- http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-es-4/s1-wstation-privileges.html
- http://www.rpublica.net/sudo/sudoers.html
- http://www.linuxtotal.com.mx/index.php?cont=info_admon_014

# El comando sudo

El comando sudo ofrece otra solución para otorgar acceso administrativo a los usuarios. Cuando un usuario de confianza antecede un comando administrativo con sudo, se le pide su propia contraseña. Luego, una vez autenticado y asumiendo que el comando es permitido, el comando administrativo es ejecutado como que si se tratase del superusuario.

El comando sudo permite un gran nivel de flexibilidad. Por ejemplo, solo los usuarios listados en el archivo de configuración /etc/sudoers tienen permitido utilizar el comando sudo y el comando es ejecutado en el shell del usuario, no en el shell de root

El comando sudo también proporciona un rastro completo para auditoría. Cada autenticación exitosa es registrada al archivo /var/log/messages y el comando emitido junto con el nombre del usuario se registran al archivo /var/log/secure.

Otra ventaja del comando sudo es que un administrador puede permitir a usuarios diferentes acceso a comandos específicos basado en sus necesidades.

Los administradores que deseen modificar el archivo de configuración de sudo, /etc/sudoers, deberían usar el comando visudo.

Para otorgarle a un usuario privilegios administrativos completos, escriba visudo y añada una línea similar a la siguiente en la sección de especificación de privilegios del usuario:

```
juan ALL=(ALL) ALL
```

Este ejemplo establece que el usuario, juan, puede utilizar sudo desde cualquier máquina y ejecutar cualquier comando.

El ejemplo de abajo ilustra la posible granularidad cuando se configura sudo:

```
%users  localhost=/sbin/shutdown -h now
```

Este ejemplo establece que cualquier usuario puede emitir el comando /sbin/shutdown -h now siempre y cuando sea emitido desde la * consola.

## Comandos de ejemplo

- Con **sudo -l** podemos saber que comando puedo ejecutar
- con **sudo -u otrousuario /comando/de/otrousuario** podemos también ejecutar comandos de otro usuario del sistema
- con **sudo -e /etc/inittab** permite con la opción -e editar un fichero como root
- con **sudo -L** podemos ver las opciones que se pueden establecer
- con **sudo -V** listamos las opciones establecidas por defecto para todos los usuarios

# Estructura de /etc/sudoers

Podemos considerar tres grandes secciones en /etc/sudoers:

- Definiciones de alias
- Ajuste de las opciones por defecto
- Reglas de acceso

Las tres secciones pueden considerarse opcionales, aunque sin la tercera ( Reglas de acceso) la existencia misma de /etc/sudoers carecería de sentido.

A continuación vamos a ver con detalle cada una de las mencionadas tres secciones.

## Definiciones de Alias

Con los alias nos referimos a cualquier tipo de elemento: comandos , usuarios ( normales o privilegiados) y hosts .Desde el momento en que hemos definido un alias para un elemento, puede utilizarse en cualquier lugar, incluida la definicion de un alias , en el que se espere un usuario, un comando o un host.

La forma general de una definición de alias es:

```
Tipo_Alias   NOMBRE_ALIAS  =  elemento1,  elemento2,  elemento3,  ......
```

Tipo_Alias Puede ser uno de los siguientes:

- Cmnd_Alias: Para definir alias de comandos.
- User_Alias: Para definir alias de usuarios “normales”.
- Runas_Alias: Para definir alias de usuarios “privilegiados”.
- Host_Alias: Para definir alias de hosts.

NOMBRE_ALIAS Es el nombre que daremos al alias. Para su composición deben seguirse unas simples normas:

- Debe ser una cadena compuesta exclusivamente por letras mayúsculas,números y guiones_bajos.
- La cadena debe comenzar necesariamente por una letra mayúscula.
- Se recomienda que en los nombres de alias se utilicen solo letras mayúsculas. En adelante,los alias se distinguirán por la utilización exclusiva de letras mayúsculas en su nombre.

elemento Son los elementos o listas de elementos para los cuales será expandido el NOMBRE_ALIAS.

- Cuando se trate de una lista, deben separarse los elementos constitutivos de la misma mediante comas (,)
- Un alias puede ser un elemento en otra definición de alias.

Es posible definir varios alias de un mismo tipo en una sola línea. Las diversas definiciones deben separarse con dos puntos (:)

```
Tipo_Alias  NOMBRE_1  =  elemento1, elemento3  :  NOMBRE_2  =   elemento2,\
```

```
elemento4  : NOMBRE_3  =  elemento5, elemento6
```

Esto equivaldria a :

```
Tipo_Alias  NOMBRE_1 = elemento1, elemento3
Tipo_Alias  NOMBRE_2 = elemento2, elemento4
Tipo_Alias  NOMBRE_3 = elemento5, elemento6
```

Como se ha dicho anteriormente, un alias puede ser un elemento en la definición de otro alias ( del mismo tipo). Así, es perfectamente correcta una sintaxis del siguiente tipo:

```
Tipo_Alias  NOMBRE_5 = elemento7, NOMBRE_3 : NOMBRE_6 = elemento8,\
elemento9, NOMBRE_1
```

La única restricción es que los alias utilizados como elementos en las definiciones de otros alias ( en nuestro caso, NOMBRE_3 y NOMBRE_1 ) deben haber sido definidos previamente. Y, por supuesto, deben ser del mismo tipo del alias que se define.

Veamos ahora como construir una definición de alias para cada uno de los Tipo_Alias :

### Alias de Usuarios: User_List

```
User_List  NOMBRE_ALIAS  =  Lista_Usuarios

     Lista_Usuarios:  elemento1, elemento2, ......
          elemento :
                 nombre_usuario
                 %nombre_grupo  (para system groups)
                 nombre_grupo  (para net groups)
                 ALIAS_USER
```

Como puede verse, la lista de usuarios puede estar formada por uno o más nombres de usuarios, nombres de grupos ( precedidos por % en el caso de “system groups” o por + en el caso de “net groups” ) y otros alias.

Un ejemplo de ello seria la siguiente definición:

```
User_List INVITADOS = asd53, delfin, %impresora, +robinsones, REDACTORES
```

Con la anterior definición establecemos que el alias INVITADOS comprende a los usuarios asd53 y delfin, a los usuarios del grupo ( system-group) impresora, a los usuarios del grupo (net-group) robinsones y a los usuarios definidos en el alias REDACTORES.

#### Alias de Usuarios Privilegiados: Runas_List

```
Runas_List  NOMBRE_ALIAS  =  Lista_Runas
     Lista_Runas :  elemento1, elemento2, .....
          elemento:
```

```
                nombre_runas
                %nombre_grupo  (para system groups)
                +nombre_grupo  (para net groups)
                #uid
                ALIAS_RUNAS
```

Como puede observarse, las Runas_List son similares a las User_List, salvo en que las Runas_list pueden contener uid, con el prefijo #.

#### Alias de Hosts : Host_List

```
Host_List  NOMBRE_ALIAS  =  Lista_hosts

    Lista_hosts :  elemento1, elemento2, .....
          elemento :
               nombre_host
               dirección IP
               números de red  (/máscara)?
               +nombre_grupo  (para net groups)
               ALIAS_HOST
```

El nombre_host puede incluir carácteres comodín (wildcards)

Si no se especifica una máscara de red con un número de red, se utilizará la máscara de la interface ethernet del host.

#### Alias de Comandos : Cmnd_List

```
Cmnd_List  NOMBRE_ALIAS  =  Lista_comandos

     Lista_comandos :  elemento1, elemento2, .....
          elemento :
              nombre_comando
              nombre_comando argumentos
              nombre_comando wildcards
              directorio
              sudoedit
              ALIAS_COMANDO
```

Una Cmnd_List es una lista de uno o más nombres de comandos, directorios y/o otros alias.

Cuando se especifique un comando, siempre debe ponerse la ruta completa al ejecutable:

```
Cmnd_List BASICOS = /bin/ls , /usr/bin/lpr
```

```
Cmnd_List AVANZADOS = /bin/kill, /bin/passwd, BASICOS
```

Cuando el nombre del comando se especifica con argumentos (incluyendo wildcards), el usuario sólo podra utilizar los argumentos especificados.

Si el nombre del comando se especifica con el “argumento” guión_bajo, _ , se indica que el comando sólo puede ejecutarse sin argumentos.

Un directorio es un path, completo y válido, acabado en /.

/usr/local/bin/

Cuando en lugar de un comando se especifica un directorio, el usuario puede ejecutar cualquier binario que se encuentre en el directorio epecificado.

En el siguiente ejemplo, el usuario podria ejecutar los comandos kill, passwd, hostname, todos los ejecutables contenidos en el directorio /usr/local/bin/ y los comandos correspondientes al alias BASICOS :

```
Cmnd_List AVANZADOS = /bin/kill, /bin/passwd, /bin/hostname, BASICOS ,
/usr/local/bin/
```

Los siguientes caracteres deben ser “escapados” con \ en el caso de ser utilizasos en los argumentos del comando: ' : , = \

El comando especial sudoedit se utiliza para permitir al usuario la ejecución de sudo -e. Como cualquier comando, puede tambien especificarse con argumentos.

## Ajustes de Opciones (Defaults)

Pueden modificarse ciertas opciones de configuración mediante una o más líneas Defaults en el archivo /etc/sudoers.

Podemos definir las opciones:

1. globalmente
2. por usuario
3. por usuario privilegiado
4. por host

Y sus respectivas sintaxis serian las siguientes:

1. **Defaults** lista_opciones
2. **Defaults:usuario** lista_opciones
3. **Defaults>usuario_privilegiado** lista_opciones
4. **Defaults@host** lista_opciones

Las listas de opciones estan contituidas por un conjunto de opciones separadas por comas:

```
opcion1, opcion2, opcion3,........
```

Podemos considerar cuatro tipos de opciones:

- Booleanos :
    - Se activan escribiendo el nombre de la opcion.
    - Se desactivan colocando el operador ! delante el nombre de la opción.

```
Defaults>root !set_logname
```

En el ejemplo anterior, se desacttiva la opción set_logname para el usuario root

```
Defaults:sebastian authenticate
```

En el ejemplo anterior, se activa la opción authenticate para el usuario sebastian

- Enteros :

Toman la forma: nombre_opcion = valor

- Cadenas :

Toman la forma: nombre_opcion = cadena

```
Defaults@SERVERS logfile=/var/log/sudo.log ,log_year
```

- Listas :

Toman la forma: nombre_opcion = valor1,valor2, ....

```
        En las listas el operador  =  puede ser sustituido por +=  o  -=,
para añadir o quitar elementos.
```

Los valores o cadenas, cuando contengan varias palabras, pueden encerrarse entre dobles comillas ("...").

Los caracteres especiales pueden "escaparse" con \

Debemos tener presente que algunas opciones no booleanas pueden ser utilizadas en un contexto booleano, lo que permite, entre otras cosas, que puedan ser deshabilitadas con el operador !

En [Opciones de Configuración (Defaults)] puede encontrarse una relación de las diversas opciones utilizadas en la configuración de sudoers con una breve explicación de cada una de ellas.

## Reglas de Acceso (User Specification)

Mediante las Reglas de Acceso vamos a definir:

- Los usuarios a los que permitimos utilizar sudo.
- Los comandos que dichos usuarios podrán ejecutar.
- En calidad de qué usuarios privilegiados podrán ejecutarlos.
- En que hosts podrán hacerlo.

La sintaxis básica es:

```
usuario   host = (usuario_privilegiado)    comando
```

Cada uno de los elementos anteriores puede ser un Alias o una lista de elementos.

El elemento *usuario_privilegiado* es opcional. Por defecto se toma *root* . Un ejemplo de regla de acceso con root como usuario_privilegiado seria:

```
USUARIO    HOST = comando1, comando2,.....
```

Cuando se especifique un comando siempre debe ponerse la ruta completa al ejecutable. (Cuando ejecutemos sudo no es necesario, bastará con el nombre del ejecutable).

```
USUARIO    HOST = /usr/bin/ls
```

En ejemplo anterior, los usuarios de alias USUARIO, en las máquinas de alias HOST podrian ejecutar, como root el comando ls

Como se comentó anteriormente [ir..] , debemos tener en cuenta que en lugar de ejecutables pueden configurarse rutas a directorios para indicar que cualquier binario de dicho directorio se incluye en la Regla de Acceso; bastará con finalizar la ruta con / y se entenderá como un directorio. Un ejemplo:

```
USUARIO   HOST = (operador) /usr/local/bin/
```

En este ejemplo los usuarios de alias USUARIO, en las máquinas de alias HOST podrian ejecutar con los privilegios del usuario operador los ejecutables que se hallen en el directorio */usr/local/bin/*.

Pueden especificarse mas de un usuario_privilegiado. En este caso, cada usuario_privilegiado especificado toma efecto a todos los comandos especificados a partir de el. Veamos un eljemplo:

```
USUARIO  HOST = (operador) comando1, comando2, (root) comado3, comando4
```

En la anterior regla de acceso se autoriza a los usuarios de alias USUARIO, en las máquinas de alias HOST ejecutar los comandos comando1 y comando2 como usuario operador y los comandos comando3 y comando4 como usuario root . Fíjate que en este caso se obtendria el mismo resultado con la siguiente regla de acceso:

```
USUARIO  HOST = comando3,comando4, (operador) comando1, comando2
```

## Etiquetas de Comando

Por si no fuera suficiente, tenemos un medio de modificar las condiciones en que el usuario puede ejecutar las disversos comandos definidos. Es mediante las llamadas ETIQUETAS DE COMANDO.

Un comando puede definirse con cero o más etiquetas asociadas a el.

En el caso de utilizar ETIQUETAS DE COMANDO, la sintaxis de la Regla de Acceso pasaria a ser:

```
usuario host = (usuario_privilegiado)  ETIQUETA:comando, ...
```

Disponemos de cuatro valores posibles para ETIQUETA:

- NOPASSWD
- PASSWD
- EXEC
- NOEXEC

Una vez se ha asociado una etiqueta a un comando, los siguientes comandos de la lista de comandos heredan el valor de la etiqueta asociada, salvo que sea “anulada” por una etiqueta con valor opuesto.

Las parejas de valores “opuestos” son:

- NOPASSWD – PASSWD
- EXEC – NOEXEC

#### Etiquetas NOPASSWD - PASSWD

Por defecto, sudo requiere que el usuario se autentifique mediante su contraseña antes de ejecutar un comando. Esto puede modificarse mediante la etiqueta NOPASSWD.

Como se ha comentado anteriormente, al aplicar la etiqueta NOPASSWD a un comando de una lista, todos los comandos subsiguientes “heredarán” esta propiedad. Podemos desactivar la “herencia” con PASSWD . Veamos esto con un ejemplo:

```
jose1 hosto = (operador) NOPASSWD: /bin/kill, /bin/ls, /usr/bin/lpr
```

La anterior definición nos dice que el usuario jose1, en el host hosto, en calidad de usuario_privilegiado operator, puede ejecutar sin necesidad de autentificarse los comandos /bin/kill, /bin/ls, y /usr/bin/lpr, .

Si deseáramos que solo pudiera ejecutar /bin/kill sin necesidad de autentificarse, desactivariamos NOPASSWD para /bin/ls y /usr/bin/lpr, mediante PASSWD :

```
jose1 hosto = (operador) NOPASSWD: /bin/kill, PASSWD: /bin/ls, /usr/bin/lpr
```

Si quisiéramos que /bin/ls fuera el único comando que pudiera ejecutarse sin autentificación, la sintaxis deberia ser:

```
jose1 hosto = (operador) /bin/kill, NOPASSWD: /bin/ls, PASSWD: /usr/bin/lpr
```

#### Etiquetas NOEXEC - EXEC

Si sudo ha sido compilado con soporte noexec y este es soportado por el sistema operativo , la etiqueta NOEXEC puede utilizarse para prevenir shell escapes.

# II.Caracteres Comodín (Wildcards)

Sudo permite la utilización de caracteres comodín de shell (shell-style wildcards) en los pathnames y en los argumentos de comandos, utililizados en el archivo /etc/sudoers.

Los comodines utilizados son:

- *
- ?
- [….]
- [!…]

Se utiliza también \ para escapar caracteres especiales

El caracter \ , como parte de un path, no puede ser sustituido por un comodin:

/usr/bin/* hará referencia a todos los archivos existentes en /usr/bin/ , pero no “expandirá” el path a , por ejemplo, /usr/bin/X11/xterm .

# III.Caracteres Especiales

El signo # es utilizado para indicar comentarios. Cada linea de comentarios debe comenzar por un signo # .

El signo ! puede utilizarse como un operador lógico de negación (not), tanto en alias como en comandos.

Las lineas largas pueden continuarse en otra linea inferior, colocando \ como último caracter de la linea que se desea interrumpir.

Los espacios en blanco entre elemenos de una linea ; incluyendo los caracteres sintacticos especiales tales como = , : ( ) , son opcionales.

Los siguientes caracteres especiales deben ser escapados con \ cuando se utilicen como parte de una palabra: @ ! = : ( ) \ ,

# IV.Prevención de Shell-Escapes

Una vez sudo ejecuta un perograma, dicho programa es libre de realizar cualquier operación que le sea propia, incluida la ejecución de otro programa. Esto puede plantear una cuestión de seguridad, ya que no es infrecuente que un programa permita shell escapes, lo que posibilita al usuario “puentear” las restricciones de sudo .

Muchos sistemas que usan bibliotecas compartidas poseen la capacidad de anular funciones por defecto de las bibliotecas apuntando una variable de entorno ( normalmente, LD_PRELOAD) a una

biblioteca compartida alternativa. En tales sistemas, la funcionalidad noexec de sudo puede utilizarse para prevenir que con un programa que se haya ejecutado através de sudo pueda ejecutarse cualquier otro programa. Debe tenerse en cuenta pero que esto es solo posible con ejecutables dinamicamente enlazados (dynamically-linked). Con ejecutables estaticamente enlazados ( statically-linked) no tiene efecto.

Para saber si sudo soporta noexec , debemos ejecutar, como root :

```
sudo -V | grep "dummy exec"
```

Si la salida contiene una linea que comienza por:

File containing dummy exec functions:

quiere decir que sudo es capaz de reemplazar ls familia de funciones exec en la biblioteca estandar,devolviendo simplemente un error. Por desgracia, no existe un medio seguro de saber si noexec se ejecutará durante el proceso de compilación. Como noexec suele ejecutarse con eficacia en la mayoria de sistemas que soportan la variable de entorno LD_PRELOAD, en el manual del sistema podremos comprobar si la citada variable y deducir el posible comportamiento de noexec .

Para activar noexec para un comando, utilizar la etiqueta de comando NOEXEC .

En el siguiente ejemplo habilitamos al usuario asd53 , sobre el host host3 ejecurar como root los comandos les y pine , con noexec activado:

asd53 host3 = NOEXEC : /usr/bin/less , /usr/bin/pine

Esto hace que los ateriores dos comandos no puedan ser utilizados para la ejecución de otro comando.

Debemos tener en cuenta que el desactivar el shell escape no es ni mucho menos la panacea. los programas ejecutados como root son capaces de realizar otras operaciones tanto o mas peligrosas ( cambiar o sobrescribir archivos, por ejempo.....)

# Ejemplo

```
# ***********************
# LinuxTotal.com.mx, ejemplo de un archivo sudoers
# sergio.gonzalez.duran@gmail.com
# ***********************

# ***********************
# DEFINCION DE ALIAS
# ***********************

# administradores con todos los privilegios
User_Alias ADMINS = sergio, ana

# administradores de red - network operators
User_Alias NETOPS = marcela, andrea
```

```
# webmasters -
User_Alias WEBMAS = cristina, juan

# supervisores de producción (todos los del grupo de sistema supervisores)
User_Alias SUPPRO = samuel, %supervisores

# usuarios que pueden conectarse desde Internet
User_Alias INETUS = NETOPS, ADMINS, samuel

# servidores web
Host_Alias WEBSERVERS = 10.0.1.100, 10.0.1.101

# servidores de aplicaciones
Host_Alias APLICACIONES = WEBSERVERS, 10.0.1.102, 10.0.1.103, mailserver

# comandos de red permitidos
Cmnd_Alias REDCMDS = /sbin/ifconfig, /sbin/iptables

# comandos de apache
Cmnd_Alias APACHECMDS = /usr/sbin/apachectl, /sbin/service httpd *

# ***********************
# DEFINCION DE OPCIONES
# ***********************

# Los usuarios administradores, requieren autentificarse con la contraseña
de 'root'
Defaults>ADMINS rootpw

# Para todos los usuarios, tienen hasta dos intentos para ingresar su
contraseña y 3 minuto para que esta expire
Defaults passwd_tries = 4, passwd_timeout = 1

# Los usuarios que se conectan desde Internet, solo tienen una oportunidad y
cero timeout lo que implica
# que cada comando que usen a través de sudo requerira siempre de
autentificación.
Defaults:INETUS passwd_tries = 1, passwd_timeout = 0

# Máscara de directorios y archivos por default, para los que ejecuten sudo
en los servidores web
Defaults@WEBSERVERS umask = 022

# ***********************
# DEFINCION DE REGLAS
# ***********************

# administradores todo se les permite en cualquier equipo (¡¡¡¡¡cuidado con
esto en la vida real!!!!!
ADMINS ALL = (ALL) ALL
```

```
# administradores de red, en todos los equipos, los comandos de red
NETOPS ALL = REDCMDS

# webmasters, en los servidores web con los comandos indicados en apachecmds
y además sin necesidad
# de contraseña acceder a las bítacoras de apache y reiniciar los
servidores.
WEBMAS WEBSERVERS = APACHECMDS, NOPASSWD: /var/log/apache/, /sbin/reboot

# supervisores, pueden ejecutar los comandos indicados en los equipos
indicados en el alias
# aplicaciones y además son ejecutados bajo el usuario apps.
SUPPRO APLICACIONES = NOEXEC: (apps) /usr/local/facturacion.exe,
/usr/local/ventas.exe, /usr/local/nomina.exe

# no definidos por alias previos, sino directamente

# regina es de recursos humanos y puede cambiar contraseñas de cualquier
usuario menos de root
regina ALL = /usr/bin/passwd *, !/usr/bin/passwd root

# david, puede apagar los equipos de aplicaciones
david APLICACIONES = /sbin/shutdown, /sbin/halt

#  El equipo firewall de la red puede ser reiniciado (no apagado) por
fernanda que es asistente de redes
fernanda firewall = /sbin/shutdown -r now
```

From:
http://wiki.intrusos.info/ - **LCWIKI**

Permanent link:
**http://wiki.intrusos.info/doku.php?id=linux:comandos:sudo**

Last update: **2023/01/18 14:36**