Updated: 2025-01-04
This is not a script. Just follow along and paste in the commands.
This file is for hardening Ubuntu desktop 24.04, not Ubuntu server.

The best results will be obtained by re-installing Ubuntu first and applying this
configuration prior to going online because we are not certain that the machine is
un-infected.

Disconnect from the internet

Make gdm (Gnome display manager) use Wayland windows server.
```
sudo nano /etc/gdm3/custom.conf
```

Find, remove the '#', and set `WaylandEnable=true`

Next, reboot.
Click on the user account.
The Gear icon then appears on the lower right.
Click and select 'Ubuntu' (not 'Ubuntu on Xorg')
And then give your password to login.
This will select Wayland, using Wayland is more secure

To verify:
```
echo $XDG_SESSION_TYPE
```

It should say: `wayland`

Wayland enforces strict isolation between applications. Clients
(applications) cannot directly access or manipulate the content of other
clients' windows or inputs. Each application only has access to its
allocated resources. X11 lacks this isolation. Applications can easily access and
modify the content of other applications' windows, inject input events (e.g., keystrokes),
and perform other potentially malicious actions.

Having done some preliminary testing of Ubuntu built-in apps, it is determined that X11 can
be removed. Ubuntu still keeps X11 for compatibility with legacy apps.
Optional, to remove X11 server, type:
```
sudo apt remove xserver-xorg-core x11-utils x11-apps
```

There will be some components left over, but wayland and gnome uses them, so they can't
be removed. We have done what we can to be secure.

If you have chosen to remove xserver then do this to prevent in from being installed:
```
cd/etc/apt/preferences.d/
```

`sudo nano` <package-name> for each of the above 3 packages
and place these lines inside
```
Package: <package-name>
Pin: release *
Pin-Priority: -1
```
Replacing `<package-name>` with the name of the package eg. `xserver-xorg-core`

Turn off WiFi peer to peer/direct mode, it allows direct connections to your laptop
bypassing the firewall. This does not affect WiFi operation.
Edit the following file (It may not exist yet)
```
sudo nano /etc/wpa_supplicant.conf
```

Put this line inside and save the file:
```
p2p_disabled=1
```

Verify that with this command:
```
nmcli
```

it should show wifi-p2p disconnected

Make sure the conf file is not overwritten on updates
```
sudo dpkg-divert --add /etc/wpa_supplicant.conf
```

If you have no need for WiFi altogether, then do these 2 lines:
```
sudo systemctl mask wpa_supplicant
sudo systemctl mask geoclue
```

To verify these 2, issue the commands:
```
sudo systemctl status wpa_supplicant
sudo systemctl status geoclue
```

they should both say it is masked and inactive

Make a second account for Daily Use
The second account does not have the capability to issue `sudo` commands
until you add the account to the admin group or make a rule via `visudo`.
This is good because any attack on apps run by this account cannot
elevate to gain root privilege. Go to Settings > System > Users to create the user

Turn off IPv6, we don't need it. Plus maybe your router/hardware firewall don't support it
then ipv6 traffic will pass right through bypassing your security
IPv6 was mainly created because we are running out of ip addresses. But with the arrival
of NAT, the ip addresses 192.168.x.x are now private, and we don't use up a publicly usable
ip address with every machine. The the urgency of IPv6 has been negated.

Go to Settings > Network, WiFi plus Ethernet, click on the Gear > ipv6 tab > disable

Set firewall to default deny incoming traffic, because we only go
outbound with firefox. Returning traffic will be allowed by default
because of firewall statefulness

```
sudo ufw default deny incoming
sudo ufw default deny outgoing
sudo ufw allow out 123/udp
sudo ufw allow out 53/udp
sudo ufw allow out 80/tcp
sudo ufw allow out 443/tcp
sudo ufw allow out 853/tcp
sudo ufw enable
```

You can check to confirm that all the rules are present
and that inbound and outbound default is set to deny

```
sudo ufw status numbered
sudo ufw status verbose
```

Using a different machine, you can use the nmap tool to scan your own PC,
Install nmap with:

```
sudo apt install nmap
```

and verify that there are no open ports by issuing :

```
nmap -O 192.168.0.0/24
```

In my personal configuration, I have the following firewall statefulness
settings in /etc/sysctl.conf

```
sudo nano /etc/sysctl.conf
```

Place these lines at the bottom:

```
net.netfilter.nf_conntrack_tcp_timeout_established=3
net.netfilter.nf_conntrack_udp_timeout=1
net.netfilter.nf_conntrack_udp_timeout_stream=1
net.netfilter.nf_conntrack_generic_timeout=1
net.netfilter.nf_conntrack_icmp_timeout=1
net.netfilter.nf_conntrack_tcp_timeout_close=2
net.netfilter.nf_conntrack_tcp_timeout_close_wait=5
net.netfilter.nf_conntrack_tcp_timeout_fin_wait=5
net.netfilter.nf_conntrack_tcp_timeout_time_wait=5
```

The firewall keeps the state open far too long for my liking.
And adversaries can take advantage of it. In my limited testing
they seem to pose no ill effect. I tested browsing to web sites,
logging in to my forums, gmail and google calendar; utilizing passwords
and yubikey security token, software & updates and software updater.
The original default values are documented here:
https://www.kernel.org/doc/html/latest/networking/nf_conntrack-sysctl.html
Use the command: `sudo sysctl -p` to have it take effect without rebooting.

Now with the firewall in place, we can connect to the internet

set tcsh as login shell instead of bash – less hackable,
```
sudo apt install tcsh
sudo sed 's:bash:tcsh:' < /etc/passwd > ~/Documents/passwd
sudo mv ~/Documents/passwd /etc/passwd
```

Reboot Now

Now issue the command:
```
echo $SHELL
```

It should say: `tcsh`
```
### No need to worry about existing scripts, because they all specify
### to use bash on the first line.
```

Now do an update + upgrade
```
sudo apt update && sudo apt upgrade
```

```
### Now we disrupt one way which infostealers work
sudo nano /usr/lib/sysctl.d/50-default.conf

### Put these lines inside:
kernel.shmmax=0
kernel.shmmin=0
kernel.msgmax=0
kernel.msgmni=0
kernel.shmmni=0
```

Reboot and use the command "`lsipc`" to verify.

Remove cups – historically hackable. Skip if you do print things, I don't
```
sudo apt remove cups-daemon
```

Tell apt never to install this cups-daemon again
```
sudo nano /etc/apt/preferences.d/cups-daemon
```

Put these lines inside. 'Pin-Priority -1' tells apt never to install this again:
```
Package: cups-daemon
Pin: release *
Pin-Priority: -1
```

Remove unnecessary networking daemons less network attack surface
wsdd enables web discovery and samba file/folder sharing with Windows
```
sudo apt remove wsdd
```

Tell apt never to install this wsdd again
```
sudo nano /etc/apt/preferences.d/wsdd
```

Put these lines inside. 'Pin-Priority -1' tells apt never to install this again:
```
Package: wsdd
```

```
Pin: release *
Pin-Priority: -1
```

Disable error reporting for Canonical - no need to report to Ubuntu/Canonical
Generally speaking, you want to eliminate network traffic because it leaves a
stateful firewall opening to a known ip address, which can be used in a
spoofing based attack
```
sudo dpkg -P whoopsie
sudo rm /usr/share/apport/whoopsie-upload-all
```

Mask the unneeded networking services we don't need so they don't run
Reduces your attack surface
```
sudo systemctl stop ModemManager.service
sudo systemctl mask ModemManager.service
sudo systemctl stop avahi-daemon.service
sudo systemctl mask avahi-daemon.service
```

Again you can verify the above with: `sudo systemctl status <serviceName>`

umask determines the permissions of any file or directory we create. This setting
allows owner and group rights only.
```
sudo nano /etc/login.defs
```

find or add: `UMASK 077`

Make sure the file is not overwritten when doing upgrades
```
sudo dpkg-divert --add /etc/login.defs
```

Reboot and then you can verify the above umask with
```
touch ~/Documents/test1
ls -la ~/Documents/test1
```

And it should say: `-rw-rw----`

Set home directory access to allow owners and respective group only
Setup home access defaults. It is not properly set because we only changed the
umask just now
```
chmod -R 770 /home/<yourAccount>/
sudo chmod -R 770 /home/<nextAccount>/
```

Edit this file and enable/un-comment (erasing the '#') in front of each setting except
the lines mentioning 'forwarding'
```
sudo nano /etc/sysctl.conf
```

Then we make sure it is not overwritten when doing upgrades
```
sudo dpkg-divert --add /etc/sysctl.conf
```

Edit this file
```
sudo nano /etc/systemd/resolved.conf
```

uncomment (remove the #) and change this to yes:
```
DNSOverTLS=
```

Then we make sure it is not overwritten when doing upgrades
```
sudo dpkg-divert --add /etc/systemd/resolved.conf
```

To verify, Reboot, then use the command:
```
resolvectl
```

It should show `+DNSoverTLS`

Remove execution rights for /tmp
```
sudo nano /usr/lib/systemd/system/tmp.mount
```

Find the line: `Options=mode=1777,strictatime,nosuid,nodev` …
And add `noexec,` after `nodev,`

Then we make sure it is not overwritten when doing upgrades
```
sudo dpkg-divert --add  /usr/lib/systemd/system/tmp.mount
```


Remove execution rights for /dev/hugepages
```
sudo nano /usr/lib/systemd/system/dev-hugepages.mount
```

Add `noexec` on line: `Options=nosuid,nodev`

Reboot, then use the `mount` command to verify that the noexec's are in place.

Now Reboot and connect to internet

Install things we need
```
sudo apt install firejail
sudo apt install apparmor-profiles
sudo apt install clamav
sudo apt install tcsh
sudo apt install openbox
```

Reboot

Firefox Snap has the 'home' snap ‘connection’ - it allows access to the entire home
directory
This violates my security directive to guard the Documents folder as it contains
Private and Confidential material in case of a breach. So we are uninstalling Firefox
Snap. Know that browsers are historically well known attack targets.
```
sudo snap remove firefox
```

Now we install the .deb version of firefox, which can be protected with Firejail,
which can blacklist /Documents folder access. First we add the mozillateam
repository.
```
sudo add-apt-repository ppa:mozillateam/ppa
```

Add the following lines to make deb firefox priority higher than the snap version
```
sudo nano /etc/apt/preferences.d/mozilla-firefox
```

And put the following lines inside it
```
Package: firefox*
Pin: release o=LP-PPA-mozillateam
Pin-Priority: 1001

Package: firefox*
Pin: release o=Ubuntu
Pin-Priority: -1
```

Now we install the deb version of firefox
```
sudo apt install firefox
```

Now we add ‘firejail’ in front of the firefox command so that firejail is used when
we click on the firefox icon.
Requires logout after change to take effect
```
sudo nano /usr/share/applications/firefox.desktop
```

Find all occurrences of “Exec=firefox”
and replace with “Exec=firejail /usr/lib/firefox/firefox -no-remote”

If firefox is updated, then you need to redo .desktop file or else firejail
won’t be invoked. So we make sure that updates don’t touch that file.
```
sudo dpkg-divert –add /usr/share/applications/firefox.desktop
```

Now we change the resolution of the Xephyr xserver
Go to Settings > Display and see what Resolution you are currently using
Then we tell firejail about it
```
sudo nano /etc/firejail/firejail.config
```

Find “xephyr-screen”
If you can see your resolution listed, then uncomment it (remove the #)
If you cannot see it, then start a new line and type it in following the way Firejail puts it.

Turn on Firejail tracelog so that blacklist violations are logged in syslog
```
sudo nano /etc/firejail/firejail.config
```

Search for "tracelog", remove the '#' , and set it to yes

Now we make sure that an firejail upgrade will not touch the file
```
sudo dpkg-divert –add /etc/firejail/firejail.config
```

Change firejail firefox profile to blacklist Documents folder access
plus add some more container security settings
```
sudo nano /etc/firejail/firefox-common.profile
```

add these lines
```
blacklist /home/<yourAccount>/Documents
blacklist  /home/<nextAccount>/Documents

blacklist /usr/lib/apg
blacklist /usr/lib/apt
blacklist /usr/lib/aspell
blacklist /usr/lib/binfmt.d
blacklist /usr/lib/brltty
blacklist /usr/lib/chkrootkit
blacklist /usr/lib/cloud-init
blacklist /usr/lib/cnf-update-db
### blacklist /usr/lib/command-not-found
blacklist /usr/lib/compat-ld
blacklist /usr/lib/console-setup
blacklist /usr/lib/cpp
blacklist /usr/lib/cups
blacklist /usr/lib/dbus-1.0
blacklist /usr/lib/debug
blacklist /usr/lib/dhcpcd
blacklist /usr/lib/dpjg
blacklist /usr/lib/dracut/
blacklist /usr/lib/emacsen-common
blacklist /usr/lib/evolution-data-server
blacklist /usr/lib/file/
blacklist /usr/lib/girepository-1.0
blacklist /usr/lib/gnome-session
blacklist /usr/lib/gnome-settings-daemon-3.0
blacklist /usr/lib/gnome-settings-daemon-47
blacklist /usr/gnome-shell
blacklist /usr/lib/firewalld
blacklist /usr/lib/firmware
blacklist /usr/lib/gnupg
blacklist /usr/lib/gnupg2
blacklist /usr/lib/gold-ld
blacklist /usr/lib/gvfs
blacklist /usr/lib/groff
blacklist /usr/lib/hdparm
blacklist /usr/lib/init
blacklist /usr/lib/initramfs-tools
blacklist /usr/lib/ispell
blacklist /usr/lib/kernel
blacklist /usr/lib/klibc-sw0VayLfV0hmLGCQE9vyfOnj81g.so
```

```
blacklist /usr/lib/klibc
blacklist /usr/lib/libpaps.so.0
blacklist /usr/lib/libpaps.so.0.0.0
blacklist /usr/lib/libreoffice
blacklist /usr/lib/linux
blacklist /usr/lib/linux-boot-probes
blacklist /usr/lib/linux-sound-base
blacklist /usr/lib/linux-tools
blacklist /usr/lib/llvm-18
blacklist /usr/lib/lb_solve
blacklist /usr/lib/linux-tools
blacklist /usr/lib/linux-tools-6.11.0-13
blacklist /usr/lib/linux-tools-6.11.0-9
blacklist /usr/lib/linux-tools-6.8.0-49
blacklist /usr/lib/linux-tools-6.1.1.0-9
blacklist /usr/lib/llvm-18
blacklist /usr/lib/locale
blacklist /usr/lib/lp_solve
blacklist /usr/lib/lsb
blacklist /usr/lib/man-db
blacklist /usr/lib/mime
blacklist /usr/lib/memtest86+
blacklist /usr/lib/modprobe.d
blacklist /usr/lib/modules
blacklist /usr/lib/modules-load.d
blacklist /usr/lib/networkd-dispatcher
blacklist /usr/lib/NetworkManager
blacklist /usr/lib/nvidia
blacklist /usr/lib/opensnitchd
blacklist /usr/lib/openssh
blacklist /usr/lib/os-probes
blacklist /usr/lib/os-release
blacklist /usr/lib/pam.d
blacklist /usr/lib/pcmcia-utils
blacklist /usr/lib/pcrlock.d
blacklist /usr/lib/pm-utils
blacklist /usr/lib/policykit-1
blacklist /usr/lib/polkit-1
blacklist /usr/lib/postfix
blacklist /usr/lib/pppd
blacklist /usr/lib/ppr
blacklist /usr/lib/python3
blacklist /usr/lib/python3.12
blacklist /usr/lib/rhythmbox
blacklist /usr/lib/rsyslog
blacklist /usr/lib/ruby
blacklist /usr/lib/snapd
blacklist /usr/lib/speech-dispatcher-modules
blacklist /usr/lib/ubiquity
blacklist /usr/lib/update-notifier
blacklist /usr/lib/valgrind
blacklist /usr/lib/rsyslog
blacklist /usr/lib/ruby
blacklist /usr/lib/sasl2
blacklist /usr/lib/sendmail
blacklist /usr/lib/shim
blacklist /usr/lib/snapd
blacklist /usr/lib/software-properties
```

```
blacklist /usr/lib/speech-dispatcher-modules
blacklist /usr/lib/sysctl.d
blacklist /usr/lib/systemd
blacklist /usr/lib/sysusers.d
blacklist /usr/lib/tmpfiles.d
blacklist /usr/lib/ubiquity
blacklist /usr/lib/ubuntu-advantage
blacklist /usr/lib/ubuntu-relase-upgrader
blacklist /usr/lib/udev
blacklist /usr/lib/udisk2
blacklist /usr/lib/ufw
blacklist /usr/lib/unity-settings-daemon
blacklist /usr/lib/updates-notifier
blacklist /usr/lib/valgrind
#blacklist /usr/lib/x86_64-linux-gnu
#blacklist /usr/lib/X11
blacklist /usr/lib/xorg
blacklist /usr/lib/xserver-xorg-video-intel

blacklist /usr/bin
blacklist /bin
blacklist /usr/sbin
blacklist /sbin
deterministic-shutdown
disable-mnt
nodbus
nonewprivs
nogroups
noroot
ignore mkfile
ignore mkdir
private-cache
private-dev
private-lib=x86_64-linux-gnu/xed,x86_64-linux-gnu/gdk-pixbuf-2.0,libenchant.so.1,librsvg-2.so.2
private-tmp
private-bin uname
private-cwd
private-etc hostname,localtime
seccomp
seccomp.block-secondary
tracelog
x11
```

To have your Yubikey work as a second factor authenticator online, find the two lines that
mention 'u2f' and insert a comment # symbol at the beginning of the line

Once again we make sure that an firejail upgrade will not touch the file
`sudo dpkg-divert –add /etc/firejail/firefox-common.profile`

You can verify the firefox protection by issuing the command:
firejail --profile=/etc/firejail/firefox.profile bash

And it should say: `Error: no suitable bash executable found`
That is because we have blacklisted the /bin and /usr/bin directories
Without any command shells, attackers cannot do much

Now we copy over the firefox apparmor profile that was installed and enable it
Even though the firejail configuration does not use this. This setting allows you
to run firefox without using Firejail and still get some protection
```
sudo cp /usr/share/apparmor/extra-profiles/firefox /etc/apparmor.d/
sudo apparmor_parser -r /etc/apparmor.d/firefox
```

One consequence of the x11 setting in firejail is that keyloggers should no longer
work. But the side effect is that you have to preset your browser window size via:
```
firejail --x11=xephyr openbox
```

You then right click on the xephyr desktop, choose Firefox using the menu, and
resize it.

You can use the same method to protect chromium or chrome.

Enable Ubuntu One LivePatch, it applies patches without need to reboot
Go to https://login.ubuntu.com and register yourself
Then:
```
sudo pro attach
```

Configure safe defaults for Firefox
You need to do these steps for each Ubuntu account because Firefox stores it's settings
separately for each

Go to Firefox > Settings
> General > Confirm before closing multiple tabs = checkmark
> Home > Homepage and new windows = Blank page
> Home > New Tabs = Blank Page
> Search > Search Suggestions > Show trending search suggestions = Uncheck
> Search > Address Bar > Shortcuts = Uncheck
> Privacy & Security > Strict radio button
> Privacy & Security > Cookies and Site Data > Delete cookies and site data when
Firefox is closed = Checkmark (This stops info-stealer malware from stealing
your cookies when you are not using Firefox)
> Privacy & Security > Passwords > Use a Primary Password = create this
You want to keep the least information so that a compromise will give less
of your info to an attacker, so no storing addresses and credit card numbers
> Privacy & Security > Autofill > Save and fill addresses = Uncheck
> Privacy & Security > Autofill > Save and fill payment methods = Uncheck
> Privacy & Security > Firefox Data Collection > Allow Firefox (3) = Uncheck
> Privacy & Security > Firefox Data Collection > Allow Firefox to install and run
studies = Uncheck
> Privacy & Security > HTTPS Only Mode > Enable HTTPS Only Mode in all
windows = selected
> Privacy & Security > Enable DNS over HTTPS > Max Protection = selected
Choose custom for the Max Protection Provider, and put in

https://dns.quad9.net/dns-query
Quad9 gives you malware protection
Go to Firefox > Add ons and themes > Find more Addons > Search for :
> PRIVACY BADGER and install it
> Also add an ad blocker of your choice to block annoying ads that block the screen

Browse to address `about:config`
Search for `tls`. Then change `security.tls.version.min` and set it to `4`.
This setting makes Firefox use the latest TLS v1.3, which has new privacy features.
However some sites have still not converted to TLS v1.3, so you will need to toggle this
value back to 3 sometimes.

Search for `network.negotiate-auth.allow-proxies` set it to `false`
Search for `security.ssl.require_safe_negotiation` set it to `true`
Search for `security.ssl.treat_unsafe_negotiation_as_broken` set it to `true`

We want scheduled anti-malware scans, do the following:
```
sudo crontab -e
```

Place the following line inside crontab to download fresh malware signatures at
21:45 and scan the whole drive at 22:00 every day
To have more than 1 scan per day, add more lines like it specifying a different time.
```
45 21 * * * /usr/bin/freshclam
0 22 * * * /usr/bin/clamscan -r /
```

So we have done the protections
Now we setup detections
Every time somebody uses sudo we want it logged to a file named sudo.log
```
sudo visudo
```

Add these lines to the bottom
```
Defaults logfile=/var/log/sudo.log
Defaults timestamp_timeout=1
```

You can view all past sudo commands issued with this:
```
sudo less /var/log/sudo.log
```

And now we install Logwatch:
```
sudo apt install logwatch
```

Here is how to use it :
```
sudo /usr/sbin/logwatch –detail high –range Today –filename <anyFilename>
less <YourProvidedFilename>
```

You can replace the word `Today` with `Yesterday` or `All`

Install ChkRootkit and rkhunter. As the package name says, they check for root kits,
which are used by hackers to hide themselves, run these periodically
```
sudo apt install chkrootkit
sudo apt install rkhunter
```

Install Wazuh SIEM (Security Information and Event Management)
It is a full featured open source security monitoring tool.
If installing on single machine, no need to install the agent
After install, use Firefox to browse to 127.0.0.1
Look at it at least once per day, so that you remember if an alert is caused by
you or if it warrants investigation.
Home > Overview > Threat Hunting > Events and investigate the alerts
https://documentation.wazuh.com/current/quickstart.html

For quick remediation, use Clonezilla disk imaging.
It is VERY IMPORTANT to have a backup! Because if all protections fail, and you
were unable to detect/remove the threat actor, you will have to restore from this backup.
You need 2 USB sticks. A small one to put the clonezilla onto. And a large one to store
the backup image or a portable HDD.
You also need to make a new image after every security improvement.
https://clonezilla.org/

Administrative Procedures to be followed religiously

```
### Do patching (Software Updater) Every Day

### Do file backups every day using Deja-vu/Gnome-backup into different
### folders of your backup HDD. Name the folders Mon to Sun.

### Disconnect from internet when connecting backup media

### Document Every Intrusion, every image recovery in a file afterwards.
### Lessons can be learned upon review.
```