

An introduction to systemd security sandboxing

Marco d'Itri

`<md@seeweb.it>`

`@md@linux.it`

Seeweb s.r.l. | The DHH group

Romhack Camp 2026 - October 3, 2026



systemd sandboxing in one slide

systemd has many features to reduce the attack surface of unmodified system services. This is a different strategy from e.g. OpenBSD which modifies the programs to have themselves reduce their own privileges.

NoNewPrivileges

Prevents SUID and SGID programs from getting these privileges.
(Also, using `DynamicUser` enables `NoNewPrivileges`.)

General sandboxing

Restrict what resources the program can access.
Mainly the `Private*`, `Protect*` and `Restrict*` directives.

Also: capabilities, LSMs (SELinux, AppArmor, Landlock and more...).

Sandboxing: DynamicUser

- A unique UID and GID are allocated for each invocation of the service.
- They are not in `/etc/passwd` but resolved by `nss-systemd(8)`.
- The lifetime of temporary files and IPC objects is tied to the lifetime of the service.
- `NoNewPrivileges`, `ProtectSystem=strict` and `ProtectHome=read-only` are implicitly enabled.
- `/var/{lib,log,spool}/` are id-mapped to the dynamic UID.

This prevents the service from interacting with others.

Sandboxing: file system access

ProtectSystem

- `true`: `/usr`, `boot`, `efi` are read-only.
- `full`: **also** `/etc`.
- `strict`: **everything** is read-only (`ReadWritePaths` sets exceptions).

ProtectHome

- `read-only`: `/home` and `/root` are read-only.
- `true`: `/home` and `/root` are inaccessible.

RootEphemeral

The service runs each time on a new copy (CoW, if possible) of its root directory or image.

Sandboxing: more file system access

`PrivateTmp`

`/tmp/` and `/var/tmp/` are unique to the service.

`PrivateDevices`

`/dev/` is mostly empty and the service is prevented from creating new device nodes.

`ProtectProc` and `ProcSubset`

Make parts of `/proc` inaccessible.

More sandboxing

SystemCallFilter

Only allow the listed system calls. Standard sets are provided: e.g. @basic-io, @mount, @network-io, @signal.

CapabilityBoundingSet

Remove some capabilities, even if User=root.

PrivateNetwork

Prevents network access.

PrivateUsers

The service has unique users and groups, mapped to a different range from the rest of the system.

Even more sandboxing

`ProtectHostname, ProtectClock, ProtectKernelTunables,
ProtectKernelModules, ProtectKernelLogs, ProtectControlGroups`

As they say.

`RestrictAddressFamilies`

Only allow the listed AFs, e.g. `AF_INET6` or `AF_UNIX`.

`Limit*`

The usual `setrlimit(2)` features: CPU time, memory, processes, file descriptors...

Currently in Debian: the methodology

- Extract all `*.service` units from all packages in Debian unstable.
- Consider the units containing an `ExecStart` directive.
- Check which units use `NoNewPrivileges` (or `DynamicUser`).
- Check which units use `Private*`, `Protect*` or `Restrict*` (indicates an attempt at hardening).

Currently in Debian: the statistics

- 1608 service units.
- 238 use sandboxing but not `NoNewPrivileges`.
- 187 use both `NoNewPrivileges` and other sandboxing.
- 31 use only `NoNewPrivileges`.

`NoNewPrivileges` is not universally applicable, but why so many units do not implement at least some sandboxing?

SUID binaries considered harmful

The problem

SUID programs are influenced by the calling environment in many ways and making them secure is hard.

The solution

systemd instead likes to use non-privileged programs talking to privileged system service (e.g. `sudo` vs. `run0`).

The problem with `NoNewPrivileges`: email

Many daemons want to send emails, using the `/usr/sbin/sendmail` interface. The most common MTAs use SUID programs and `NoNewPrivileges` breaks them.

Postfix

By default, a SUID helper program (`postdrop`) writes a file to a queue directory for further processing.

Exim

A single monolithic daemon is started to directly write to the spool or even directly do local deliveries (writing to a file as another user!). The less is said, the better.

(There are still other SUID programs in a default install, but `systemd` provides alternatives or has a plan for most of them. Notably: `crontab` and `shadow`.)



The design of Postfix

Design requirement: mail **must not be lost** even if the daemon is not running.

Two options, both suboptimal

- `/usr/sbin/postdrop` is SGID.
- `/var/spool/postfix/maildrop/` is world writeable (but this allows a denial of service by local users).

Upstream is working on a solution using a socket-activated postfix component.

Any questions?



`https://www.linux.it/~md/text/debian-sandboxing-asg2026.pdf`
(Google ... Marco d'Itri ... I'm feeling lucky)

