify                    root
• SuSEconfig: SuSEfirewall2_update            root
• SuSEconfig: xntp.caveats                    root
• WARNING: SuSEconfig GNOME purge             root
- *** SECURITY information for transpluto ***  huhn@transpluto.cologn
• *** SECURITY information for transpluto ***  suse9@transpluto.colog

*** SECURITY information for transpluto ***
Von:    huhn@transpluto.cologne.de
An:     root@transpluto.cologne.de
Datum:  26.11.2003 19:30

transpluto : Nov 26 19:30:13 : huhn : user NOT in sudoers ;
TTY=pts/12 ; PWD=/home/huhn ; USER=root ;
COMMAND=/bin/bash

14 Nachrichten, 12 ungelesene.
```

Figure 2: Security is at a premium – sudo reports illegitimate access

Granular Control

The *Host alias specification* section of */etc/sudoers* allows you to specify the computers where specific *sudo* commands should apply. You can use the *Host_Alias* to create a group of computers by specifying their names or defining an IP address scope. This feature only makes sense if you will be applying a centrally managed *sudo* configuration to multiple computers.

The *User_Alias* section allows you to group users who require the same privileges. First define the alias type (e.g. *User_Alias*), then an alias name (which can contain capital letters, underscores, and numbers), a mapping as indicated by the " = " sign, and finally the comma-separated username. Let's add the users *huhn* and *petronella* to a group which is allowed to shutdown the machine:

```
# User alias specification
User_Alias SHUTTERSDOWN=↩
petronella,huhn
```

The next step is to define an alias for the *shutdown* command in the *Cmnd alias specification* section. To do so, enter the absolute path to the required program:

```
# Cmnd alias specification
Cmnd_Alias DOWN = /sbin/shutdown
```

To tell *sudo* that the *SHUTTERSDOWN* are allowed to run this command, we need another entry below *User privilege specification*:

```
SHUTTERSDOWN ALL = DOWN
```

Users in the *SHUTTERSDOWN* group can now shut down the computer by typing *sudo /sbin/shutdown*. But there is an easier way to grant a single user permission to run a single command. For example, the entry

```
huhn ALL = /usr/sbin/visudo
```

grants the user *huhn* permission to edit the */etc/sudoers* file using the *sudo /usr/sbin/visudo* command.

Restricted or Free?

A simple entry in */etc/sudoers* allows you to withdraw privileges for individual users. The syntax for doing so is as follows:

```
SHUTTERSDOWN ALL = DOWN
petronella ALL = !DOWN
```

It is important to specify the exception immediately after the rule, as the file is parsed top down. This allows you to keep the *SHUTTERSDOWN* group, which might be permitted to run other commands, while at the same time restricting *petronella* from shutting the computer down. If *petronella* attempts to run the command, she will simply be shown a message like the following one: "*Sorry, user petronella is not allowed to execute '/usr/sbin/visudo' as root on asteroid.linux-magazine.com.*"

If you want to suppress the password prompt for individual or multiple commands, simply set the *NOPASSWD* flag:

```
SHUTTERSDOWN ALL=NOPASSWD:DOWN
```

Instead of reducing the level of security that *sudo* provides, you can increase it by forcing users to enter their passwords every time they run *sudo*. By default *sudo* runs a kind of ticket system with a timeout which ensures, for example, that an orphaned *root* shell on the console will not open up the machine to compromise by all and sundry. The default ticket validity for most distros is 15 minutes. But you can set the timeout to 0 minutes by adding the following line to */etc/sudoers*:

```
Defaults timestamp_timeout = 0
```

Optional

sudo also has a few command line parameters. The most important of these is probably *-s*, which allows you to launch a root shell. There is no need to configure X server access, simply typing *sudo -s* will suffice to allow the administrator to launch X programs.

The *-L* switch lists all the options in the */etc/sudoers* file. If you want to extend your ticket without running a command, simply enter *sudo -v*. If the timeout has already expired, you will be prompted to enter your password. You can also drop a ticket by entering *sudo -k*. The *-b* flag allows you to run a command in the background – however, you will not be able to move it back to the foreground with the normal shell job control command, *fg*. ■

GLOSSARY

UID: *Each user is identifiable by means of a UID ("User IDentification number"), which is uniquely mapped to that user's account. You can easily find out your own UID by typing "echo $UID".*

GID: *Besides the UID, users have a so-called GID ("Group IDentification number") which indicates their group membership. The members of a group can share privileges. The "id" command indicates your current UID and GID.*