Application Security

Sandboxing features of the Linux kernel and systemd

2025-05-22, NLUUG (Utrecht) - Michael Boelen

$ whoami

- Michael Boelen
  - Blogger
    - Linux Audit
    - linux.vooreenbeginner.nl
    - meereco.nl
  - Interests
    - Linux and Unix
    - Information Security
    - Chess ♟
    - Metal detecting
  - Open source developer
    - ~~rkhunter~~ / Lynis
  - Volunteer
    - Communication Commission @ NLLGG
    - Webmaster @ NLUUG
    - Participation council (primary school)
    - Neighborhood watch program
  - More at michaelboelen.com
  - Mastodon: @mboelen

Before we begin

- Questions?
  - Short:
  - Long: at the end
- Slides will be published
- Share your insights today:
  - Tag **@mboelen** and **#nluug** at Mastodon

This talk

- Make applications a bit more secure
  - Reduce attack surface
  - Limit impact
- Using the features of:
  - Linux (kernel)
  - systemd
- **Goals**:
  - More secure software implementations
  - Knowledge sharing
  - Cooperation (system hardening)

The problem

```
$ systemd-analyze security
UNIT                                     EXPOSURE PREDICATE HAPPY
ModemManager.service                          6.3 MEDIUM    😐
NetworkManager.service                        7.8 EXPOSED   🙁
accounts-daemon.service                       5.5 MEDIUM    😐
alsa-state.service                            9.6 UNSAFE    😨
anacron.service                               9.6 UNSAFE    😨
avahi-daemon.service                          9.6 UNSAFE    😨
blueman-mechanism.service                     9.6 UNSAFE    😨
colord.service                                3.5 OK        🙂
cron.service                                  9.6 UNSAFE    😨
cups-browsed.service                          9.3 UNSAFE    😨
cups.service                                  9.6 UNSAFE    😨
dbus.service                                  9.5 UNSAFE    😨
dm-event.service                              9.5 UNSAFE    😨
dmesg.service                                 9.6 UNSAFE    😨
emergency.service                             9.5 UNSAFE    😨
getty@tty1.service                            9.6 UNSAFE    😨
getty@tty7.service                            9.6 UNSAFE    😨
irqbalance.service                            8.9 EXPOSED   🙁
kerneloops.service                            9.2 UNSAFE    😨
lightdm.service                               9.6 UNSAFE    😨
lvm2-lvmpolld.service                         9.5 UNSAFE    😨
mintsystem.service                            9.6 UNSAFE    😨
networkd-dispatcher.service                   9.6 UNSAFE    😨
plymouth-halt.service                         9.5 UNSAFE    😨
plymouth-poweroff.service                     9.5 UNSAFE    😨
plymouth-reboot.service                       9.5 UNSAFE    😨
plymouth-start.service                        9.5 UNSAFE    😨
polkit.service                                1.6 OK        🙂
power-profiles-daemon.service                 5.1 MEDIUM    😐
rc-local.service                              9.6 UNSAFE    😨
rescue.service                                9.5 UNSAFE    😨
```

Solution: Application security

- **Generic**
  - Patch
  - Sane defaults
  - Application-specific security measures
- **External barriers**
  - File system
  - Security frameworks
  - Sandboxing

Linux kernel

- Security modules / Frameworks
  - AppArmor / SELinux
- Namespaces
- Control Groups (cgroups)
- Secure Computing (seccomp / seccomp-bpf)

Linux kernel

- **AppArmor**
  - Debian / Ubuntu
  - Fairly easy to use
- **SELinux**
  - RHEL / Fedora
  - Not so easy to use for beginners [1]

[1] If there are presentations "SELinux is easy", that's a hint

Linux kernel

- **Namespaces**
  - The Matrix!
  - Available namespaces
    - Inter-Process Communication
    - Mount
    - Processes
    - Time
    - Users
    - *and others...*

Linux kernel

- **cgroups (Control Groups)**
  - **Limit resources**
    - Reduce access
  - **Prioritize resources**
    - Increase amount of time
  - **Accounting**
    - Measure, billing
  - **Control**
    - Freeze, snapshot, resume

Linux kernel

- **seccomp**
  - Secure Computing
  - Defines which **system calls** are allowed
  - Policy: **continue** or **instant kill**
  - Great for **sandboxing**
  - Examples: Flatpak, systemd, snap
- **seccomp-bpf**
  - Similar to seccomp, slightly different implementation
  - Using *Berkeley Packet Filter*
  - Examples: Android, Chrome, Firefox, OpenSSH

Sandboxing

- Sandbox usage is still limited
  - Web browsers
  - Security-minded tools
  - Sandbox tools
    - Firejail
    - Bubblewrap
  - And… **systemd**!

systemd

- systemd and seccomp =
  - systemd-nspawn (containers)
  - **service units**

Let’s take one step back: systemd in a nutshell

systemd

- What is systemd?
  - System manager
  - Service manager

systemd

- Units
  - timer
  - **service**
  - automount
  - mount
  - path
  - *etc.*

systemd

- **Management**
  - systemctl **cmd** MYAPP.service
    - start | stop | restart | reload | reload-or-restart
    - enable | disable | mask | unmask
- **Configuration**
  - systemctl **cmd** MYAPP.service
    - cat | edit
- **Logging**
  - systemctl status MYAPP.service
  - journalctl -u MYAPP.service

systemd

- Getting more out of systemd
  - **Units**
    - Linux Audit: Overview of systemd units
  - **Cheat sheets**
    - Linux Audit: journalctl
    - Linux Audit: systemctl

systemd

- Changing existing units
  - **Override**
    - Purpose: complements initial configuration
  - **How?**
    - systemctl edit --full myapplication.service
    - **systemctl edit myapplication.service**
    - /usr/lib/systemd/system/myapplication.service.d/*.conf
    - /etc/systemd/system/myapplication.service.d/*.conf

systemd

Editing **/etc/systemd/system/cron.service.d/override.conf**

Anything between here and the comment below will become the new contents of the file

Lines below this comment will be discarded

/lib/systemd/system/cron.service
```
# [Unit]
# Description=Regular background program processing daemon
# Documentation=man:cron(8)
# After=remote-fs.target nss-user-lookup.target
#
# [Service]
# EnvironmentFile=-/etc/default/cron
# ExecStart=/usr/sbin/cron -f $EXTRA_OPTS
# IgnoreSIGPIPE=false
# KillMode=process
# Restart=on-failure
#
# [Install]
# WantedBy=multi-user.target
```

systemd

```
### Editing /etc/systemd/system/cron.service.d/override.conf

### Anything between here and the comment below will become the new contents of the file

[Service]
CPUSchedulingPolicy=idle
IOSchedulingClass=idle
Nice=19

### Lines below this comment will be discarded

### /lib/systemd/system/cron.service
# [Unit]
# Description=Regular background program processing daemon
# Documentation=man:cron(8)
# After=remote-fs.target nss-user-lookup.target
#
# [Service]
# EnvironmentFile=-/etc/default/cron
# ExecStart=/usr/sbin/cron -f $EXTRA_OPTS
# IgnoreSIGPIPE=false
# KillMode=process
# Restart=on-failure
#
# [Install]
# WantedBy=multi-user.target
```

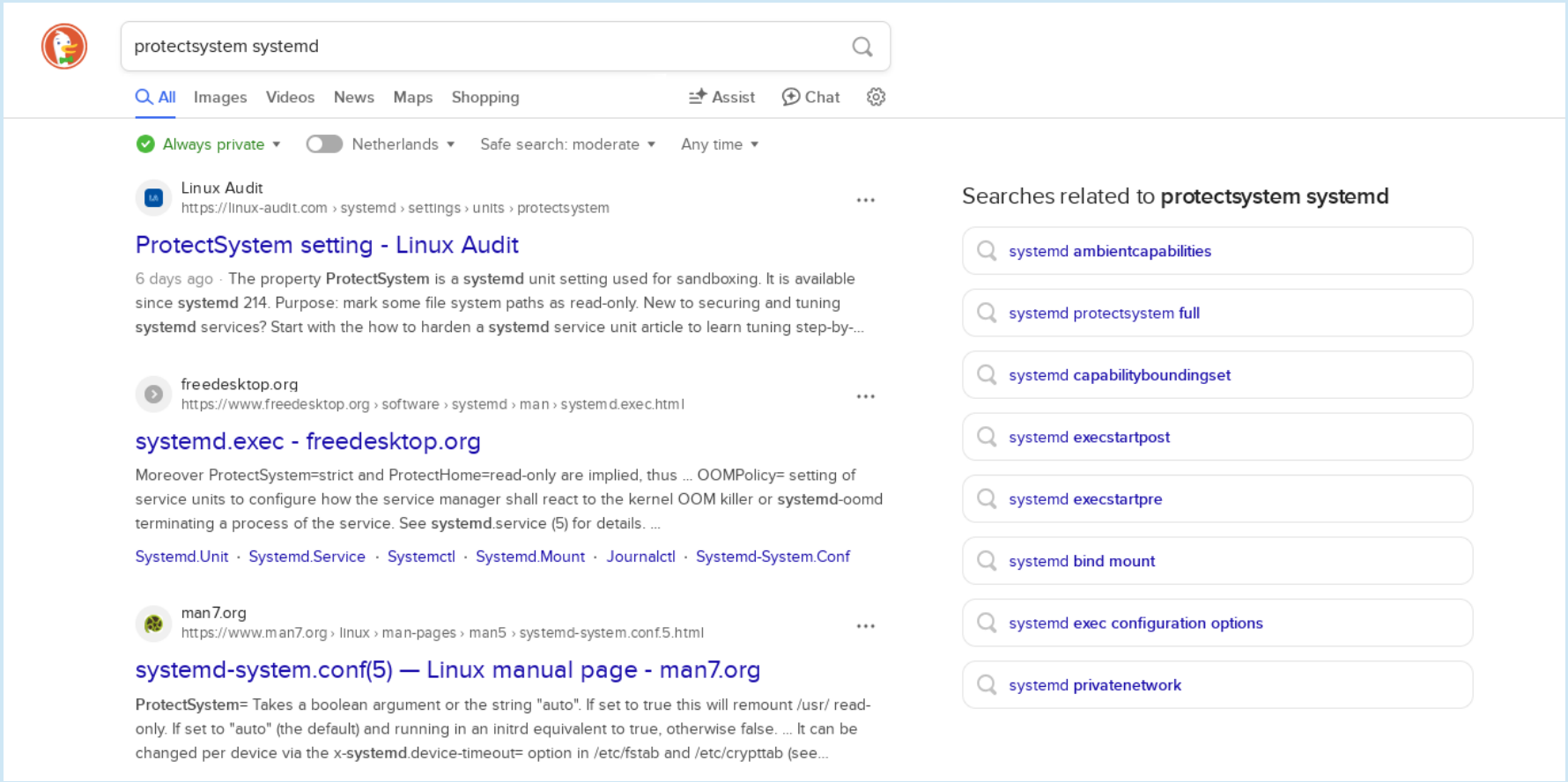
systemd

- Example: **ProtectSystem** (property)
  - Marks parts of the filesystem as read-only
  - Very powerful
  - But…
    - How to discover this property?
    - How to configure?
    - When (not) to use?

systemd

- Security options for units
  - 40+ properties (and counting)
  - Most are for sandboxing
  - Available in official documentation, but...
    - Technical
    - Difficult
    - Lack of examples

systemd


protectsystem systemd
All
Images
Videos
News
Maps
Shopping
Assist
Chat
Always private
Netherlands
Safe search: moderate
Any time
Linux Audit
https://linux-audit.com › systemd › settings › units › protectsystem
ProtectSystem setting - Linux Audit
6 days ago · The property ProtectSystem is a systemd unit setting used for sandboxing. It is available since systemd 214. Purpose: mark some file system paths as read-only. New to securing and tuning systemd services? Start with the how to harden a systemd service unit article to learn tuning step-by-...
freedesktop.org
https://www.freedesktop.org › software › systemd › man › systemd.exec.html
systemd.exec - freedesktop.org
Moreover ProtectSystem=strict and ProtectHome=read-only are implied, thus ... OOMPolicy= setting of service units to configure how the service manager shall react to the kernel OOM killer or systemd-oomd terminating a process of the service. See systemd.service (5) for details. ...
Systemd.Unit · Systemd.Service · Systemctl · Systemd.Mount · Journalctl · Systemd-System.Conf
man7.org
https://www.man7.org › linux › man-pages › man5 › systemd-system.conf.5.html
systemd-system.conf(5) — Linux manual page - man7.org
ProtectSystem= Takes a boolean argument or the string "auto". If set to true this will remount /usr/ read-only. If set to "auto" (the default) and running in an initrd equivalent to true, otherwise false. ... It can be changed per device via the x-systemd.device-timeout= option in /etc/fstab and /etc/crypttab (see...
Searches related to protectsystem systemd
systemd ambientcapabilities
systemd protectsystem full
systemd capabilityboundingset
systemd execstartpost
systemd execstartpre
systemd bind mount
systemd exec configuration options
systemd privatenetwork

systemd

Name

systemd.exec — Execution environment configuration

Synopsis

service`.service`, *socket*`.socket`, `mount.mount`, `swap.swap`

Description

Unit configuration files for services, sockets, mount points, and swap devices share a subset of configuration options which define the execution environment of spawned processes.

This man page lists the configuration options shared by these four unit types. See systemd.unit(5) for the common options of all unit configuration files, and systemd.service(5), systemd.socket(5), systemd.swap(5), and systemd.mount(5) for more information on the specific unit configuration files. The execution specific configuration options are configured in the [Service], [Socket], [Mount], or [Swap] sections, depending on the unit type.

In addition, options which control resources through Linux Control Groups (cgroups) are listed in systemd.resource-control(5). Those options complement options listed here.

Implicit Dependencies

A few execution parameters result in additional, automatic dependencies to be added:

- Units with `WorkingDirectory=`, `RootDirectory=`, `RootImage=`, `RuntimeDirectory=`, `StateDirectory=`, `CacheDirectory=`, `LogsDirectory=` or `ConfigurationDirectory=` set automatically gain dependencies of type `Requires=` and `After=` on all mount units required to access the specified paths. This is equivalent to having them listed explicitly in `RequiresMountsFor=`.
- Similarly, units with `PrivateTmp=` enabled automatically get mount unit dependencies for all mounts required to access `/tmp/` and `/var/tmp/`. They will also gain an automatic `After=` dependency on systemd-tmpfiles-setup.service(8).
- Units whose standard output or error output is connected to `journal` or `kmsg` (or their combinations with console output, see below) automatically acquire dependencies of type `After=` on `systemd-journald.socket`.
- Units using `LogNamespace=` will automatically gain ordering and requirement dependencies on the two socket units associated with `systemd-journald@.service` instances.

Paths

The following settings may be used to change a service's view of the filesystem. Please note that the paths must be absolute and must not contain a "`..`" path component.

`ExecSearchPath=`

Takes a colon separated list of absolute paths relative to which the executable used by the `Exec*=` (e.g. `ExecStart=`, `ExecStop=`, etc.) properties can be found. `ExecSearchPath=` overrides `$PATH` if `$PATH` is not supplied by the user through `Environment=`, `EnvironmentFile=` or `PassEnvironment=`. Assigning an empty string removes previous assignments and setting `ExecSearchPath=` to a value multiple times will append to the previous setting.

Added in version 250.

`WorkingDirectory=`

Takes a directory path relative to the service's root directory specified by `RootDirectory=`, or the special value "`~`". Sets the working directory for executed processes. If set to "`~`", the home directory of the user specified in `User=` is used. If not set, defaults to the root directory when systemd is running as a system instance and the respective user's home directory if run as user. If the setting is prefixed with the "`-`" character, a missing working directory is not considered fatal. If `RootDirectory=`/`RootImage=` is not set, then `WorkingDirectory=` is relative to the root of the system running the service manager. Note that setting this parameter might result in additional dependencies to be added to the unit (see above).

`RootDirectory=`

Takes a directory path relative to the host's root directory (i.e. the root of the system running the service manager). Sets the root directory for executed processes, with the pivot_root(2) or chroot(2) system call. If this is used, it must be ensured that the process binary and all its auxiliary files are available in the new root. Note that setting this parameter might result in additional dependencies to be added to the unit (see above).

The `MountAPIVFS=` and `PrivateUsers=` settings are particularly useful in conjunction with `RootDirectory=`. For details, see below.

If `RootDirectory=`/`RootImage=` are used together with `NotifyAccess=` the notification socket is automatically mounted from the host into the root environment, to ensure the notification interface can work correctly.

Note that services using `RootDirectory=`/`RootImage=` will not be able to log via the syslog or journal protocols to the host logging infrastructure, unless the relevant sockets are mounted from the host, specifically:

The host's os-release(5) file will be made available for the service (read-only) as `/run/host/os-release`. It will be updated automatically on soft reboot (see: systemd-soft-reboot.service(8)), in case the service is configured to survive it.

Example 1. Mounting logging sockets into root environment

```
BindReadOnlyPaths=/dev/log /run/systemd/journal/socket /run/systemd/journal/stdout
```

In place of the directory path a "`.v/`" versioned directory may be specified, see systemd.v(7) for details.

This option is only available for system services, or for services running in per-user instances of the service manager in which case `PrivateUsers=` is implicitly enabled (requires unprivileged user namespaces support to be enabled in the kernel via the "`kernel.unprivileged_userns_clone=`" sysctl).

`RootImage=`

Takes a path to a block device node or regular file as argument. This call is similar to `RootDirectory=` however mounts a file system hierarchy from a block device node or loopback file instead of a directory. The device node or file system image file needs to contain a file system without a partition table, or a file system within an MBR/MS-DOS or GPT partition table with only a single Linux-compatible partition, or a set of file systems within a GPT partition table that follows the Discoverable Partitions Specification.

When `DevicePolicy=` is set to "`closed`" or "`strict`", or set to "`auto`" and `DeviceAllow=` is set, then this setting adds `/dev/loop-control` with `rw` mode, "`block-loop`" and "`block-blkext`" with `rwm` mode to `DeviceAllow=`. See systemd.resource-control(5) for the details about `DevicePolicy=` or `DeviceAllow=`. Also, see `PrivateDevices=` below, as it may change the setting of `DevicePolicy=`.

Units making use of `RootImage=` automatically gain an `After=` dependency on `systemd-udevd.service`.

The host's os-release(5) file will be made available for the service (read-only) as `/run/host/os-release`. It will be updated automatically on soft reboot (see: systemd-soft-reboot.service(8)), in case the service is configured to survive it.

In place of the image path a "`.v/`" versioned directory may be specified, see systemd.v(7) for details.

This option is only available for system services and is not supported for services running in per-user instances of the service manager.

Added in version 233.

systemd

Setting	Description	Available since
CapabilityBoundingSet	Define what capabilities are allowed within the service unit	21
DeviceAllow	Allow access to a device	208
DevicePolicy	Define level of access to devices in /dev	208
ExecPaths	Define the paths from which programs can be executed	231
InaccessiblePaths	Define paths that should not be accessible	231
IPAccounting	Define if accounting on network packets and bytes should be used	235
KeyringMode	Controls kernel session keyring and define what is available to the service	235
LockPersonality	Prevent processes switching their personality, a kernel execution domain	235
MemoryDenyWriteExecute	Block creation or alteration of memory segments to become writable and executable as well	231
NoExecPaths	Exclude paths from which programs can be executed	231
NoNewPrivileges	Prevent processes from gaining new privileges	187
PrivateDevices	Only allow access to a subset of devices in /dev	209
PrivateMounts	Provides a separated mount namespace to the service	239
PrivateNetwork	Defines if access to the network interfaces of the host is possible	33
PrivatePIDs	Define a new PID namespace for the process and its children	257
PrivateTmp	Define new namespace for /tmp and /var/tmp directory	1
PrivateUsers	Define a new user namespace for the process and its children	232
ProcSubset	Define the subset of access by unit to /proc	247
ProtectClock	Limit access to clock information	245
ProtectControlGroups	Limit write access to control groups directory structure under /sys/fs/cgroup	232
ProtectHome	Define what level of access is possible to home directories	214
ProtectHostname	Defines if hostname or NIS domain name can be changed	242

systemd

- Providing an alternative:
  - **Guides**
    - systemd service hardening
    - Resolving basic issues with failed systemd service
    - Steps to take when a service unit fails after hardening
  - **Unit settings** (properties)
    - Overview
    - + Alternative description
    - + Examples
  - **Ready-to-use[1] profiles**
    - Hardening profiles

[1] may need adjustments depending on your distribution and configuration

systemd

```
###############################################################################
# Source: https://linux-audit.com/systemd/hardening-profiles/nginx/
# Profile version: 0.4 [2025-01-06]
###############################################################################
# Customizations:
# - Insert here the changes you made to the profile
###############################################################################

[Service]
# ==============================
# Paths
# ==============================
# Deny access to /dev/shm directory, suggested when using MemoryDenyWriteExecute=yes
# Details: https://linux-audit.com/systemd/settings/units/inaccessiblepaths/
InaccessiblePaths=/dev/shm
# Do not allow execution of files, except nginx itself
# Details: https://linux-audit.com/systemd/settings/units/noexecpaths/
NoExecPaths=/
# Details: https://linux-audit.com/systemd/settings/units/execpaths/
ExecPaths=/usr/sbin/nginx /usr/lib
# Allow creation of PID file and writing to log files (access|error).log
# Details: https://linux-audit.com/systemd/settings/units/readwritepaths/
ReadWritePaths=/run /var/log/nginx

# ==============================
# Capabilities and system calls
# ==============================
# Only allow: bind to ports < 1024, change file permissions/ownership so nginx workers can use them
# Details: https://linux-audit.com/systemd/settings/units/capabilityboundingset/
CapabilityBoundingSet=CAP_NET_BIND_SERVICE CAP_CHOWN CAP_DAC_OVERRIDE CAP_SETGID CAP_SETPCAP CAP_SETUID
```

systemd

Be aware after enabling new settings: Errors

The best kind of errors:

- Quick
- Explosive
- Instill fear and doubt

Conclusion: Awesome errors!

systemctl restart nginx.service

Job for nginx.service failed because the control process exited with error code.

See "systemctl status nginx.service" and "journalctl -xeu nginx.service" for details.

systemd

```
# systemctl restart nginx.service
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xeu nginx.service" for details.
# systemctl status nginx.service
× nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
    Drop-In: /etc/systemd/system/nginx.service.d
             └─override.conf
     Active: failed (Result: exit-code) since Tue 2025-03-11 14:22:15 CET; 2min 8s ago
   Duration: 2d 4h 36.240s
       Docs: man:nginx(8)
    Process: 3235 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=203/EXEC)
        CPU: 26ms

Mar 11 14:22:14 debian-test systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Mar 11 14:22:15 debian-test (nginx)[3235]: nginx.service: Failed to execute /usr/sbin/nginx: Permission denied
Mar 11 14:22:15 debian-test (nginx)[3235]: nginx.service: Failed at step EXEC spawning /usr/sbin/nginx: Permission denied
Mar 11 14:22:15 debian-test systemd[1]: nginx.service: Control process exited, code=exited, status=203/EXEC
Mar 11 14:22:15 debian-test systemd[1]: nginx.service: Failed with result 'exit-code'.
Mar 11 14:22:15 debian-test systemd[1]: Failed to start nginx.service - A high performance web server and a reverse proxy server.
```

Question:
What could be the cause of the 'permission denied'?

systemd

```
# systemctl restart nginx.service
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xeu nginx.service" for details.
# systemctl status nginx.service
× nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
    Drop-In: /etc/systemd/system/nginx.service.d
             └─override.conf
     Active: failed (Result: exit-code) since Tue 2025-03-11 14:22:15 CET; 2min 8s ago
   Duration: 2d 4h 36.240s
       Docs: man:nginx(8)
    Process: 3235 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=203/EXEC)
        CPU: 26ms

Mar 11 14:22:14 debian-test systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Mar 11 14:22:15 debian-test (nginx)[3235]: nginx.service: Failed to execute /usr/sbin/nginx: Permission denied
Mar 11 14:22:15 debian-test (nginx)[3235]: nginx.service: Failed at step EXEC spawning /usr/sbin/nginx: Permission denied
Mar 11 14:22:15 debian-test systemd[1]: nginx.service: Control process exited, code=exited, status=203/EXEC
Mar 11 14:22:15 debian-test systemd[1]: nginx.service: Failed with result 'exit-code'.
Mar 11 14:22:15 debian-test systemd[1]: Failed to start nginx.service - A high performance web server and a reverse proxy server.
```

Solution: ExecPaths=/usr/sbin/nginx **/usr/lib**

systemd

- **Troubleshooting**
  - Generic tips
    - Apply in small steps
    - systemctl status myapplication.service
    - journalctl -u myapplication.service
  - Capabilities and system calls
    - Use **ldd** and **strace**
    - Use property **SystemCallLog**
      - SystemCallLog=~@system-service chroot

Let's harden some services together!

- **Help the project**
  - What services do you run?
  - Server or desktop?
- **Contact**
  - Details: michaelboelen.com
  - @mboelen

Thank you!