20+ essential Linux security commands

Here are some of the most important security commands for everyday work on Linux systems. If you're interested in security issues on your Linux system, don't ignore these helpful security commands.

There are many aspects to security on Linux systems - from setting up an account to making sure other users don't have more privileges than they need to do the job. Here are some of the most important security commands for everyday work on Linux systems.

Each of the basic Linux commands below is clearly defined with functions and tasks, for example, how to write commands. Some commands are too familiar to longtime Linux users, but there are still quite new commands. If you're interested in security issues on your Linux system, don't ignore these helpful security commands.

22 Linux security commands

1. Sudo command
2. Visudo command
3. Who and w orders
4. Last order
5. Find command
6. File command
7. Order which
8. Ss order
9. Ufw command
10. Iptables command
11. Ip command
12. Ip route command
13. Kill, pkill and killall commands
14. Passwd command
15. Order pwck
16. Setfacl and getfacl commands
17. Order sestatus & apparmor
    1. sestatus
    2. apparmor

Sudo command

Running privileged commands with sudo, instead of switching users to root, is a necessary action because it helps ensure that you only use root privileges when needed and limit the impact of errors. Your access to the sudo command depends on the settings in the **/ etc / sudoers file** and **/ etc / group.**

```
$ sudo adduser shark
Adding user `shark '.
Adding new group `shark '(1007) .
Thêm new user `shark '(1007) v?i nhóm` shark' .
Creating home directory `/ home / shark '.
Sao chép t? t?p tin `/ etc / skel '.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
?ang thay ??i thông tin ng??i dùng cho Shark
Hãy nh?p giá tr? m?i, ho?c b?m ENTER ?? m?c ??nh
Full Name []: shark
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the correct information?  [Y / n] Y
```

For example, if you run sudo and want to find out what your role is, you will receive confirmation that you are running the command as root.

```
$ sudo whoami
root
```

If you manage sudo settings for multiple users, you will also be comfortable with the visudo command.

Visudo command

The visudo command allows you to change the **/ etc / sudoers** file by opening the file in the text editor and checking your changes to the syntax. Run the command 'sudo visudo' and make sure you understand the syntax. Privileges can be assigned by a user or by a group. On most Linux systems, the **/ etc / sudoers file** will be configured with the same groups shown below, allowing the privileges assigned to the groups to be set in the **/ etc / group file.** In those cases, you absolutely do not need to use the visudo command, but just be familiar with the groups that have root privileges this way, and make your updates for the **/ etc / group file.**

```
% admin ALL = (ALL) ALL
% sudo ALL = (ALL: ALL) ALL
% wheel ALL = (ALL: ALL) ALL
```

Note: The group name has the% symbol preceded.

You can display the group that provides access to sudo in your **/ etc / group file** as follows:

```
$ egrep "admin | sudo | wheel" / etc / group
sudo: x: 27: shs, jdoe
```

The easiest way to give someone sudo privileges is to add them to the authorized group in **/ etc / group** . However, that means they can run any command as root. If you want some users to have root access for a certain

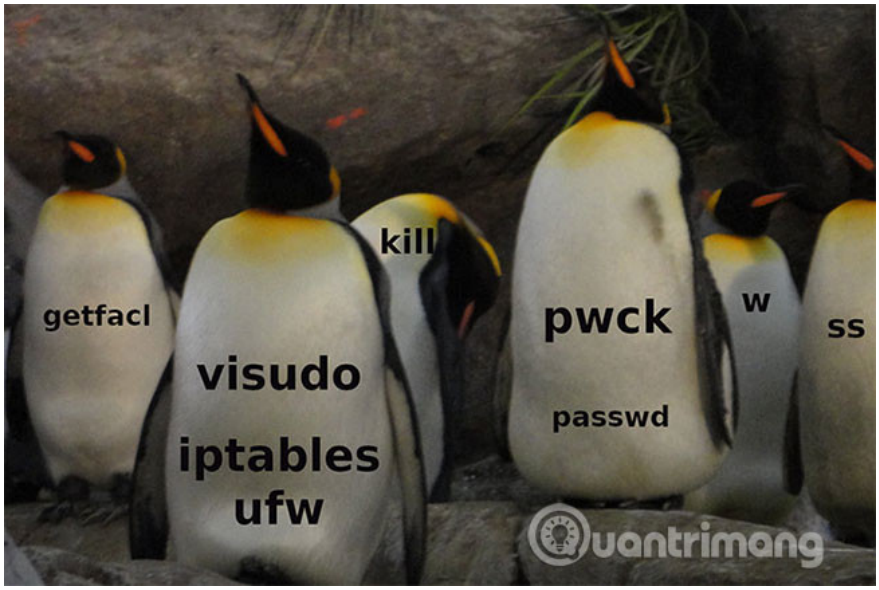
set of commands (for example, adding and deleting accounts), you can specify the commands you want they can run through the following command:

```
Cmnd_Alias ??ACCT_CMDS = / usr / sbin / adduser, / usr / sbin / deluser
```

Then provide the user or user group with the ability to run these commands with sudo with the following command:

```
nemo ALL = (ALL) ACCT_CMDS
% techs ALL = (ALL: ALL) ACCT_CMDS
```

The first line allows the user "Nemo" to run twp commands (adduser and deluser) with sudo, while the second command assigns the same privileges to anyone in the "tech" group in the **/ etc / group file.**



Who and w orders

The who and w command tells you who logged into the system, in which w displayed more information, such as where they logged in, when they logged in and how long they were logged in.

```
$ w
18:03:35 up 9 days, 22:48, 2 users, average load: 0.00, 0.00, 0.00
USER TTY LOGIN FROM IDLE @ JCPU PCPU WHAT
joe tty2 / dev / tty2 27Apr18 9days 7:34 0.09s / usr / lib / x86_64-linux
shs pts / 1 192.168.0.15 09:50 7.00s 0.28s 0.00sw
```

Use the " **sudo update-alternatives - config editor** " command, if you do not like to use the default editor called when running visudo. It will provide some optional editors and change your settings.

Last order

The last command shows you recent user login information and is often useful when you are trying to track other changes or activities.

```
last $ nemo
nemo pts / 1 192.168.0.15 Wed May 2 07:01 - 08:29 (01:27)
wtmp begins Tue May 1 10:21:35 2018
```

'Nemo' has not logged in for a while. He can go on vacation (or maybe go fishing?) Or just left the company. This kind of information can be helpful in deciding whether or not you need to monitor this.

Find command

The find command is used for many types of searches. When it comes to security, you may find yourself looking for files without owners (without corresponding accounts) or files that anyone can write and execute. The find command is easy to implement, but requires some familiar features and many options to determine what you're looking for. The first command in these two commands will find files without a specific owner. The second command will find files that anyone can run and modify.

```
$ sudo find / home -nouser
$ sudo find / -perm -o = wx
```

Remember that -o in the second command refers to the "other" group - not the owner and not the group associated with the file.

File command

The file command looks at a file and determines the file type based on the content of the file, not the file name. The file '.jpg' in the example below obviously is not a real .jpeg file, but an executable file.

```
jdoe @ stinkbug: ~ $ ls -l
total 24
-rw-r - r-- 1 root root 0 Apr 13 09:59 empty
-rwxr-xr-x 1 jdoe jdoe 18840 May 10 17:39 myphoto.jpg
-rwx ------ 1 jdoe jdoe 24 May 2 07:06 trythis
jdoe @ stinkbug: ~ $ file myphoto.jpg
myphoto.jpg: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
```

Order which

The which command specifies the executable file to run when you type its name. This is not always the same as you think. If a Trojan has been inserted into the file system in a location displayed in your search path, it will be run instead. This is a good reason to ensure that your search path includes directories like / **usr** / **bin,** before it adds less standard locations and especially before "." (current directory).

```
$ which date
/ usr / local / bin / date === probably not what we wanted
```

You can check the user's search path by switching to the user role and doing the following:

```
shs @ stinkbug: ~ $ sudo su - nemo
@ nemo stinkbug: ~ $ echo $ PATH
/ usr / local / bin: / usr / bin: / bin: / usr / local / games: / usr / games:
```

Even if the user's search path is set in the system file like **/ etc / profile** or **/etc/bash.bashrc,** they may have been changed by local settings.

```
$ which date
/ usr / local / bin / date === probably not what we wanted
```

Ss order

The ss command is a tool for investigating sockets and allows you to do things like display listening ports and active connections. If you don't add some conditions, ss will display more information than you might want to consider. After all, many parts of the operating system communicate via sockets. If you want to create a list of established connections or listening ports (ie services available to external systems), commands like this will be very useful.

For established connections:

```
$ ss -t
Send-Q Local Recv-Q Local Address: Port Peer Address: Port
ESTAB 0 224 192.168.0.20:ssh 192.168.0.15 m²6647

$ ss |  grep ESTAB |  grep tcp
tcp ESTAB 0 64 192.168.0.20:ssh 192.168.0.15:64885
```

For listening ports:

```
$ ss -ltn
Send-Q Local Recv-Q Local Address: Port Peer Address: Port
LISTEN 0 128 *: 22 *: *
LISTEN 0 5 127.0.0.1:631 *: *
LISTEN 0 50 *: 445 *: *
LISTEN 0 50 *: 139 *: *
LISTEN 0 128 *: 5355 *: *
LISTEN 0 128 ::: 22 ::: *
LISTEN 0 5 :: 1: 631 ::: *
LISTEN 0 50 ::: 445 ::: *
LISTEN 0 128 ::: 5355 ::: *
LISTEN 0 50 ::: 139 ::: *
```

Note : Port 631 (CUPS) only listens on the loopback interface (127.0.0.1).

Ufw command

If you are running a firewall on your Linux system - an important step to control access to the system, the commands are used to start / end, turn on / off, modify and display the status. or operating rules are very important. Here are some sample commands for ufw that you will find on many Ubuntu systems:

```
$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

```
To Action From
- ------ ----
22 ALLOW IN 192.168.0.0/24
```

This firewall is active and only allows connections from the local network to ssh. The following commands will:

1. Set the rules shown above
2. Disable the firewall.

```
$ sudo ufw allow from 192.168.0.0/24 to any port 22
$ sudo ufw disable
```

Iptables command

It is important to know how to list firewall rules for iptables. These commands will give you a complete list of netfilter rules:

```
sudo iptables -vL -t filter
sudo iptables -vL -t nat
sudo iptables -vL -t mangle
sudo iptables -vL -t raw
sudo iptables -vL -t security
```

Ip command

The ip command allows you to display information on your network interfaces. In the example below, we will see loopback (loop) and the public interface.

```
$ ip a
1: worry:
mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link / loopback 00:00:00: 00:00:00 brd 00:00: 00: 00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 :: 1/128 scope host
valid_lft forever preferred_lft forever
2: enp0s25:   mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
link / ether 00: 1e: 4f: c8: 43: fc brd ff: ff: ff: ff: ff: ff
inet 192.168.0.20/24 brd 192.168.0.255 scope global dynamic enp0s25
59794sec valid_lft valid_lft 59794sec
inet6 fe80 :: f233: 4f72: 4556: 14c2 / 64 scope link
valid_lft forever preferred_lft forever
```

Ip route command

The ip route command will display your routing table:

```
$ ip route
default via 192.168.0.1 dev enp0s25 proto static metric 100
```

```
169.254.0.0/16 dev enp0s25 scope link metric 1000
192.168.0.0/24 dev enp0s25 proto kernel scope link src 192.168.0.20 metric 100
```

Kill, pkill and killall commands

Unix and Linux systems provide a convenient option for commands to terminate processes, whenever you want. You can end with the process ID or by its name. You can end each process by one or by group. In any case, different end commands are available for you to use when needed. Examples include:

```
$ kill 1234
$ pkill bad
$ killall badproc
```

Passwd command

The passwd command is probably quite obvious when it comes to system security, but it should not be ignored in any necessary security list. Changing passwords, especially when many users come and go or their roles change, is very important.

However, the passwd command is not only used to change the password. You can also use it with sudo privileges to change other users' passwords, lock / unlock or expire accounts, check account status and change settings to determine password expiration or time password warning.

Check the man page (man passwd) for details and use the following commands:

```
$ sudo passwd nemo == change nemo's password
$ sudo passwd -e dory == expire dory's password (forces her to reset it)
$ sudo passwd -i shark == disable shark's account
```

Order pwck

The pwck command performs checks on your **/ etc / passwd** and **/ etc / shadow files** , ensuring required fields are present, files and directories exist, etc.

```
$ sudo pwck
user 'squash': directory '/ home / squash' does not exist
user 'squash': program '/ bin / bsh' does not exist
```

Setfacl and getfacl commands

Don't let **rwxr-x ---** fool you into thinking that this is all that is needed to authorize files on Linux systems. With the setfacl and getfacl commands, you can grant someone who is not the owner of the file and not a member of the linked group (and you also don't want them to be a member of this group) access into the file. Suppose you want to 'Nemo' have the right to read to a file outlining your ufw setup guidelines and that's it. Use the following command to modify the access control list (ACL) for the file:

```
$ setfacl -mu: nemo: r ufw-setup
```

The getfacl command will then show that the change has been made:

```
$ getfacl ufw-setup
# file: ufw-setup
# owner: shs
# group: shs
user :: rwx
user: nemo: r-- ===
group :: rw- #effective: r--
mask :: r--
other :: ---
```

Order sestatus & apparmor

The sestatus and apparmor commands can display the status of SELinux tools and apparmor, providing isolation between applications that use mandatory access control. If you are using one or more tools, you should know how to display their status.

sestatus

```
$ sudo sestatus
SELinux status: enabled
SELinuxfs mount: / sys / fs / selinux
SELinux root directory: / etc / selinux
Loaded policy name: targeted
Current mode: enforcing
Mode from config file: enforcing
Policy MLS status: enabled
Policy deny_unknown status: allowed
Max kernel policy version: 28
```

apparmor

```
$ sudo apparmor_status
apparmor module is loaded.
18 profiles are loaded.
18 profiles là trong ch? ?? ch? ??.
/ sbin / dhclient
/ usr / bin / evince
/ usr / bin / evince-previewer
/ usr / bin / evince-previewer // sanitized_helper
/ usr / bin / evince-thumbnailer
/ usr / bin / evince-thumbnailer // sanitized_helper
/ usr / bin / evince // sanitized_helper
/usr/lib/NetworkManager/nm-dhcp-client.action
/ usr / lib / NetworkManager / nm-dhcp-helper
/ usr / lib / connman / scripts / dhclient-script
/ usr / lib / cups / backend / cups-pdf
/ usr / lib / snapd / snap-confine
/ usr / lib / snapd / snap-confine // mount-namespace-capture-helper
/ usr / sbin / cups-browsed
/ usr / sbin / cupsd
```

```
/ usr / sbin / cupsd // third_party
/ usr / sbin / ippusbxd
/ usr / sbin / tcpdump
0 profiles are in complain mode.
3 processes have profiles defined.
3 processes are in enforce mode.
/ sbin / dhclient (705)
/ usr / sbin / cups-browsed (30173)
/ usr / sbin / cupsd (26828)
0 processes are in complain mode.
0 các ti?n trình không ???c xác ??nh, nh?ng có xác ??nh profile.
```

You should also know how to start and stop these tools.

```
$ sudo /etc/init.d/apparmor start
$ sudo /etc/init.d/apparmor stop
$ sudo /etc/init.d/apparmor restart
```

For SELinux, different modes represent:

```
enforcing - SELinux security policy is enforced
permissive - SELinux prints warnings instead of enforcing
disabled - SELinux is fully disabled
```

Many commands on Linux systems can help you manage security. The above descriptions only introduce these commands, but do not explain everything about how they work or can be used.

1. Useful commands in Unix / Linux
2. 14 interesting Linux commands in Terminal