The postmarketOS and Alpine Linux Conference 2026

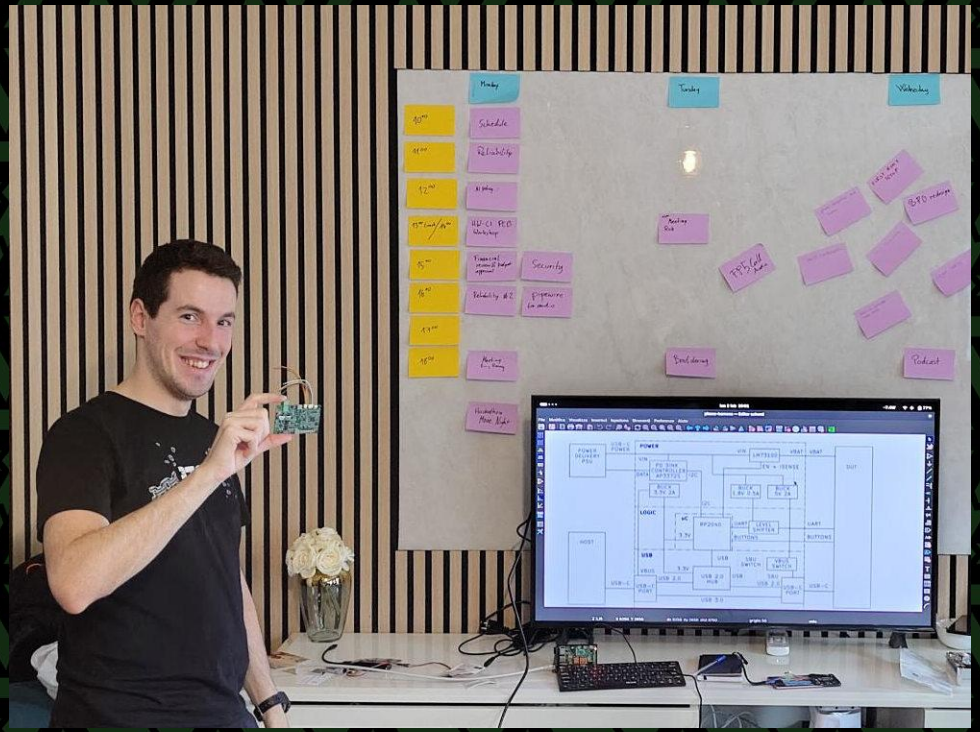




SELinux for postmarketOS

Achill and Aelin

How this all started



Discretionary Access Control (DAC)

- Restricts access based on the identity of the owner
- Access can be passed further on by anyone with access
- You should be familiar with owner-based DAC
  - Owner controls permissions to access an object
  - Most prominent example: Unix file permission bits

Basics of Mandatory Access Control (MAC)

- MAC has a different design
  - A *subject* gets permission to access/modify an *object*
  - Subjects are usually processes or threads
  - Objects can be files, directories, TCP/UDP ports, shared memory segments, or IO devices
  - *Policies* define the rules which *subjects* can access/modify which *objects*

- Why do we need it?
  - A lot of components are involved in a smartphone (e.g. mobile shell, userspace daemons,..)
  - Most runs either with root permissions or on the phones owner's user
  - If a single component is compromised, they have more permissions than they'd need

Different MAC implementations and their users

- SELinux
  - Used by Fedora, RHEL, openSUSE, Android, Gentoo (optional)
- AppArmor (see later)
  - Used by Debian, Ubuntu
- Tomoyo
  - Version merged into mainline is not at feature parity with earlier implementation
- SMACK
  - Used to be used by MeeGo and Tizen
- All implemented via the Linux Security Modules (LSM) framework
- All of them have absolute banger logos except AppArmor
  - Please don't base your choice of MAC implementation on the logo though

Basics of SELinux

- Explaining everything about SELinux is out of scope for this talk
- TL;DR:
  - Objects have labels in the format `user:role:type:level`
  - User, role and level are mainly used for MLS
  - SELinux policies define what is allowed on a system (e.g. processes labeled `httpd_t` can read files labeled `httpd_content_t`, but not `shadow_t`)
  - Policies can provide toggles via SELinux booleans (think of them like settings switches)
- There's an amazing presentation from Red Hat Summit, go watch it!



History of MAC in postmarketOS

- Even postmarketOS had AppArmor in 2021

- Used to “sandbox” user apps, instead of Flatpak/Portals (postmarketos#18)
- Dropped because of maintenance efforts & incompatibility with Portals (pmaports#1596)
- Since 2022, we don’t have a MAC implementation enabled

Comparison to AppArmor

- AppArmor can only restrict: read, write, append, execute, lock, link
  - As such, it is based on file paths rather than labels
- SELinux is orders of magnitude more powerful
  - e.g. loading kernel modules, binding to sockets, SHM access, ...
- AppArmor identifies files by path rather than inode
- SELinux requires more userspace integration (see later)
- SELinux requires filesystem support for xattrs (doesn't work on NFS)
- => AppArmor is simpler than SELinux, yet less stricter & less flexible

SELinux components

- Kernel built with LSM and SELinux support
- Init system must be aware of SELinux and load policies into the kernel at boot
  - Systemd and OpenRC both support this
- User-space packages
  - `libselinux` provides various SELinux-interfaces to make packages SELinux-aware
  - policycoreutils and libselinux-utils provide userspace tooling to modify the policy at runtime
  - SELinux-aware packages usually use the `-Z` flag to show SELinux context information
  - *Ideally* the package manager would also be aware (see later)
  - .. and more: https://github.com/SELinuxProject/selinux/wiki/Userspace-Packages

Policy packaging

- Different policies
  - refpolicy, Gentoo's refpolicy fork, Fedora's selinux-policy
- Different types of policies
  - Targeted: Specific services are restricted, others are unconfined
  - MLS (Multi-Level-Security): Implement different clearance levels
- Packaging
  - `testing/selinux-policy` is Fedora's policy adapted for Alpine Linux
  - Busybox hard to integrate with a SELinux policy, because of its single binary design
    - SELinux uses inodes to identify files
    - Thus it always has the same permissions no matter how it is invoked (good!)
    - ...but Fedora's policies use a different label for shell executables
    - ...which means ash cannot get this label, since it's just... busybox
    - "Everything and the kitchen sink" binaries are *bad* by design
  - How sustainable is this?

Package integration

- We now link various packages against `libselinux`
  - Including coreutils, linux-pam, networkmanager, gdm, …
  - e.g. that cp & mv re-set the contexts and `ls -Z` works
- It's always loaded also on non-SELinux systems, but is a no-op
  - `libselinux` is about ~160 KiB
  - Just like any other integration that requires .so linking
- Hope that keeping it out of busybox + OpenRC keeps Alpine happy

The current approach

- SELinux-enabled repository in extra-repos/selinux
  - Forks of a few packages in aports to link against `libselinux`
    - coreutils, procps-ng, sudo, openssh, shadow, …
    - All fully upstreamable to Alpine pending figuring some details out
- `postmarketos-base-selinux` metapackage
  - Pulls in all the SELinux utilities and… coreutils and bash
  - And an APK hook (more on that later)
  - Adds systemd services for autorelabling on certain conditions
- pmbootstrap prompts for SELinux if systemd is selected (off by default)
  - If enabled, it installs `postmarketos-base-selinux`
  - Also labels all files inside the generated images correctly
- Long-term goal is enabled-by-default in Duranium

apk-tools

- Our package manager needs to set labels for installed packages' files
- No native apk support for setting SELinux labels
- APK does have triggers
  - Packages can define a trigger-script & a list of directories
  - APK will run the script if any package's action modifies one of the directories
  - …but we don't know which *files* actually changed
  - It only tells your script which path rule matched
- Re-labeling *all files* is very slow and destroys your disk

- => We need apk to be able to tell triggers which files *triggered* the trigger-script

Demo

Join us!

#selinux:postmarketos.org