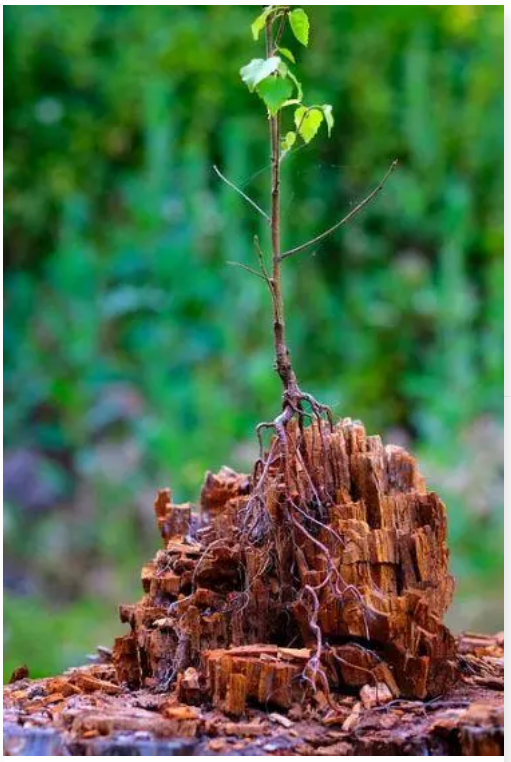
Safely Use Root Commands Protection against Physical Attacks

User Account Isolation (developers)

This page gives tips on using `sudo` / root (privileged) commands safely. The root account is a special user account in Unix-based operating systems that has complete access to all files and commands on a system. It is typically used only for system administration tasks that require unrestricted access.



2
1

Contents

collapse ↑

Introduction

Learn why 'root' is important, what it is used for, and how to use `sudo` / root (privileged) commands safely.

Rationale for Protecting the Root Account

What is the point on a typical single user Linux desktop computer of separating privileged administrator account (called `root` account) and limited user accounts (such as for example user `user`)?

It is assumed that most desktop computer users are single user computers. I.e. computers being used by only one person. Rather it is assumed that most users are only using a single login user account which will be referred to as user `user`.

Quote xkcd authorization (https://xkcd.com/1200/) :

> If someone steals my laptop while I'm logged in, they can read my email, take my money, and impersonate me to my friends, but at least they can't install drivers without my permission.

Quote (https://askubuntu.com/questions/16178/why-is-it-bad-to-log-in-as-root) user discussion:

> Most people will consider their home directory as more important than root dirs

> Once a malicious program has access to my home folder, I dont care if it also has access to the admin content

This is true for most users using single user computers, using only one user account and no virtual machines. As a counter measure this is why this documentation recommends compartmentalization, that is, running different activities in different virtual machines or even on different hardware.

The rationale of prevention of root compromise has the following goals: [1]

- Protect the host operating system: If using virtual machine (VM): It is much less likely that malware will break out of a virtual machine if it does not have root access within the VM. [2]
- Protection from rootkits: Root access allow malware to install rootkits, which can be very difficult to detect and remove.
- Protect the virtualizer: It is harder to attack the virtualizer without root access. (Applies only when using virtual machines.)
- Protect the hardware: A compromised host operating system might result in malware infecting the hardware, i.e. malware could install a persistent hardware backdoor (such as in BIOS or other firmware)

surviving even re-installation of the host operating system. In many cases, root access is required before hardware can be attacked. [3]

- Protect against compromised non-root users: it is harder for potentially compromised non-root users (such as `www-data`) to access user `user` or other parts of the system. This is important when considering that even single-user systems have many system-level user accounts.
- Sandboxing: Sandboxing applications can prevent applications getting exploited by attackers [4] or limit the severity of the exploit since if sandboxing is successfully, malware will be trapped inside the sandbox. Sandboxing is a lot harder, less efficient or even impossible when applications are running as root. See also AppArmor, `apparmor.d` (Full System AppArmor Profile) and sandbox-app-launcher.

Kicksecure implements various security hardening to Enforce Strong Linux User Account Isolation.

Once proposal Multiple Boot Modes for Better Security (an Implementation of Untrusted Root) has been implemented there will be a strong guidance for users to better separate their limited (everyday use) account (`user`) and administrative account (`admin`). This would result in a robust Prevention of Malware Sniffing the Root Password.

Default Passwords

The default passwords for Kicksecure are:

- **Default username:**
  ```
  user
  ```
- **Default password:** No password required. (Passwordless login.) [5]

The default root account is locked (or should be locked). [6] This is a purposeful security feature -- see below for further details.

Users can change or set a password for security reasons if this is useful in their case based on this Information.

Running with Root Access

Run with Administrative Rights

To run an application with root rights.

1. Use a privilege elevation utility to run commands as root.

Note: Replace `command` with the actual command.

- command line interface (CLI): `sudo command`
- graphical user interface (GUI): See Graphical Applications with Root Rights.

2. Password entry.

If a password has been configured, the utility will prompt for it.

User Password versus Root Password

Tools such as sudo and lxsudo prompt for the user's password. This is different from the root password.

Security and Best Practices

General Security Advice

Commands that require root permissions should be run individually using `sudo`. In all cases:

- Do not login as root.
- Do not run `sudo su`.

Inappropriate Use of Root Rights

Do not think of root as a shortcut to fix issues.

It is very much discouraged to establish the following behavior:
`application problem` → "try `sudo` / root".

Only use `sudo` / `lxsudo` / root if there is a strong rationale for doing so. Otherwise...

Inappropriate Use of Root Rights:

- Can cause additional non-security related issues. [7] Related is also later chapter Graphical Applications with Root Rights.
- Risks harmful code being run as root.

Graphical Applications and Root Rights

1 Do not run graphical user interface (GUI) with `sudo` !

It is discouraged to run graphical user interface (GUI) (https://www.computerhope.com/jargon/g/gui.htm) applications with `sudo gui-application`.

- Never login as `root` as explained above.
- This includes, never use `sudo su` and then start GUI applications.

Doing so would be an Inappropriate Use of Root Rights. That would fail in many cases and is a limitation inherited from Debian. If this action is attempted, error messages like those below will appear. [8]

```
No protocol specified
```

```
cannot connect to X server :0
```

2 If there is a legitimate reason to start GUI applications with root rights, use `lxsudo` instead. [9]
Syntax:

```
lxsudo application-name
```

For example to start the partition manager `gparted` by default with root rights.

```
lxsudo gparted
```

[10]

3 To edit files which can only be edited with root rights. Use the following syntax.

Note: Replace `/path/to/file/name` with the actual path to the file.

Open file `/path/to/file/name` in an editor with root rights.

Kicksecure Kicksecure for Qubes Others and Alternatives

See Open File with Root Rights for detailed instructions on why to use `sudoedit` for better security and how to use it.

```
sudoedit /path/to/file/name
```

For example:

Open file `/etc/default/keyboard` in an editor with root rights.

Kicksecure Kicksecure for Qubes Others and Alternatives

See Open File with Root Rights for detailed instructions on why to use `sudoedit` for better security and how to use it.

```
sudoedit /etc/default/keyboard
```

Polkit PolicyKit pkexec for GUI applications

Use of Polkit (https://en.wikipedia.org/wiki/Polkit) (formerly PolicyKit) (`pkexec`) might also be appropriate for running GUI applications with root rights. Usually such applications should have desktop shortcuts or wrappers which make use of `pkexec` . There are no (or rare) known cases where users need to run `pkexec` on the command line.

Wayland

Running GUI applications as root (`lxsudo` , `sudo --set-home` , `pkexec` will be more difficult once Kicksecure has been ported from X11 to Wayland. This is because Wayland requires applications to access `$XDG_RUNTIME_DIR/$WAYLAND_SOCKET` to open GUI windows. [11] You may run into this when using Other Desktop Environments using Wayland.

The following command should work to run Wayland applications as root in most instances. Replace `application-name` as appropriate:

```
sudo XDG_RUNTIME_DIR=$XDG_RUNTIME_DIR application-name
```

Root Account Management

Enable Root Account

For security reasons the root account is locked and expired by default in Kicksecure `15.0.0.3.6` and above. For most users there should be no need to use the root account. If it must be enabled for some reason, run the following commands.

(Kicksecure for Qubes: Kicksecure Template)

If you can use `sudo` , you can skip the following box.

If you cannot use `sudo` :

- Kicksecure: Boot into recovery mode.
- Kicksecure-Qubes: Open a Qubes Root Console.

1 Unexpire the root account.

```
sudo chage --expiredate -1 root
```

2 Set a root password.

According to the Change Password instructions. Note: These instructions are for the user `user` account. Replace user `user` with `root` .

Disable Root Account

Earlier versions of Kicksecure (version numbers lower than `15.0.0.3.6`) come with the root account enabled by default. Most users should disable it by running the following commands.

(Kicksecure-Qubes: Kicksecure Template).

Lock the account.

```
sudo passwd --lock root
```

[12]

In the future, use `sudo` instead when it is necessary.

Avoid Root Login

Should the user login as `root` ? No. See footnote for rationale. [13]

Troubleshooting

Permissions Fix

After inappropriate use of root rights, attempt to fix:

Open a terminal.

If you are using Kicksecure inside Qubes, complete the following steps.

`Qubes App Launcher (blue/grey "Q") → Kicksecure App Qube (commonly named kicksecure) → Xfce Terminal`

If you are using a graphical Kicksecure with Xfce, run.

`Start Menu → Xfce Terminal`

Run the following command to reset permissions of user `user` 's home folder `/home/user` back to owner `user` and group `user` .

```
sudo chown --recursive user:user /home/user
```

Reset User Account Password

The following steps can be used in case the password has been forgotten and needs to be reset.

1 Launch a root terminal.

- Kicksecure: Boot into recovery mode.
- Kicksecure-Qubes: Open a Qubes Root Console.

2 Notes.

- This process will be similar to the change password wiki chapter which is recommended to read as it contains instructions / links on how to change and test the keyboard layout.
- This is unspecific to Kicksecure. It should be a very similar process on Debian or most other Linux distributions. It can also be resolved as per Self Support First Policy.

3 Set a new password.

Run the following command.

```
sudo passwd user
```

4 Reboot.

```
sudo reboot
```

5 Done.

The process of password reset has been completed.

Unlock User Account: Excessive Wrong Password Entry Attempts

The following steps can be used in case the user entered the wrong password too many times, which resulted in the user account being automatically locked. (This is related to security feature Bruteforcing Linux User Account Passwords Protection.)

1 Launch a terminal that can run commands as root.

If you cannot login anymore, see the bullet points below:

- Kicksecure: Boot into recovery mode.
- Kicksecure-Qubes: Open a Qubes Root Console.

2 Run the following command.

Note: Replace `user` with the actual name of the user that you wish to unlock.

```
sudo faillock --user user --reset
```

3 Done.

Unlocking of user account has been completed.

Console Unlock

1 Launch a root terminal.

- Kicksecure: Boot into recovery mode.
- Kicksecure-Qubes: Open a Qubes Root Console.

2 Run the following command.

Note: Replace `user` with the Linux user account name which should be allowed to login on the login console.

```
sudo adduser user console
```

Advanced Users

Prevent Malware from Sniffing the Root Password

Rationale for Separate admin Account

If Linux user account `user` is compromised, malware can easily steal the administrative (" `sudo` ") password. [14] Therefore it is more secure (rationale) to perform administrative actions such as running `sudo` from a separate `admin` account that is less likely to get compromised, since this reduces the chances of malware sniffing the password to escalate to administrative ("root") access.

The basic concept is a separation of the following users:

- **user** `user` **:** Perform everyday actions such as running web browsers.
- **user** `admin` **:** Perform administrative actions such as installing additional packages.

Questions and answers:

- Is running applications such as browser under user `admin` less secure? Yes, that defeats this concept.
- Is running applications such as browser under user `user` more secure? Yes, because it becomes harder for malware to perform privilege escalation attacks to gain to administrative ("root") access.
- What is so bad about malware escalating to administrative? See rationale.
- Why use user `admin` and not simply the `root` user? See Avoid Root Login.

This process is currently for advanced users only since it is quite cumbersome, i.e. has bad usability. The usability of this will be improved once proposal Multiple Boot Modes for Better Security (an Implementation of Untrusted Root) has been implemented.

Overview of Steps

To more securely perform administrative tasks that require root access, see the following overview steps below. Detailed technical steps are available further below.

1. Prerequisite knowledge: how to switch to a different virtual console, usage of the SysRq key and login spoofing.

2. These instructions are ideally applied after installing the host / VM when it is still considered free of malware.
3. Create a new user account `admin`.
4. Add it to the group `sudo`.
5. Login as user `admin`.
6. Remove user `user` from group `sudo`.
7. Only then perform administrative tasks according to the instructions below.

Detailed Steps

This setup only needs to be completed once.

1 Platform specific notice:

- Kicksecure: No special notice.
- (Kicksecure-Qubes: Kicksecure Template)

2 Create a new user account `admin`.

```
sudo adduser admin
```

3 Add user `admin` to group `sudo` and `console`.

```
sudo adduser admin sudo
```

Allow login into login console.

```
sudo adduser admin console
```

4 Perform the following steps securely using `sudo`. Use one of the methods below.

Non-GUI Environment Method Logout Method Power-off and Power-on Cycle Method

Substitute User (su) Command

The majority of users do not need to utilize the `su` command. [20].

In Kicksecure, by default:

- group `sudo` membership is required to use `su` (https://github.com/Kicksecure/security-misc/blob/master/usr/share/pam-configs/wheel-security-misc). [21]
- User `user` is a member of group `sudo`. (This might change in a later release.)

To permit the `su` command from user `user`, complete the following steps.

(Kicksecure-Qubes: perform these steps in Kicksecure Template.)

1 Enable the root account.

2 Add user `user` to group `root`.

```
sudo adduser user root
```

3 Re-enable SUID.

Set suid. Note: It is okay if the second command fails.

```
sudo permission-hardener disable /bin/su
sudo permission-hardener disable /usr/bin/su
```

[22]

4 Add SUID whitelisting.

```
sudo mkdir -p /etc/permission-hardener.d
```

Open file `/etc/permission-hardener.d/20_user.conf` in an editor with root rights.

Kicksecure Kicksecure for Qubes Others and Alternatives

See Open File with Root Rights for detailed instructions on why to use `sudoedit` for better security and how to use it.

```
sudoedit /etc/permission-hardener.d/20_user.conf
```

Add.

```
/bin/su exactwhitelist
/usr/bin/su exactwhitelist
```

5 Save.

6 Done.

Steps to permit `su` command from user `user` are complete.

Root Login

Root login within a virtual console will be disabled by default after upgrades. [23] [24]

To enable login from a virtual console, first apply the Enable Root Account instructions further above, then complete the steps below.

1 To allow root login, `/etc/securetty` must be configured. [25]

Open file `$(realpath /etc/securetty)` in an editor with root rights.

Kicksecure | Kicksecure for Qubes | Others and Alternatives

See Open File with Root Rights for detailed instructions on why to use `sudoedit` for better security and how to use it.

```
sudoedit $(realpath /etc/securetty)
```

2 Add the following content.

Note: Add one or more tty depending on your circumstances; see file `/etc/securetty.security-misc-orig` .

- Kicksecure:

```
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
tty9
tty10
```

- Kicksecure-Qubes:

```
hvc0
```

3 Save the file.

Recovery Mode

Root login is possible using recovery mode. [26]

When the root account is disabled, passwordless root login using recovery mode is possible; see below for the security impact.

Qubes Root Console

1 Open a dom0 terminal.

```
Qubes App Launcher (blue/grey "Q") → System Tools → Xfce Terminal
```

2 Run the following command.

Replace `vm-name` with the name of the actual VM where you want to open a root console.

```
qvm-run -u root vm-name xfce4-terminal
```

3 Done.

A Qubes root console will now be available.

Passwordless Recovery Mode Security Discussion

This is only relevant on the host and not inside virtual machines.

Passwordless recovery mode is allowed because a locked root password would break the rescue and emergency shell. Therefore the security-misc (https://github.com/Kicksecure/security-misc) package enables a passwordless rescue and emergency shell. This is the same solution that Debian will likely adapt for Debian installer. [27]

With passwordless root login, using recovery mode is allowed (through use of the security-misc package) on the host. To prevent adverse security effects posed by lesser adversaries with physical access to the machine, set up BIOS password protection, bootloader grub password protection and/or full disk encryption.

dsudo - default password sudo

`dsudo` is a Kicksecure specific feature. [28]

As long as still using the default password (not having changed sudo password), commands can be run as root without entering a password. This is useful for users having issues with changing the keyboard layout and for testing VMs.

Instead of using

```
sudo
```

use

```
dsudo
```

See Also

- System Recovery using SysRq Key
- Login spoofing
- Strong Linux User Account Isolation

Development

- Kicksecure code: Restrict access to the root account (https://github.com/Kicksecure/security-misc/pull/2

2) .

- https://forums.whonix.org/t/sysrq-magic-sysrq-key/8079/68
- https://forums.whonix.org/t/should-lesser-adversaries-with-physical-access-be-part-of-the-threat-model-of-whonix-whonix-host-kicksecure/7997

Footnotes

1. Also see: Permissions.
2. https://github.com/QubesOS/qubes-issues/issues/2695#issuecomment-301316132
3. For example flash utilities for Linux require root access. In theory, it's conceivable of software bugs in firmware or hardware resulting in hardware compromise without prior root compromise. No such examples happening in the wild were known to the author at time of writing.
4. An exploit or payload might require a function which is unavailable inside the sandbox.
5. Rationale for Change from Default Password changeme to Empty Default Password
6. In new builds of Kicksecure version `15.0.0.3.6`. Earlier Kicksecure builds did not lock the root account by default and should be locked.
7.
   - Applications supposed to be run as user but run as root might create root owned files. These file permissions error can lead to additional issues.
   - Inter process communications such as with dbus might be broken.
8.
   - https://help.ubuntu.com/community/RootSudo#Graphical_sudo
   - https://www.psychocats.net/ubuntu/graphicalsudo
9.
   - Reason primarily: not breaking the system, reliability. Non-reason: security.
   - https://askubuntu.com/questions/270006/why-should-users-never-use-normal-sudo-to-start-graphical-applications
   - In past there was `gksudo`, `kdesudo`. Nowadays with more and more applications using PolicyKit or polkit, these applications are no longer available as of Debian `buster`. `lxsudo` is an alternative.
10. `sudo` with `-H` / `--set-home` would also be OK. Syntax:
    ```
    sudo -H application-name
    ```
    Or.
    ```
    sudo --set-home application-name
    ```
    For example to start the partition manager `gparted` by default with root rights.
    ```
    sudo -H gparted
    ```
    Or.
    ```
    sudo --set-home gparted
    ```
11. https://unix.stackexchange.com/a/423287/535873

12. No longer expiring the root account since this broke adduser, see: https://forums.whonix.org/t/restrict-root-access/7658/59 (To prevent SSH login, see: Linux Locking An Account (https://www.cyberciti.biz/faq/linux-locking-an-account/) . This might prevent other login methods but this requires further investigation.)

```
sudo chage --expiredate 0 root
```

13. Why not login as root? This is for reasons of history, legacy. Even during the times of X11, `root` login was frowned upon. Nowadays, Wayland outright refuses to run graphical user interface (GUI) applications with root rights. Therefore for strong user isolation, logging in as `root` user is to be avoided. In an ideal world, the extra `admin` user would be unnecessary and people could simply use the `root` user. Or even better, all references to `root` would be removed and replaced with `admin`. However, this kind of education and convincing amongst many upstream projects for the purpose of Dev/Strong Linux User Account Isolation is totally unrealistic for organisatioral reasons, which are elaborated on the Linux User Experience versus Commercial Operating Systems page.
14. Any graphical application can see what is typed in another graphical application, for any user. Quote (https://blog.invisiblethings.org/2011/04/23/linux-security-circus-on-gui-isolation.html) Joanna Rutkowska, security researcher, founder and advisor (formerly architecture, security, and development) of Qubes OS:

    > One application can sniff or inject keystrokes to another one, can take snapshots of the screen occupied by windows belonging to another one, etc.

    If an application is compromised with an exploit due to a security vulnerability, it can be used as malware by the attacker. Once/if the application is not effectively confined by a mandatory access control (MAC) framework like AppArmor or firejail, it can compromise the user account where it is running and then proceed from there.

    See also sudo password sniffing for technical details.

15. Unless perhaps advanced users manage to run a different X server on a different virtual console. This might not be possible, secure. Depends on if the exclusive lock of X can be suspended while using an X server in a different virtual console. This has not been researched.
16. This step might be unnecessary. Not researched yet.
17. Pressing `Alt + Crtl + F7` results in `tty2`. This is to make these instructions compatible with most Linux distributions as well as Qubes.
    - Most Linux distributions login CLI virtual consoles on `tty1` (`Alt + Crtl + F1`) by default and X Window System on `tty7` (`Alt + Crtl + F7`).
    - Qubes X Window System by default runs on `tty1` .(`Alt + Crtl + F1`)

    `tty2` (`Alt + Crtl + F2`) will be for most users an unused virtual console which can be used for the purpose of this chapter.
18. An X Window System non-root user cannot sniff keystrokes of different (non-)root users utilizing a different virtual console (`tty`).
19. Non-simplified: applications run by user `user` in a different virtual console or run through systemd (user) services can be left running.

20. `su` is sometimes incorrectly referred to as the *superuser* command. It allows (http://www.linfo.org/su.html) :

    > … a change to a login session's owner (i.e., the user who originally created that session by logging on to the system) without the owner having to first log out of that session.
    >
    > Although su can be used to change the ownership of a session to any user, it is most commonly employed to change the ownership from an ordinary user to the root (i.e., administrative) user, thereby providing access to all parts of and all commands on the computer or system.

    By comparison, `sudo` makes it possible to execute system commands without the root password.

21. Implemented in package security-misc (https://github.com/Kicksecure/security-misc) .
22. ```
    sudo chmod 4755 /bin/su
    sudo chmod 4755 /usr/bin/su
    ```
23. security-misc `/etc/securetty (https://github.com/Kicksecure/security-misc/blob/master/etc/securetty.security-misc)` is empty by default.
24. When trying to login as root in a virtual console it will reply:

    > Login incorrect.

    Without previously asking for a password. This is not the worst case for usability and is better than asking for password and then failing.
25. `sudoedit` will not follow symlinks, therefore `realpath` is used.
26. https://forums.whonix.org/t/restrict-root-access/7658/46
27.
    - https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=802211
    - /etc/systemd/system/emergency.service.d/override.conf
    - /etc/systemd/system/rescue.service.d/override.conf
28. https://forums.whonix.org/t/dsudo-default-password-sudo/8766
    ```