ARTICLE

How to Install and Configure SELinux on CentOS 8

SELinux is one of the most effective containment controls on CentOS 8, but only if it is enabled, configured correctly, and validated before production. This article explains what changes when you turn it on, how to verify policy mode and labels, and how to decide whether to run enforcing or permissive in your environment.

Operating Systems - August 03, 2026

Why SELinux installation and configuration matters on CentOS 8

The practical problem is simple: a service can be installed, started, and still fail later because mandatory access control is not enabled or is configured in a way that does not match the workload. On CentOS 8, SELinux is not an optional decoration for hardened systems; it is a containment layer that limits what a process can do even if that process is compromised or misbehaving.

If you are preparing a server for production, you need to know more than whether SELinux is present. You need to know whether it is installed, whether the policy is active, whether the system is in enforcing or permissive mode, whether file contexts are correct, and what to verify before you trust the machine with live traffic. After reading this article, you will be able to determine whether SELinux is applicable to your CentOS 8 system, apply a practical validation workflow, and decide what to check before production use.

Key takeaways

SELinux on CentOS 8 is usually already available as part of the base system, but the operational task is to confirm that the policy is installed, enabled at boot, and aligned with the services you run. The most important distinctions are:

- Enforcing means policy decisions are actively blocking unauthorized access.
- Permissive means SELinux logs denials without blocking them, which is useful for testing.
- Disabled means SELinux is not participating at all, which removes the control entirely.

For most production workloads, the right approach is to keep SELinux enabled, validate service behavior in permissive mode if needed, fix labeling or policy issues, and only then move to enforcing. If you are also tightening the host around SSH, pair this work with How to Harden CentOS 8 SSH Access with Key Authentication so authentication and access control are treated as complementary controls rather than separate hardening tasks.

How SELinux works on CentOS 8

SELinux applies a mandatory access control policy to processes, files, ports, and other system objects. Instead of relying only on Unix permissions and service-level safeguards, it adds context-based rules that can prevent a process from reading, writing, binding, or transitioning into an unexpected domain.

On CentOS 8, the most relevant operational concepts are the policy mode, the file label, and the service domain. A daemon does not simply run as a user account; it runs within a SELinux context that determines what it can access. That is why a service can appear healthy from systemd?s perspective while still failing to serve content, write logs, or bind to a port because SELinux denies the action.

This is also why installation and configuration are not just package tasks. They are validation tasks. A correctly installed SELinux policy with the wrong file labels is still a broken deployment.

A compact workflow for installing and configuring SELinux

The following workflow is intentionally compact. It is not a full remediation playbook, but it is the fastest way to confirm that SELinux is installed, active, and ready for controlled use.

The operational logic is straightforward: confirm the policy package is installed, verify the kernel and user-space view of the mode, test the workload, read the denials, and fix the cause rather than bypassing the control.

Installing or verifying the SELinux components

On CentOS 8, SELinux support is normally part of the system base, so the main concern is verification rather than a fresh standalone installation. The packages that matter most in day-to-day administration are the policy, utilities, and labeling tools. A minimal check is enough to confirm that the host has the expected components:

If a package is missing, install it using the standard package manager for your environment. The important point is not the command itself, but the expectation that policy, labels, and troubleshooting tools all need to be available together. Without them, you cannot inspect denials or correct mislabeled files in a disciplined way.

If your environment is built from minimal images, golden templates, or stripped-down cloud instances, verify SELinux support before the system is handed to application teams. In many operational failures, the issue is not that SELinux is unavailable; it is that the team discovers too late that the host was built with inconsistent policy expectations.

Configuring the active mode

The main configuration file is /etc/selinux/config, which determines the default mode for future boots. The system can be set to enforcing, permissive, or disabled. In practice, the choice should follow your deployment stage and risk tolerance.

A typical configuration looks like this:

The targeted policy is the standard baseline for most servers because it confines many network-facing services without trying to police every user process equally. That default is often the right starting point for general-purpose CentOS 8 servers.

The mode change you make in the config file is not enough by itself if the running kernel state differs. Use both sestatus and getenforce to confirm what is active now, and reboot only when you intend to make the default persistent. In other words, the config file defines the boot-time intent, but the current runtime mode is the operational fact.

What to verify after installation or mode changes

The critical validation is not whether the file was edited successfully; it is whether the runtime behaves as expected. After any SELinux mode change, confirm the following:

- The current mode is what you intended.
- The policy type matches your server role.
- Denials are either absent or explainable.
- Services that depend on nonstandard paths, ports, or identities still function.
- Required labels are present on application data, logs, and content roots.

For example, a web server that serves files from a custom document root may need the correct file context before it can read content. A database service writing to a relocated data directory may require label correction if the directory was created manually or restored from backup.

This is where many operators move too quickly. They see a denial, disable the control, and move on. A more reliable approach is to identify the exact object and action involved, then determine whether the fix is a relabel, a port context adjustment, or a policy update. If your use case requires service-specific policy work, How to Configure SELinux Policies on CentOS 8 covers that part of the workflow in more detail.

Practical scenario: a custom application server that stops reading its data directory

Consider a CentOS 8 host running a custom application under systemd. The service starts correctly after deployment, but the application cannot read files from /srv/appdata because the directory was created during a migration and inherited an unlabeled or incorrect context. The application logs show permission-like failures, but Unix ownership and mode bits appear correct.

This is a typical SELinux symptom pattern. The process has enough classic filesystem permission to access the path, but SELinux denies the access because the file context does not match what the service domain is allowed to use. In this case, the likely remedy is not a broad security exception. It is to inspect the label and relabel the directory so the service can access the expected type safely.

That distinction matters operationally. A label fix preserves the containment model, while an ad hoc policy relaxation can create a long-lived exception that is hard to justify later. If the service truly needs nonstandard access, document the reason and make the change explicit rather than masking the problem.

What this means in practice

For most teams, installing and configuring SELinux on CentOS 8 means treating security policy as part of deployment readiness, not as an afterthought. The first operational rule is to start with the default targeted policy unless you have a documented reason not to. The second rule is to validate the application in permissive mode when you are onboarding a new service or restoring one from backup. The third rule is to fix labels and policy assumptions before you accept the system into production.

In practice, this changes how you troubleshoot. A service failure is not automatically a Unix permissions issue, a systemd issue, or an application bug. SELinux has to be part of the differential diagnosis whenever access failures are involved. That is especially true for web content, database files, custom ports, and application data outside standard paths.

Decision guidance: enforcing, permissive, or disabled

The right mode depends on your objective.

Use enforcing when the service is already validated and you want SELinux to actively prevent unauthorized access. This is the normal production target for a hardened host.

Use permissive when you are testing a new deployment, migrating data, or diagnosing denials and need visibility without immediate blockage. Permissive is a transition state, not a steady-state security posture.

Use disabled only when there is a documented operational reason and you fully understand the security trade-off. On a general-purpose CentOS 8 server, disabled should be the exception, not the baseline.

A useful decision rule is simple: if the host is meant to carry production traffic, keep SELinux enabled; if the workload is not yet validated, gather denials in permissive mode; if someone proposes disabling SELinux, ask what specific business constraint requires removing the control entirely and whether the same outcome can be achieved with labeling or policy adjustments.

Common mistakes to avoid

The most common mistakes are predictable and avoidable.

One mistake is assuming SELinux is working because the package is installed. Installation alone does not prove enforcement, and it does not prove correct labeling.

Another mistake is changing /etc/selinux/config and assuming the running state changed immediately. Boot-time configuration and current runtime mode are related but not identical.

A third mistake is treating denials as noise and disabling SELinux instead of inspecting the event. Denials are often the most direct evidence of a mislabeled file, an unexpected port, or a service path that was never accounted for during deployment.

A fourth mistake is relabeling only part of the filesystem. If a service depends on a tree of files, both the directory and the nested content may need to be corrected.

A fifth mistake is documenting exceptions informally. If a workload truly needs unusual access, record the rationale, the affected path or port, and the review date so the exception does not become permanent by accident.

Production readiness checklist

Before you call a CentOS 8 host ready for production with SELinux enabled, verify the following:

- sestatus shows SELinux is enabled.
- getenforce matches the intended runtime mode.
- /etc/selinux/config matches the desired boot-time state.
- Required policy packages are installed.
- Application data, logs, and content paths have the correct labels.
- Any custom ports or nonstandard paths have been reviewed for SELinux impact.
- Recent AVC denials have been examined and explained.
- No temporary permissive setting remains in place without an owner and deadline.

If any of those checks fail, treat the server as not yet ready for full production trust.

Final takeaway

To install and configure SELinux on CentOS 8 correctly is to treat it as an operational control, not just a package or a config file. Confirm the policy is present, decide on the correct mode, validate the runtime state, and use denials as evidence to fix labels or policy gaps before production. That approach preserves security without turning SELinux into a source of avoidable outages.

Use this guidance together with Ubuntu AppArmor profiling and RHEL vulnerability scanning to connect the workflow with related operational context already available on the site.