Engineer

Privilege

🎟 a right a system grants to a person or process

What it is

A privilege is permission to do something a system otherwise forbids, held by a person or by a running program because an authority decided it should have it: read another user's files, bind a port below 1024, create accounts, run a database function. People speak of privileges in the plural, because a subject rarely holds just one, and of "privileged" and "unprivileged" accounts or processes to say which side of that line something sits on. NIST's glossary adds the part that is easy to forget: privileges describe the authorised behaviour of a subject, and they are defined by an authority and written down in policy or rules.

The word sits next to two others. Access control is the act of deciding each request; a privilege is one of the facts that decision consults about whoever is asking. A permission is usually the same idea seen from the resource's side: the file says who may read it, and the reader holds the right to do so.

On Linux the word has a precise meaning. The capabilities(7) manual describes the traditional Unix split: privileged processes, which run with an effective user ID of 0 and are known as root, bypass all kernel permission checks, while every other process is checked against its credentials. Since Linux 2.2 the kernel has divided root's power into separate units called capabilities, which can be switched on and off one at a time, so a program that needs only to bind a low port can be given `CAP_NET_BIND_SERVICE` and nothing else. The credentials(7) page explains whose identity is being checked: the real user ID says who owns a process, the effective user ID is what the kernel compares against permissions, and the saved set-user-ID lets a program set aside a raised identity and pick it up again later.

Databases and organisations use the same word for their own grants. A database role holds privileges on schemas, tables and functions. An administrator account holds privileges a staff account does not. The idea is identical at every level: a right, granted on purpose, to cross a line that is otherwise closed.

Use the concept in a decision

Put “What is a privilege?” to work in a real decision. We can help relate the concept to the system you are building or buying. Accounts, passwords and permissions arranged so people hold exactly the access their job needs, and leavers lose it that day.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Wrote 1,000 RBAC rules to secure Linux servers — 1,000 access rules written to close a security gap

Wrote 1,000 RBAC rules to secure Linux servers Get in touch →

Why it matters

The privileges a program holds are the size of the damage it can do once something goes wrong with it. That is the stated aim of capabilities: a program that has been compromised does less harm with a narrow set of rights than the same program running as root. A web server with a code execution bug and root privilege hands the whole machine to whoever found the bug; the same bug in a process that can read its own directory and bind one port reaches very little.

NIST writes this down as least privilege. Control AC-6 in SP 800-53 asks an organisation to allow only the accesses that users, and the processes acting for them, need to do their assigned tasks. Its enhancements turn that into routine work: administrators use an unprivileged account for everyday tasks such as email, privileged accounts are restricted to the people who need them, the use of privileged functions is logged, and privileges are reviewed and removed when the need for them can no longer be shown.

The defaults rarely start there. In systemd, a system service with no `User=` line runs as root, and `NoNewPrivileges=`, the setting that stops a service and its children from ever gaining more through `execve()`, is off unless someone turns it on. Database tools often connect as the owner of every table because that is how the schema was created. New staff are often copied from an existing colleague’s account, rights and all. Each default is convenient on the first day and expensive on the day something is breached.

The trade-off is real. Narrowing privileges takes time, because someone has to find out what a program actually does, and a set that is too narrow breaks things at runtime rather than at install time. The decision a team faces is usually where to spend that time first: on anything that faces the internet, on anything holding data people would miss, and on the accounts that can change the other accounts. For a buyer or an auditor the useful question is simple and direct: who and what holds administrator rights here, and why.

What it looks like in practice

On the company's own host, thirteen unit files ran with the privileges the supervisor gives by default, which is most of them. Every unit template gained a privilege ceiling fixed at the unit's own, a system-call filter, restricted address families, and memory that is writable or executable but never both. The hardening score for the dead-man's switch went from 9.6 to 1.5, the git forge from 8.3 to 1.5, and the two backup units from 9.6 to 2.3 and 2.5. One directive turned out to be misspelled in three templates, and the supervisor had been logging an unknown key and starting the unit anyway, so every unit is now run through the supervisor's own verifier during each converge. The page also records what the score leaves out: a directive that breaks a unit does so at runtime.

On an application platform, the API logs in to the database as a least-privilege role that can only EXECUTE the app functions, which run as SECURITY DEFINER with parameterised queries. Entitlements such as tier, role, organization and ship scoping are re-checked on the server on every request, and the gates in the interface are treated as UX only.

Across one organisation, identities and access were consolidated across M365, 1Password, Red Hat SSO and OKTA SSO, least privilege was enforced, and onboarding and offboarding were standardised so that access was revoked promptly instead of lingering after someone moved on.

Where it goes wrong

A service runs as root because its unit says nothing. The supervisor's default for a system service is root, so a unit without a `User=` line holds every privilege the machine has. Give each service its own user, or let systemd allocate one with `DynamicUser=`, which creates a user when the unit starts and releases it when the unit stops.

One capability turns out to be most of root. `CAP_SYS_ADMIN` covers so many operations that its own manual calls it overloaded and “the new root”. A team that grants it to unlock one feature has granted most of root. Look for the specific capability the program needs, and keep the bounding set to that list:

```
[Service]
User=web
NoNewPrivileges=yes
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

Those settings let a service running as an ordinary user bind port 80, and every other capability is removed from its bounding set.

A process climbs above where it started. A program running as an ordinary user can still execute a set-user-ID binary or a file with capabilities attached, and the kernel raises its privileges for that run. Defenders recognise this as a process whose effective user ID differs from its real one. `NoNewPrivileges=yes` closes that route for a service and every child it starts.

Rights pile up as people change jobs. Someone moves teams and keeps the old access alongside the new, and after a few years their account holds the union of every role they ever had. AC-6 answers this with periodic review, and AC-2 with account management tied to transfers and departures, so that a change of job is also a change of access.

The application’s own login can do everything. When an API connects as the owner of the schema, a single injection flaw reaches every table. A login role that can only call the functions the application uses turns the same flaw into a much smaller problem.

Further reading

- Privilege, in the NIST CSRC glossary — the CNSSI 4009 definition, with the note that an authority defines privileges.
- capabilities(7) — how Linux splits root into separate capabilities and how threads carry them.
- credentials(7) — real, effective, saved and filesystem user IDs, and what each one decides.

- systemd.exec(5) — `User=`, `DynamicUser=`, `NoNewPrivileges=` and the capability settings.
- NIST SP 800-53 Rev. 5 — control AC-6, least privilege, and the enhancements that make it routine.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is a privilege?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/privilege/