Developer Workstation Hygiene

Your code is your reputation. Protect it.

THE LINUX FOUNDATION

Who am I?

- Director of IT Infrastructure Security at The Linux Foundation
- Kernel.org systems administrator
- Long-time Linux enthusiast
- Russian-Canadian (more Canadian by this point)
- https://paranoidbeavers.ca/
- @mricon

Topics covered

1. PGP code signing best practices
2. Securing your private keys
3. Securing your browsing habits

PGP is king

In the open-source development world, PGP is the de-facto identity verification mechanism.

Everyone who participates in open-source development should create and maintain a PGP key.

On Linux, this means GnuPG.

PGP and git

1. Sign your tags ("**`git tag -s`**")
   a. Git pull will automatically verify it and show verification results during merge

2. Sign your commits ("**`git commit -S`**")
   a. Then "**`git merge --verify-signatures -S`**"
   b. Warning: all commits MUST be signed, so don't do this unless your project has very strict commit signature hygiene already

WoT vs. TOFU

- So few people understand how the Web of Trust works, GnuPG now implements “Trust on First Use” (TOFU).
  - New in GnuPG-2.2
  - Think of it as “SSH trust model”
  - You can combine this with WoT (tofu+pgp)

- TOFU is still pretty new and documentation is fairly scarce, but it’s coming

Securing your private keys

ACES up your sleeve:

- C -- certify (“master”)
- E -- encrypt
- S -- sign
- A -- authenticate

The “C” key is your jewel

You should guard the “C” (“master”) key with particular care.

If you lose control of your master key, you will have to make a new one and go through the hassle of notifying people that your key has changed.

Everyone will be confused, you included.

Do not lose control of your master key.

E, S, A should be subkeys

- You should only be using your master key to create ("certify") new subkeys that you'll be using for actual work.
    - That, and sign other people's keys.

- Expiration dates on subkeys don't really matter
    - There are valid reasons to have short-lived subkeys, but unless you're a well-respected high-profile developer, using short-lived subkeys will probably just annoy everyone.

Your master key should be offline

- This used to be hard, but is much easier now
    - There’s plenty of docs, just google for “debian subkeys guide”
- Put your master key on (multiple) removable media
- You can print it out and OCR later as a last-resort backup

Your subkeys should live on a crypto token

- This used to be hard, but isn't any more
  - Lots of guides exist
- Crypto tokens:
  - GnuPG-capable smartcards (requires a reader)
  - USB tokens (basically, reader+smartcard)
- Don't generate keys on the card
  - RNG capability is much better on Linux
  - You should back them up before moving to card

The “A” subkey

- You can use the “A” subkey as your ssh private key
  - Requires replacing the ssh-agent with gpg-agent
  - Still annoyingly hard and you have to fight Gnome
  - Works great once you do overcome these hurdles
  - Of more interest to sysadmins

- Using an “A” key for your ssh needs is the best kind of 2-factor auth

PGP Best Practices (summary)

Good:

- Have a 4096-bit master key
- Have a strong passphrase
- Have 2048+ bit subkeys for S/E/A
- Sign all git tags
- Check sigs when applying PRs
- Use TOFU approach

Best:

- Store master key offline
- Store subkeys on a crypto token
- Sign all git commits
- Require all devs sign their commits
- Verify commits on merge
- Maintain a web of trust

Questions?

Securing the rest of your desktop

- Run a modern distro
- Apply updates ASAP
- Ditch `X11`
- Use `firejail` to sandbox apps

Run a modern distro

- Avoid fad distros and stick to something that’s been around for a while
- Pick distros that support SecureBoot
- Pick distros with plenty of community-driven support
- Upgrade to newer releases when old ones reach EOL
- It’s tempting to customize things heavily, but this complicates upgrades and forces you to stick to EOL versions

Apply updates ASAP

- Newer, unknown bugs offer better security than older, known bugs.
    - Certain conditions apply™.
- Set up notifications when new updates are available and install them promptly and on a regular basis.
- In most cases, a weekly update schedule is perfectly reasonable.
- Some updates require a reboot, so make sure you do.

Ditch X11 for Wayland

- X protocol gives the following access to any app that asks:
    - Entire screen contents
    - Full clipboard
    - All input events
- An attacker who can control a single app controls your full desktop
- Wayland solves most of the above (except clipboard)
- You can still run X-based apps under Wayland
- There is some functionality you may miss
    - Screen sharing, colour calibration, etc.. All to be fixed eventually.

Firejail Security Sandbox

- Uses seccomp-bpf and Linux namespaces to sandbox processes
  - Lightweight
  - Robust
  - Works out of the box

- Use it to sandbox applications that receive and process untrusted input

Entirely unbiased comparison

Malware:

Software that downloads and executes arbitrary code without user's knowledge or express permission. Has full access to the user's private files and passwords.

Browser:

Software that downloads and executes arbitrary code without user's knowledge or express permission. Has full access to the user's private files and passwords.

Not just browsers

- Chat software
  - Did you know that Slack is basically a reskinned Chromium?
- Email software
  - Did you know that Thunderbird is basically a reskinned Firefox?
- Video conferencing software
  - Did you know that it’s basically just a reskinned pile of crap?

Why do they need to...

...access these files?

- .ssh/*
- .gnupg/*
- Your git trees
- Your browser history
- Your local keyring

...perform these actions?

- sudo
- ps
- gpg --sign
- git commit -f

Your code and your coder reputation are only as protected as the junkiest desktop app, browser plugin, or browser extension you happen to have installed.

Firejail to the rescue

- Firejail sets up a namespace and a syscall filter for each app
  - Restricted view of $HOME with limited whitelisted dirs
  - Minimal /dev
  - Restricted /usr, /etc
  - Private /tmp
  - Minimal /usr/bin with only the binaries that app needs

- You can set up its own network namespace
  - Firejail can set up a separate IP just for the app
  - Can use a NAT-ing bridge if it's available (e.g. via libvirt)
  - You can also throttle an app's bandwidth

Firejail profiles

- Pre-written profiles exist for hundreds of commonly used Linux apps
  - Firefox
  - Thunderbird
  - Evolution
  - Evince
  - Mutt
  - Etc

- Quick demonstration

To sum it up

- PGP is your open-source world identity
- Use PGP signatures to clearly mark what is your code
- Take great care to protect your PGP keys
  - Move your master key to offline storage
  - Create subkeys for daily work
  - Move them to a smartcard device
- Take extra steps to protect yourself from misbehaving apps
  - Apps will have bugs
  - You will make mistakes
  - Use Firejail to protect yourself from both

Guide forthcoming

I am writing a longer guide aimed at developers using Linux.

It’s not quite ready to go (sorry!).

Please Github-watch this repo:

https://github.com/lfit/itpol/

(There is a guide for sysadmins there already!)

Thank you!

Questions?

Slides here: https://mricon.com/talks/osseu17.pdf

THE LINUX FOUNDATION

Contact Us

The Linux Foundation

1 Letterman Drive
Building D, Suite D4700
San Francisco CA 94129
Phone/Fax: +1 415 7239709
www.linuxfoundation.org

General Inquiries

info@linuxfoundation.org

Membership

membership@linuxfoundation.org

Corporate Training

training@linuxfoundation.org

Event Sponsorship

sponsorships@linuxfoundation.org

THE LINUX FOUNDATION

Legal Notices

THE LINUX FOUNDATION