THE LINUX FOUNDATION
TRAINING

A SysAdmin's Essential Guide to Linux Workstation Security

How to work from anywhere and keep your data, identity, and sanity

Contents

A SysAdmin's Essential Guide to Linux Workstation Security

How to work from anywhere and keep your data, identity, and sanity

By Konstantin Ryabitsev
Director of IT Infrastructure Security
at The Linux Foundation

This document is aimed at teams of systems administrators who use Linux workstations to access and manage your project's IT infrastructure.

If your systems administrators are remote workers, you may use this set of guidelines to help ensure that their workstations pass core security requirements in order to reduce the risk that they become attack vectors against the rest of your IT infrastructure.

Even if your systems administrators are not remote workers, chances are that they perform a lot of their work either from a portable laptop in a work environment, or set up their home systems to access the work infrastructure for after-hours/emergency support. In either case, you can adapt this set of recommendations to suit your environment.

Limitations

This, by no means, is an exhaustive "workstation hardening" document, but rather an attempt at a set of baseline recommendations to avoid most glaring security errors without introducing too much inconvenience. You may read this document and think it is way too paranoid, while someone else may think this barely scratches the surface. Security is just like driving on the highway — anyone going slower than you is an idiot, while anyone driving faster than you is a crazy person. These guidelines are merely a basic set of core safety rules that is neither exhaustive, nor a replacement for experience, vigilance, and common sense.

We're sharing this document as a way to bring the benefits of open-source collaboration to IT policy documentation. If you find it useful, we hope you'll contribute to its development by making a fork for your own organization and sharing your improvements.

Structure

Each section is split into two areas:

- The checklist that can be adapted to your project's needs
- Free-form list of considerations that explain what dictated these decisions

Checklist priority levels

The items in each checklist include the priority level, which we hope will help guide your decision:

(**ESSENTIAL**) items should definitely be high on the consideration list. If not implemented, they will introduce high risks to your workstation security.

(**NICE**) to have items will improve the overall security, but will affect how you interact with your work environment, and probably require learning new habits or unlearning old ones.

(**PARANOID**) is reserved for items we feel will significantly improve your workstation security, but will require a lot of adjustment to the way you interact with your operating system.

Remember, these are only guidelines. If you feel these priority levels do not reflect your project's commitment to security, you should adjust them as you see fit.

Choosing the right hardware

We do not mandate that our admins use a specific vendor or a specific model, so this section addresses core considerations when choosing a work system.

Checklist

- [ ] System supports SecureBoot (*ESSENTIAL*)
- [ ] System has no firewire, thunderbolt or ExpressCard ports (*NICE*)
- [ ] System has a TPM chip (*NICE*)

Considerations

SecureBoot

Despite its controversial nature, SecureBoot offers prevention against many attacks targeting workstations (Rootkits, "Evil Maid," etc), without introducing too much extra hassle. It will not stop a truly dedicated attacker, plus there is a pretty high degree of certainty that state security agencies have ways to defeat it (probably by design), but having SecureBoot is better than having nothing at all.

Alternatively, you may set up Anti Evil Maid which offers a more wholesome protection against the type of attacks that SecureBoot is supposed to prevent, but it will require more effort to set up and maintain.

Firewire, thunderbolt, and ExpressCard ports

Firewire is a standard that, by design, allows any connecting device full direct memory access to your system (see Wikipedia). Thunderbolt and ExpressCard are guilty of the same, though some later implementations of Thunderbolt attempt to limit the scope of memory access. It is best if the system you are getting has none of these ports, but it is not critical, as they usually can be turned off via UEFI or disabled in the kernel itself.

TPM Chip

Trusted Platform Module (TPM) is a crypto chip bundled with the motherboard separately from the core processor, which can be used for additional platform security (such as to store full-disk encryption keys), but is not normally used for day-to-day workstation operation. At best, this is a nice-to-have, unless you have a specific need to use TPM for your workstation security.

Pre-boot environment

This is a set of recommendations for your workstation before you even start with OS installation.

Checklist

- [ ] UEFI boot mode is used (not legacy BIOS) (*ESSENTIAL*)
- [ ] Password is required to enter UEFI configuration (*ESSENTIAL*)
- [ ] SecureBoot is enabled (*ESSENTIAL*)
- [ ] UEFI-level password is required to boot the system (*NICE*)

Considerations

UEFI and SecureBoot

UEFI, with all its warts, offers a lot of goodies that legacy BIOS doesn't, such as SecureBoot. Most modern systems come with UEFI mode on by default.

Make sure a strong password is required to enter UEFI configuration mode. Pay attention, as many manufacturers quietly limit the length of the password you are allowed to use, so you may need to choose high-entropy short passwords vs. long passphrases (see below for more on passphrases).

Depending on the Linux distribution you decide to use, you may or may not have to jump through additional hoops in order to import your distribution's SecureBoot key that would allow you to boot the distro. Many distributions have partnered with Microsoft to sign their released kernels with a key that is already recognized by most system manufacturers, therefore saving you the trouble of having to deal with key importing.

As an extra measure, before someone is allowed to even get to the boot partition and try some badness there, let's make them enter a password. This password should be different from your UEFI management password, in order to prevent shoulder-surfing. If you shut down and start a lot, you may choose to not bother with this, as you will already have to enter a LUKS passphrase and this will save you a few extra keystrokes.

Distro choice considerations

Chances are you'll stick with a fairly widely-used distribution such as Fedora, Ubuntu, Arch, Debian, or one of their close spin-offs. In any case, this is what you should consider when picking a distribution to use.

Checklist

- [ ] Has a robust MAC/RBAC implementation (SELinux/AppArmor/GrSecurity (*ESSENTIAL*)
- [ ] Publishes security bulletins (*ESSENTIAL*)
- [ ] Provides timely security patches (*ESSENTIAL*)
- [ ] Provides cryptographic verification of packages (*ESSENTIAL*)
- [ ] Fully supports UEFI and SecureBoot (*ESSENTIAL*)
- [ ] Has robust native full disk encryption support (*ESSENTIAL*)

Considerations

SELinux, AppArmor, and GrSecurity/PaX

Mandatory Access Controls (MAC) or Role-Based Access Controls (RBAC) are an extension of the basic user/group security mechanism used in legacy POSIX systems. Most distributions these days either already come bundled with a MAC/RBAC implementation (Fedora, Ubuntu), or provide a mechanism to add it via an optional post-installation step (Gentoo, Arch, Debian). Obviously, it is highly advised that you pick a distribution that comes pre-configured with a MAC/RBAC system, but if you have strong feelings about a distribution that doesn't have one enabled by default, do plan to configure it post-installation.

Distributions that do not provide any MAC/RBAC mechanisms should be strongly avoided, as traditional POSIX user- and group-based security should be considered insufficient in this day and age. If you would like to start out with a MAC/RBAC workstation, AppArmor and GrSecurity/PaX are generally considered easier to learn than SELinux. Furthermore, on a workstation, where there are few or no externally listening daemons, and where user-run applications pose the highest risk, GrSecurity/PaX will offer more security benefits than just SELinux.

Distro security bulletins

Most of the widely used distributions have a mechanism to deliver security bulletins to their users, but if you are fond of something esoteric, check whether the developers have a documented mechanism of alerting the users about security vulnerabilities and patches. Absence of such mechanism is a major warning sign that the distribution is not mature enough to be considered for a primary admin workstation.

Timely and trusted security updates

Most of the widely used distributions deliver regular security updates, but is worth checking to ensure that critical package updates are provided in a timely fashion. Avoid using spin-offs and "community rebuilds" for this reason, as they routinely delay security updates due to having to wait for the upstream distribution to release it first.

These days, it is hard to find a distribution that does not use cryptographic signatures on packages, updates metadata, or both. That being said, fairly widely used distributions have been known to go for years before introducing this basic security measure (Arch, I'm looking at you), so this is something worth checking.

Distros supporting UEFI and SecureBoot

Check that the distribution supports UEFI and SecureBoot. Find out whether it requires importing an extra key or whether it signs its boot kernels with a key already trusted by systems manufacturers (e.g. via an agreement with Microsoft). Some distributions do not support UEFI/SecureBoot but offer alternatives to ensure tamper-proof or tamper-evident boot environments (Qubes-OS uses Anti Evil Maid, mentioned earlier). If a distribution doesn't support SecureBoot and has no mechanisms to prevent boot-level attacks, look elsewhere.

Full disk encryption

Full disk encryption is a requirement for securing data at rest, and is supported by most distributions. As an alternative, systems with self-encrypting hard drives may be used (normally implemented via the on-board TPM chip) and offer comparable levels of security plus faster operation, but at a considerably higher cost.

Distro installation guidelines

All distributions are different, but here are general guidelines:

Checklist

- [ ] Use full disk encryption (LUKS) with a robust passphrase (*ESSENTIAL*)
- [ ] Make sure swap is also encrypted (*ESSENTIAL*)
- [ ] Require a password to edit bootloader (can be same as LUKS) (*ESSENTIAL*)
- [ ] Set up a robust root password (can be same as LUKS) (*ESSENTIAL*)
- [ ] Use an unprivileged account, part of administrators group (*ESSENTIAL*)
- [ ] Set up a robust user-account password, different from root (*ESSENTIAL*)

Considerations

Full disk encryption

Unless you are using self-encrypting hard drives, it is important to configure your installer to fully encrypt all the disks that will be used for storing your data and your system files. It is not sufficient to simply encrypt the user directory via auto-mounting cryptfs loop files (I'm looking at you, older versions of Ubuntu), as this offers no protection for system binaries or swap, which is likely to contain a slew of sensitive data. The recommended encryption strategy is to encrypt the LVM device, so only one passphrase is required during the boot process.

The `/boot` partition will usually remain unencrypted, as the bootloader needs to be able to boot the kernel itself before invoking LUKS/dm-crypt. Some distributions support encrypting the /boot partition as well (e.g. Arch), and it is possible to do the same on other distros, but likely at the

cost of complicating system updates. It is not critical to encrypt `/boot` if your distro of choice does not natively support it, as the kernel image itself leaks no private data and will be protected against tampering with a cryptographic signature checked by SecureBoot.

Choosing good passphrases

Modern Linux systems have no limitation of password/passphrase length, so the only real limitation is your level of paranoia and your stubbornness. If you boot your system a lot, you will probably have to type at least two different passwords: one to unlock LUKS, and another one to log in, so having long passphrases will probably get old really fast. Pick passphrases that are 2-3 words long, easy to type, and preferably from rich/mixed vocabularies.

Examples of good passphrases (yes, you can use spaces):

- nature abhors roombas
- 12 in-flight Jebediahs
- perdon, tengo flatulence

Weak passphrases are combinations of words you're likely to see in published works or anywhere else in real life, and you should avoid using them, as attackers are starting to include such simple passphrases into their brute-force strategies.

Examples of passphrases to avoid:

- Mary had a little lamb
- you're a wizard, Harry
- to infinity and beyond

You can also stick with non-vocabulary passwords that are at least 10-12 characters long, if you prefer that to typing passphrases.

Unless you have concerns about physical security, it is fine to write down your passphrases and keep them in a safe place away from your work desk.

Root, user passwords and the admin group

We recommend that you use the same passphrase for your root password as you use for your LUKS encryption (unless you share your laptop with other trusted people who should be able to unlock the drives, but shouldn't be able to become root). If you are the sole user of the laptop, then having your root password be different from your LUKS password has no meaningful security advantages. Generally, you can use the same passphrase for your UEFI administration, disk encryption, and root account — knowing any of these will give an attacker full control of your system anyway, so there is little security benefit to have them be different on a single-user workstation.

You should have a different, but equally strong password for your regular user account that you will be using for day-to-day tasks. This user should be member of the admin group (e.g. `wheel` or similar, depending on the distribution), allowing you to perform `sudo` to elevate privileges.

In other words, if you are the sole user on your workstation, you should have 2 distinct, robust, equally strong passphrases you will need to remember:

Admin-level, used in the following locations:

- UEFI administration
- Bootloader (GRUB)
- Disk encryption (LUKS)
- Workstation admin (root user)

User-level, used for the following:

- User account and sudo
- Master password for the password manager

All of them, obviously, can be different if there is a compelling reason.

Post-installation hardening

Post-installation security hardening will depend greatly on your distribution of choice, so it is futile to provide detailed instructions in a general document such as this one. However, here are some steps you should take:

Checklist

- [ ] Globally disable firewire and thunderbolt modules (*ESSENTIAL*)
- [ ] Check your firewalls to ensure all incoming ports are filtered (*ESSENTIAL*)
- [ ] Make sure root mail is forwarded to an account you check (*ESSENTIAL*)
- [ ] Set up an automatic OS update schedule, or update reminders (*ESSENTIAL*)
- [ ] Check to ensure sshd service is disabled by default (*NICE*)
- [ ] Configure the screensaver to auto-lock after a period of inactivity (*NICE*)
- [ ] Set up logwatch (*NICE*)
- [ ] Install and use rkhunter (*NICE*)
- [ ] Install an Intrusion Detection System (*NICE*)

Considerations

Blacklisting modules

To blacklist a firewire and thunderbolt modules, add the following lines to a file in `/etc/modprobe.d/blacklist-dma.conf`:

```
blacklist firewire-core
blacklist thunderbolt
```

The modules will be blacklisted upon reboot. It doesn't hurt doing this even if you don't have these ports (but it doesn't do anything either).

Root mail

By default, root mail is just saved on the system and tends to never be read. Make sure you set your `/etc/aliases` to forward root mail to a mailbox that you actually read, otherwise you may miss important system notifications and reports:

```
# Person who should get root's mail
root:          bob@example.com
```

Run `newaliases` after this edit and test it out to make sure that it actually gets delivered, as some email providers will reject email coming in from nonexistent or non-routable domain names. If that is the case, you will need to play with your mail forwarding configuration until this actually works.

Firewalls, sshd, and listening daemons

The default firewall settings will depend on your distribution, but many of them will allow incoming `sshd` ports. Unless you have a compelling legitimate reason to allow incoming ssh, you should filter that out and disable the `sshd` daemon.

```
systemctl disable sshd.service
systemctl stop sshd.service
```

You can always start it temporarily if you need to use it.

In general, your system shouldn't have any listening ports apart from responding to ping. This will help safeguard you against network-level 0-day exploits.

Automatic updates or notifications

It is recommended to turn on automatic updates, unless you have a very good reason not to do so, such as fear that an automatic update would render your system unusable (it's happened in the past, so this fear is not unfounded). At the very least, you should enable automatic notifications of available updates. Most distributions already have this service automatically running for you, so chances are you don't have to do anything. Consult your distribution documentation to find out more.

You should apply all outstanding errata as soon as possible, even if something isn't specifically labeled as "security update" or has an associated CVE code. All bugs have the potential of being security bugs and erring on the side of newer, unknown bugs is generally a safer strategy than sticking with old, known ones.

Watching logs

You should have a keen interest in what happens on your system. For this reason, you should install logwatch and configure it to send nightly activity reports of everything that happens on your system. This won't prevent a dedicated attacker, but is a good safety-net feature to have in place.

Note, that many systemd distros will no longer automatically install a syslog server that logwatch needs (due to systemd relying on its own journal), so you will need to install and enable rsyslog to make sure your `/var/log` is not empty before `logwatch` will be of any use.

Rkhunter and IDS

Installing rkhunter and an intrusion detection system (IDS) like `aide` or `tripwire` will not be that useful unless you actually understand how they work and take the necessary steps to set them up properly (such as, keeping the databases on external media, running checks from a trusted environment, remembering to refresh the hash databases after performing system updates and configuration changes, etc). If you are not willing to take these steps and adjust how you do things on your own workstation, these tools will introduce hassle without any tangible security benefit.

We do recommend that you install `rkhunter` and run it nightly. It's fairly easy to learn and use, and though it will not deter a sophisticated attacker, it may help you catch your own mistakes.

Personal workstation backups

Workstation backups tend to be overlooked or done in a haphazard, often unsafe manner.

Checklist

- [ ] Set up encrypted workstation backups to external storage (*ESSENTIAL*)
- [ ] Use zero-knowledge backup tools for off-site/cloud backups (*NICE*)

Considerations

Full encrypted backups to external storage

It is handy to have an external hard drive where one can dump full backups without having to worry about such things like bandwidth and upstream speeds (in this day and age most providers still offer dramatically asymmetric upload/download speeds). Needless to say, this hard drive needs to be in itself encrypted (again, via LUKS), or you should use a backup tool that creates encrypted backups, such as `duplicity` or its GUI companion, `deja-dup`. I recommend using the latter with a good randomly generated passphrase, stored in a safe offline place. If you travel with your laptop, leave this drive at home to have something to come back to in case your laptop is lost or stolen.

In addition to your home directory, you should also back up `/etc` and `/var/log` for various forensic purposes.

Above all, avoid copying your home directory onto any unencrypted storage, even as a quick way to move your files around between systems, as you will most certainly forget to erase it once you're done, exposing potentially private or otherwise security sensitive data to snooping hands — especially if you keep that storage media in the same bag with your laptop or in your office desk drawer.

Selective zero-knowledge backups off-site

Off-site backups are also extremely important and can be done either to your employer, if they offer space for it, or to a cloud provider. You can set up a separate duplicity/deja-dup profile to only include most important files in order to avoid transferring huge amounts of data that you don't really care to back up off-site (internet cache, music, downloads, etc).

Alternatively, you can use a zero-knowledge backup tool, such as SpiderOak, which offers an excellent Linux GUI tool and has additional useful features such as synchronizing content between multiple systems and platforms.

Best practices

What follows is a curated list of best practices that we think you should adopt. It is most certainly non-exhaustive, but rather attempts to offer practical advice that strikes a workable balance between security and overall usability.

Graphical environment

The venerable X protocol was conceived and implemented for a wholly different era of personal computing and lacks important security features that should be considered essential on a networked workstation. To give a few examples:

- Any X application has access to full screen contents
- Any X application can register to receive all keystrokes, regardless into which window they are typed

A sufficiently severe browser vulnerability means attackers get automatic access to what is effectively a builtin keylogger and screen recorder and can watch and capture everything you type into your root terminal sessions.

You should strongly consider switching to a more modern platform like Wayland, even if this means using many of your existing applications through an X11 protocol wrapper. With Fedora starting to default to Wayland for all applications, we can hope that most software will soon stop requiring the legacy X11 layer.

Browsers

There is no question that the web browser will be the piece of software with the largest and the most exposed attack surface on your system. It is a tool written specifically to download and execute untrusted, frequently hostile code. It attempts to shield you from this danger by employing multiple mechanisms such as sandboxes and code sanitization, but they have all been previously defeated on multiple occasions. You should learn to approach browsing websites as the most insecure activity you'll engage in on any given day.

There are several ways you can reduce the impact of a compromised browser, but the truly effective ways will require significant changes in the way you operate your workstation.

1: Use two different browsers (*ESSENTIAL*)

This is the easiest to do, but only offers minor security benefits. Not all browser compromises give an attacker full unfettered access to your system — sometimes they are limited to allowing one to read local browser storage, steal active sessions from other tabs, capture input entered into the browser, etc. Using two different browsers, one for work/high security sites, and another for everything else will help prevent minor compromises from giving attackers access to the whole cookie jar. The main inconvenience will be the amount of memory consumed by two different browser processes.

Here's what we recommend:

Firefox for work and high security sites

Use Firefox to access work-related sites, where extra care should be taken to ensure that data like cookies, sessions, login information, keystrokes, etc, should most definitely not fall into attackers' hands. You should NOT

use this browser for accessing any other sites except select few.

You should install the following Firefox add-ons:

- [ ] NoScript (*ESSENTIAL*)
  - NoScript prevents active content from loading, except from user whitelisted domains. It is a great hassle to use with your default browser (though offers really good security benefits), so we recommend only enabling it on the browser you use to access work-related sites.

- [ ] Privacy Badger (*ESSENTIAL*)
  - EFF's Privacy Badger will prevent most external trackers and ad platforms from being loaded, which will help avoid compromises on these tracking sites from affecting your browser (trackers and ad sites are very commonly targeted by attackers, as they allow rapid infection of thousands of systems worldwide).

- [ ] HTTPS Everywhere (*ESSENTIAL*)
  - This EFF-developed Add-on will ensure that most of your sites are accessed over a secure connection, even if a link you click is using http:// (great to avoid a number of attacks, such as SSL-strip).

- [ ] Certificate Patrol (*NICE*)
  - This tool will alert you if the site you're accessing has recently changed their TLS certificates — especially if it wasn't nearing expiration dates or if it is now using a different certification authority. It helps alert you if someone is trying to man-in-the-middle your connection, but generates a lot of benign false-positives.

You should leave Firefox as your default browser for opening links, as NoScript will prevent most active content from loading or executing.

Chrome/Chromium for everything else

Chromium developers are ahead of Firefox in adding a lot of nice security features (at least on Linux), such as seccomp sandboxes, kernel user namespaces, etc, which act as an added layer of isolation between the sites you visit and the rest of your system. Chromium is the upstream open-source project, and Chrome is Google's proprietary binary build based on it (insert the usual paranoid caution about not using it for anything you don't want Google to know about).

It is recommended that you install **Privacy Badger** and **HTTPS Everywhere** extensions in Chrome as well and give it a distinct theme from Firefox to indicate that this is your "untrusted sites" browser.

2: Use firejail (*ESSENTIAL*)

Firejail is a project that uses Linux namespaces and seccomp-bpf to create a sandbox around Linux applications. It is an excellent way to help build additional protection between the browser and the rest of your system. You can use Firejail to create separate isolated instances of Firefox to use for different purposes — for work, for personal but trusted sites (such as banking), and one more for casual browsing (social media, etc).

Firejail is most effective on Wayland, unless you use X11-isolation mechanisms (the `–x11 flag`). To start using Firejail with Firefox, please refer to the documentation provided by the project:

- Firefox Sandboxing Guide

3: Fully separate your work and play environments via virtualization (*PARANOID*)

See QubesOS project, which strives to provide a "reasonably secure" workstation environment via compartmentalizing your applications into separate fully isolated VMs. You may also investigate SubgraphOS that achieves similar goals using container technology (currently in Alpha).

Use Fido U2F for website 2-factor authentication

Fido U2F is a standard developed specifically to provide a mechanism for 2-factor authentication and combat credential phishing. Regular OTP (one-time password) mechanisms are ineffective in the case where the attacker is able to trick you into submitting your password and token into a malicious site masquerading as a legitimate service. The U2F protocol will store site authentication data on the USB token that will prevent you from accidentally giving an attacker both your password and your one-time token if you try to use it on anything other than the legitimate website.

See this site for a curated list of services providing Fido U2F support:

- dongleauth.info

Note, that not all browsers currently support U2F-capable hardware tokens, and if you use sandboxes or virtualization-based isolation around your browser, you may have to work extra hard to enable USB pass-through from the application to your USB token.

Password managers

Checklist

- [ ] Use a password manager (*ESSENTIAL*)
- [ ] Use unique, randomly generated passwords on unrelated sites (*ESSENTIAL*)
- [ ] Use a password manager that supports team sharing (*NICE*)
- [ ] Use a separate password manager for non-website accounts (*NICE*)

Considerations

Using strong, unique, randomly generated passwords should be a critical requirement for every member of your team. Credential theft is happening all the time — either via compromised computers, stolen database dumps, remote site exploits, or any number of other means. No credentials should be reused across different sites, ever.

In-browser password manager

Every browser has a mechanism for saving passwords that is fairly secure and can sync with vendor-maintained cloud storage while keeping the data encrypted with a user-provided passphrase. However, this mechanism has important disadvantages:

1. It does not work across browsers
2. It does not offer any way of sharing credentials with team members

There are several well-supported, free-or-cheap password managers that are well-integrated into multiple browsers, work across platforms, and offer group sharing (usually as a paid service). Solutions can be easily found via search engines.

Standalone password manager

One of the major drawbacks of any password manager that comes integrated with the browser is the fact that it's part of the application that is most likely to be attacked by intruders. If this makes you uncomfortable (and it should), you may choose to have two different password managers — one for websites that is integrated into your browser, and one that runs as a standalone application. The latter can be used to store high-risk credentials such as root passwords, database passwords, other shell account credentials, etc.

It may be particularly useful to have such tool for sharing superuser account credentials with other members of your team (server root passwords, ILO passwords, database admin passwords, bootloader passwords, etc).

A few tools can help you:

- KeePassX, which improves team sharing in version 2
- Pass, which uses text files and PGP and integrates with git
- Django-Pstore, which uses GPG to share credentials between admins
- Hiera-Eyaml, which, if you are already using Puppet for your infrastructure, may be a handy way to track your server/service credentials as part of your encrypted Hiera data store

Securing SSH and PGP private keys

Personal encryption keys, including SSH and PGP private keys, are going to be the most prized items on your workstation — something the attackers will be most interested in obtaining, as that would allow them to further attack your infrastructure or impersonate you to other admins. You should take extra steps to ensure that your private keys are well protected against theft.

Checklist

- [ ] Strong passphrases are used to protect private keys (*ESSENTIAL*)
- [ ] PGP Master key is stored on removable storage (*NICE*)
- [ ] Auth, Sign and Encrypt Subkeys are stored on a smartcard device (*NICE*)
- [ ] SSH is configured to use PGP Auth key as ssh private key (*NICE*)

Considerations

The best way to prevent private key theft is to use a smartcard to store your encryption private keys and never copy them onto the workstation. There are several manufacturers that offer OpenPGP capable devices:

- Kernel Concepts, where you can purchase both the OpenPGP compatible smartcards and the USB readers, should you need one.
- Yubikey, which offers OpenPGP smartcard functionality in addition to many other cool features (U2F, PIV, HOTP, etc).
- NitroKey, which is based on open-source software and hardware

It is also important to make sure that the master PGP key is not stored on the main workstation, and only subkeys are used. The master key

will only be needed when signing someone else's keys or creating new subkeys — operations which do not happen very frequently. You may follow the Debian's subkeys guide to learn how to move your master key to removable storage and how to create subkeys.

You should then configure your gnupg agent to act as ssh agent and use the smartcard-based PGP Auth key to act as your ssh private key. We publish a detailed guide on how to do that using either a smartcard reader or a Yubikey NEO.

If you are not willing to go that far, at least make sure you have a strong passphrase on both your PGP private key and your SSH private key, which will make it harder for attackers to steal and use them.

Hibernate or shut down, do not suspend

When a system is suspended, the RAM contents are kept on the memory chips and can be read by an attacker (known as the Cold Boot Attack). If you are going away from your system for an extended period of time, such as at the end of the day, it is best to shut it down or hibernate it instead of suspending it or leaving it on.

SELinux on the workstation

Checklist

- [ ] Make sure SELinux is enforcing on your workstation (*ESSENTIAL*)

Considerations

SELinux is a Mandatory Access Controls (MAC) extension to core POSIX permissions functionality. It is mature, robust, and has come a long way since its initial roll-out. Regardless, many sysadmins to this day repeat the outdated mantra of "just turn it off."

That being said, SELinux will have limited security benefits on the workstation, as most applications you will be running as a user are going to be running unconfined. It does provide enough net benefit to warrant leaving it on, as it will likely help prevent an attacker from escalating privileges to gain root-level access via a vulnerable daemon service.

Our recommendation is to leave it on and enforcing.

Further reading

The world of IT security is a rabbit hole with no bottom. If you would like to go deeper, or find out more about security features on your particular distribution, please check out the following links:

- Fedora Security Guide
- CESG Ubuntu Security Guide
- Debian Security Manual
- Arch Linux Security Wiki
- Mac OSX Security

License

Choosing the right hardware

- [ ] System supports SecureBoot (*ESSENTIAL*)
- [ ] System has no firewire, thunderbolt or ExpressCard ports (*NICE*)
- [ ] System has a TPM chip (*NICE*)

Pre-boot environment

- [ ] UEFI boot mode is used (not legacy BIOS) (*ESSENTIAL*)
- [ ] Password is required to enter UEFI configuration (*ESSENTIAL*)
- [ ] SecureBoot is enabled (*ESSENTIAL*)
- [ ] UEFI-level password is required to boot the system (*NICE*)

Distro choice considerations

- [ ] Has a robust MAC/RBAC implementation (SELinux/AppArmor/GrSecurity) (*ESSENTIAL*)
- [ ] Publishes security bulletins (*ESSENTIAL*)
- [ ] Provides timely security patches (*ESSENTIAL*)
- [ ] Provides cryptographic verification of packages (*ESSENTIAL*)
- [ ] Fully supports UEFI and SecureBoot (*ESSENTIAL*)
- [ ] Has robust native full disk encryption support (*ESSENTIAL*)

Distro installation guidelines

- [ ] Use full disk encryption (LUKS) with a robust passphrase (*ESSENTIAL*)
- [ ] Make sure swap is also encrypted (*ESSENTIAL*)
- [ ] Require a password to edit bootloader (can be same as LUKS) (*ESSENTIAL*)
- [ ] Set up a robust root password (can be same as LUKS) (*ESSENTIAL*)
- [ ] Use an unprivileged account, part of administrators group (*ESSENTIAL*)
- [ ] Set up a robust user-account password, different from root (*ESSENTIAL*)

Password managers

- [ ] Use a password manager (*ESSENTIAL*)
- [ ] Use unique passwords on unrelated sites (*ESSENTIAL*)
- [ ] Use a password manager that supports team sharing (*NICE*)
- [ ] Use a separate password manager for non-website accounts (*NICE*)

Post-installation hardening

- [ ] Globally disable firewire and thunderbolt modules (*ESSENTIAL*)
- [ ] Check your firewalls to ensure all incoming ports are filtered (*ESSENTIAL*)
- [ ] Make sure root mail is forwarded to an account you check (*ESSENTIAL*)
- [ ] Set up an automatic OS update schedule, or update reminders (*ESSENTIAL*)
- [ ] Check to ensure sshd service is disabled by default (*NICE*)
- [ ] Configure the screensaver to auto-lock after a period of inactivity (*NICE*)
- [ ] Set up logwatch (*NICE*)
- [ ] Install and use rkhunter (*NICE*)
- [ ] Install an Intrusion Detection System (*NICE*)

Browsing

- [ ] Use two different browsers (*ESSENTIAL*)
- [ ] Use Firefox for work and high security sites. Install the following Firefox add-ons:
  - [ ] NoScript (*ESSENTIAL*)
  - [ ] Privacy Badger (*ESSENTIAL*)
  - [ ] HTTPS Everywhere (*ESSENTIAL*)
  - [ ] Certificate Patrol (*NICE*)
- [ ] Use Chrome/Chromium for everything else
- [ ] Use two different browsers, one inside a dedicated VM (*NICE*)
- [ ] Fully separate your work and play environments via virtualization (*PARANOID*)

Securing SSH and PGP private keys

- [ ] Strong passphrases are used to protect private keys (*ESSENTIAL*)
- [ ] PGP Master key is stored on removable storage (*NICE*)
- [ ] Auth, Sign and Encrypt Subkeys are stored on a smartcard device (*NICE*)
- [ ] SSH is configured to use PGP Auth key as ssh private key (*NICE*)

SELinux on the workstation

- [ ] Make sure SELinux is enforcing on your workstation (*ESSENTIAL*)

Personal workstation backups

- [ ] Set up encrypted workstation backups to external storage (*ESSENTIAL*)
- [ ] Use zero-knowledge backup tools for off-site/cloud backups (*NICE*)

Checklist priority levels

The items in each checklist include the priority level, which we hope will help guide your decision.

(**ESSENTIAL**) items should definitely be high on the consideration list. If not implemented, they will introduce high risks to your workstation security.

(**NICE**) to have items will improve the overall security, but will affect how you interact with your work environment, and probably require learning new habits or unlearning old ones.

(**PARANOID**) is reserved for items we feel will significantly improve your workstation security, but will require a lot of adjustment to the way you interact with your operating system.

Advance Your Career With Training From The Linux Foundation

The Linux Foundation is the go-to place for training and certification on some of the hottest and most important software technologies. As the home to 50+ leading open source projects — including Linux, Node.js, Cloud Foundry, Kubernetes, and many others — The Linux Foundation offers:

- Detailed, hands-on training from true industry experts that you just can't find anywhere else.
- Certifications created by collaborating with some of the top companies and subject-matter experts in the world. And they're available online anytime, anywhere, saving you a trip to the testing center.
- Unparallelled expertise and experience in teaching people to use and profit from open source technology.

So take your expertise to the next level with training straight from the source. For more information on The Linux Foundation's training and certification programs, please visit: http://training.linuxfoundation.org.

THE LINUX FOUNDATION
TRAINING

THE LINUX FOUNDATION

The Linux Foundation promotes, protects and standardizes Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about our Linux Training program, please visit us at training.linuxfoundation.org.