sudo

以其他身份来执行命令

补充说明

sudo命令 用来以其他身份来执行命令，预设的身份为root□在/etc/sudoers中设置了可执行sudo指令的用户。若其未经授权的用户企图使用sudo□则会发出警告的邮件给管理员。用户使用sudo时，必须先输入密码，之后有5分钟的有效期限，超过期限则必须重新输入密码。

语法

```
sudo(选项)(参数)
```

选项

```
-b□在后台执行指令；
-E□继承当前环境变量
-h□显示帮助；
-H□将HOME环境变量设为新身份的HOME环境变量；
-k□结束密码的有效期限，也就是下次再执行sudo时便需要输入密码；。
-l□列出目前用户可执行与无法执行的指令；
-p□改变询问密码的提示符号；
-s<shell>□执行指定的shell□
-u<用户>：以指定的用户作为新的身份。若不加上此参数，则预设以root作为新的身份；
-v□延长密码有效期限5分钟；
-V □显示版本信息。
```

参数

指令：需要运行的指令和对应的参数。

实例

```
$ sudo su -
# env | grep -E
'(HOME|SHELL|USER|LOGNAME|^PATH|PWD|TEST_ETC|TEST_ZSH|TEST_PRO|TEST_BASH|TES
T_HOME|SUDO)'
```

这个命令相当于使用root超级用户重新登录一次shell□只不过密码是使用的当前用户的密码。而且重要是，该命令会 **重新加载/etc/profile文件以及/etc/bashrc文件等系统配置文件，并且还会重新加载root用户的$SHELL环境变量所对应的配置文件**，比如□root超级用户的$SHELL是/bin/bash□则会加载/root/.bashrc等配置。如果是/bin/zsh□则会加载/root/.zshrc等配置，执行后是完全的root环境。

```
$ sudo -i
# env | grep -E
'(HOME|SHELL|USER|LOGNAME|^PATH|PWD|TEST_ETC|TEST_ZSH|TEST_PRO|TEST_BASH|TES
T_HOME|SUDO)'
```

这个命令基本与 `sudo su -` 相同，执行后也是root超级用户的环境，只不过是多了一些当前用户的信息。

```
$ sudo -s
# env|grep -E
'(HOME|SHELL|USER|LOGNAME|^PATH|PWD|TEST_ETC|TEST_ZSH|TEST_PRO|TEST_BASH|TES
T_HOME|SUDO)'  --color
```

这个命令相当于**以当前用户的$SHELL开启了一个root超级用户的no-login的shell，不会加载/etc/profile等系统配置**。所以/etc/profile文件中定义的TEST_ETC环境变量就看不到了，但是会**加载root用户对应的配置文件**，比如root用户的$SHELL是/bin/zsh，那么会加载/root/.zshrc配置文件，执行完后，不会切换当前用户的目录。

配置sudo必须通过编辑/etc/sudoers文件，而且只有超级用户才可以修改它，还必须使用visudo编辑。之所以使用visudo有两个原因，一是它能够防止两个用户同时修改它；二是它也能进行有限的语法检查。所以，即使只有你一个超级用户，你也最好用visudo来检查一下语法。

visudo默认的是在vi里打开配置文件，用vi来修改文件。我们可以在编译时修改这个默认项，visudo不会擅自保存带有语法错误的配置文件，它会提示你出现的问题，并询问该如何处理，就像：

```
>>> sudoers file: syntax error, line 22 <<
```

此时我们有三种选择：键入“e”是重新编辑，键入“x”是不保存退出，键入“Q”是退出并保存。如果真选择Q，那么sudo将不会再运行，直到错误被纠正。

现在，我们一起来看一下神秘的配置文件，学一下如何编写它。让我们从一个简单的例子开始：让用户Foobar可以通过sudo执行所有root可执行的命令。以root身份用visudo打开配置文件，可以看到类似下面几行：

```
# Runas alias specification
# User privilege specificationroot    ALL=(ALL)ALL
```

我们一看就明白个差不多了，root有所有权限，只要仿照现有root的例子就行，我们在下面加一行（最好用tab作为空白）：

```
foobar ALL=(ALL)    ALL
```

保存退出后，切换到foobar用户，我们用它的身份执行命令：

```
[foobar@localhost ~]$ ls /root
ls: /root: 权限不够

[foobar@localhost ~]$ sudo ls /root
PassWord:
anaconda-ks.cfg Desktop install.log install.log.syslog
```

好了，我们限制一下foobar的权利，不让他为所欲为。比如我们只想让他像root那样使用ls和ifconfig，把那一行改为：

```
foobar localhost=    /sbin/ifconfig,   /bin/ls
```

再来执行命令：

```
[foobar@localhost ~]$ sudo head -5 /etc/shadow
```

```
Password:
Sorry, user foobar is not allowed to execute '/usr/bin/head -5 /etc/shadow'
as root on localhost.localdomain.

[foobar@localhost ~]$ sudo /sbin/ifconfigeth0      Linkencap:Ethernet HWaddr
00:14:85:EC:E9:9B...
```

现在让我们来看一下那三个ALL到底是什么意思。第一个ALL是指网络中的主机，我们后面把它改成了主机名，它指明foobar可以在此主机上执行后面的命令。第二个括号里的ALL是指目标用户，也就是以谁的身份去执行命令。最后一个ALL当然就是指命令名了。例如，我们想让foobar用户在linux主机上以jimmy或rene的身份执行kill命令，这样编写配置文件：

```
foobar    linux=(jimmy,rene)    /bin/kill
```

但这还有个问题□foobar到底以jimmy还是rene的身份执行？这时我们应该想到了sudo -u了，它正是用在这种时候□ foobar可以使用sudo -u jimmy kill PID或者sudo -u rene kill PID，但这样挺麻烦，其实我们可以不必每次加-u，把rene或jimmy设为默认的目标用户即可。再在上面加一行：

```
Defaults:foobar    runas_default=rene
```

Defaults后面如果有冒号，是对后面用户的默认，如果没有，则是对所有用户的默认。就像配置文件中自带的一行：

```
Defaults    env_reset
```

另一个问题是，很多时候，我们本来就登录了，每次使用sudo还要输入密码就显得烦琐了。我们可不可以不再输入密码呢？当然可以，我们这样修改配置文件：

```
foobar localhost=NOPASSWD:     /bin/cat, /bin/ls
```

再来sudo一下：

```
[foobar@localhost ~]$ sudo ls /rootanaconda-ks.cfg Desktop install.log
install.log.syslog
```

当然，你也可以说“某些命令用户foobar不可以运行”，通过使用!操作符，但这不是一个好主意。因为，用!操作符来从ALL中“剔出”一些命令一般是没什么效果的，一个用户完全可以把那个命令拷贝到别的地方，换一个名字后再来运行。

日志与安全

sudo为安全考虑得很周到，不仅可以记录日志，还能在有必要时向系统管理员报告。但是□sudo的日志功能不是自动的，必须由管理员开启。这样来做：

```
touch /var/log/sudo
vi /etc/syslog.conf
```

在syslog.conf最后面加一行（必须用tab分割开）并保存：

```
local2.debug                    /var/log/sudo
```

重启日志守候进程,

```
ps aux grep syslogd
```

把得到的syslogd进程的PID输出的第二列是PID填入下面：

```
kill –HUP PID
```

这样sudo就可以写日志了：

```
[foobar@localhost ~]$ sudo ls /rootanaconda-ks.cfg
Desktop install.log
install.log.syslog
$cat /var/log/sudoJul 28 22:52:54 localhost sudo:   foobar :
TTY=pts/1 ; pwd=/home/foobar ; USER=root ; command=/bin/ls /root
```

不过，有一个小小的“ 缺陷”sudo记录日志并不是很忠实：

```
[foobar@localhost ~]$ sudo cat /etc/shadow > /dev/null
cat /var/log/sudo...Jul 28 23:10:24 localhost sudo:   foobar : TTY=pts/1 ;
PWD=/home/foobar ; USER=root ; COMMAND=/bin/cat /etc/shadow
```

重定向没有被记录在案！为什么？因为在命令运行之前shell把重定向的工作做完了sudo根本就没看到重定向。这也有个好处，下面的手段不会得逞：

```
[foobar@localhost ~]$ sudo ls /root > /etc/shadowbash: /etc/shadow: 权限不够
```

sudo有自己的方式来保护安全。以root的身份执行sudo-V，查看一下sudo的设置。因为考虑到安全问题，一部分环境变量并没有传递给sudo后面的命令，或者被检查后再传递的，比如PATHHOMESHELL等。当然，你也可以通过sudoers来配置这些环境变量。