Sleep Securely

Sacha

March 12, 2025

Contents

1 Synopsis

When a macOS system goes to sleep, it preserves the environment *exactly* as it was when the system sleep event was initiated. This is excellent for most people, as it is predictable and comfortable, convenient. Power users have other needs beyond this, often wanting alternative conveniences for their *less usual* habits, such as muting the system (we don't know what space we'll be in next), proactively turning off bluetooth and WiFi (reducing our security vulnerability), as well as other conveniences.

This journal is a *literate* journal describing the process in detail, with the intention that it is *tangled* into the target file.

2 Basic requirements

To simplify script handling and installation, a `justfile` will be created with basice *recipes*.

```
[private]
@help:
 just --list

# Add shell script executable status
[private]
chmod:
 @echo "==> Setting executable flag"
 chmod +x ./scripts/sleep_securely.sh
 @echo "--> Executable flag successfully set\n"

# Install script
install: chmod
 @echo "==> Installing script"
 cp ./scripts/sleep_securely.sh ~/bin/sleep_securely.sh
 @echo "--> Script successfully installed\n"
```

3 Actions to take upon sleep

Our target script file is a shell script, so we need to specify the script header first.

```
#!/bin/bash
```

Arguably the most valuable and confidential resource is the password library, hosted on the excellent KeePassXC application. As good as it is, this application still has to store the key somewhere in memory in exchange for giving us quick and convenient access to the password database. Let's ensure that this is the very first—and most important—action taken of all the security steps. Even if all aspects of this script fail, this one is the most valuable and impactful: kill KeePassXC, ejecting it from memory.

```
# Quit KeePassXC upon sleep
/usr/bin/killall KeePassXC
```

Prior to more destructive changes, let us also initiate a controlled quitting of Parallels Desktop. This application is used to run any number of servers for isolation ***and*** security. Its networking is able to interfere with the establishment of wifi connections, particularly some behind ***captive*** wifi setups, found in many coffee shops and locations advertising ***free*** wireless.

```
# Tell Parallels Desktop application to quit gently, if it is already open
if pgrep -q prl_client; then
    osascript -e 'tell application "Parallels Desktop" to quit'
fi
```

As I bring work to a close in a location I felt comfortable in, and ***trusted***, I cannot know with certainty where I will be waking the system up again. As the wireless capability comes to life, it starts out by scanning for known access points, giving away unintended information. In the interests of reducing exposure in unknown public settings, let's turn off the wireless network and force a positive decision to bring it up manually when desired.

```
# Disable Wi-Fi
networksetup -setairportpower en0 off
```

In the same vein, as well as for the pure convenience of not having the laptop try to ***snatch*** bluetooth connections to headphones and mice during its sleep, I want to bring down the bluetooth connection.

```
# Disable Bluetooth
/opt/homebrew/bin/blueutil -p 0
```

Unmount all external currently mounted volumes. This is needed to unmount any SSH file system mounted volumes relying on Tailscale (see below), which is why this step ***must*** run before Tailscale is taken offline. From a security perspective, it is also desirable to unmount any external volumes ***especially*** the encrypted ones, forcing me to manually remount them, reentering any encryption keys when needed once again.

```
# Unmount all external volumes
diskutil list external | grep -E '^\/' | while read -r volume; do
    diskutil unmount "$volume"
done
```

One known issue with privacy and use of secure VPNs, is that Tailscale, while incredibly useful, can result in alternative, unsecured, network routing. Before going to sleep, take Tailscale offline; it's easy to take it back online when needed again.

```
# Disable Tailscale
if [ -x "/Applications/Tailscale.app/Contents/MacOS/Tailscale" ]; then
    "/Applications/Tailscale.app/Contents/MacOS/Tailscale" down
fi
```

When closing the lid, literally and figuratively, on a project, there's no way of knowing where we'll be when we reawaken the computer. Just imagine the embarrassment of opening the lid and a movie or song resuming at full blast in a library, or worse, a quiet business meeting! Let's always mute the volume before going to sleep, it's safer.

```
# Mute system volume to prevent unpleasant surprises!
osascript -e "set volume with output muted"
```

3.1 Quit all sensitive applications

Given that it's impossible to know who will next *open*, or wake up, your computer, it's safest to proactively ***kill*** all communications applications with confidential information.

```
# Close all email applications (MUAs)
pkill -x "Proton Mail"
pkill -x "Thunderbird"
pkill -x "thunderbird"
pkill -x "Mail"

# Close sensitive communication applications
pkill -x "Discord"
pkill -x "FaceTime"
pkill -x "Messages"
pkill -x "Microsoft Teams"
pkill -x "Signal"
pkill -x "Skype"
pkill -x "Slack"
pkill -x "Telegram"
pkill -x "Trello"
pkill -x "Viber"
pkill -x "WhatsApp"

pkill -x "Zoom"
pkill -x "Zoom.us"
pkill -x "zoom"
pkill -x "zoom.us"

pkill -x "Authy Desktop"
```

3.2 Clear the clipboard

The clipboard is a wonderful thing, it's there to help us move blocks of information around, but it is liable to store many things we don't want to share: passwords copied from a password manager, confidential text from an email or private message, sensitive information personally identifying you. From a convenience standpoint, we rely on this functionality to paste *recently* copied text; once the system has been suspended and resumed, there is no longer a reasonable context around what was recent.

Objectively the best thing to do is to clear the clipboard to prevent any data leakage.

```
# Clear clipboard contents
/usr/bin/pbcopy < /dev/null
```

3.3 SSH security

SSH is an invaluable system administration tool for logging on to remote machines securely. As this is an important function in maintenance, development and other administration tasks, an agent is provided for convenience, `ssh-agent`, to help track and load users' identity keys.

Once keys are loaded, the agent provides user convenience by keeping private keys loaded, so that the passphrase doesn't need to be entered repeatedly. Removing keys from the agent is a good security practice as anyone walking up to the machine after waking from sleep cannot simply log in to a remote machine; once the key is deleted from the agent, `ssh-agent` won't be able to use it. This means it can't expose any secret information or establish unauthorized SSH sessions on other devices. It's straightforward to delete the key using `ssh-add -D`.

```
# Clear SSH agent identities
ssh-add -D
```

Rather than stop there, why even keep `ssh-agent` running? Let's kill it, ensuring that not only keys but also any other information it may be holding in memory is released and unusable.

```
# Securely wipe SSH agent
killall ssh-agent
```

3.4 Browsing privacy

macOS keeps a record of every file downloaded from the internet. These records include download dates, sources, and applications used. Clearing the responsible SQLite database removes this download history, evidence of downloaded files that might be sensitive, making it harder for malicious actors to see what has been downloaded, reducing leakage of internet activity.

The way it works is that when you download files, macOS adds a quarantine attribute, triggering "This file was downloaded from the internet" warnings. The database keeps metadata about these downloads

If you download a document from a secure source before sleep, this command ensures that when your system wakes up, there's no record of that download in the system's databases.

This action isn't destructive, only removing metadata about downloaded files, not the files themselves. The system will continue to function normally, and future downloads will simply start creating new entries in the database. This is an excellent privacy-enhancing measure that leaves no obvious trace of your download activity between sleep sessions, while having zero impact on system functionality.

Below is the command that will clear this database on close:

```
# Download trace elimination
sqlite3 ~/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV* "DELETE FROM LSQuarantineE
```

3.4.1 How does it work?

This is the path to the macOS quarantine database:

`~/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV*`

The database is located in the user's `Library/Preferences` folder, and the `*` is a wildcard matching any version number (e.g., QuarantineEventsV2, QuarantineEventsV3).

The following SQL command removes all records from the `LSQuarantineEvent` table:

```
DELETE FROM LSQuarantineEvent;
```