HOW-TO GUIDE

How to Configure SELinux Booleans on Red Hat Linux

SELinux booleans let you change specific policy behavior without disabling SELinux or writing custom modules. This guide shows how to find the right boolean, change it safely, verify the effect, and roll back if needed.

Operating Systems - July 31, 2026

Quick version

When an application is blocked by SELinux but the service itself is otherwise healthy, a boolean is often the safest first change. SELinux booleans let you toggle selected policy behavior without disabling SELinux or building a custom policy module.

The practical workflow is:

- Identify the denied action and confirm it is a boolean-related case.
- List candidate booleans.
- Change the boolean temporarily and test.
- Make the change persistent only if the result is correct and operationally acceptable.
- Revert the change if it creates a broader access path than you intended.

If you are still reading AVC denials, start by correlating them with the service context and labeling first. For structured triage, How to Audit Red Hat SELinux Policy Violations on RHEL is the right companion article.

What SELinux booleans are and when to use them

A SELinux boolean is a policy switch that enables or disables a predefined behavior in SELinux policy. In operational terms, it gives you a controlled way to loosen or tighten a specific policy decision without turning SELinux off.

Use a boolean when:

- The service is failing because policy is intentionally blocking a known and acceptable behavior.
- The denial is a common, documented exception rather than a sign of incorrect labeling or application design.
- You want a narrowly scoped policy adjustment instead of a custom module.

Do not reach for a boolean first when the root cause is likely:

- Incorrect file labels or directory contexts.
- A service running in the wrong domain.
- A port or protocol exposure issue that belongs in firewall or service configuration rather than SELinux policy.

That distinction matters because SELinux and firewall rules solve different problems. If access is blocked at the network layer, adjust ports and filtering; if access is blocked by policy, adjust SELinux. For a broader hardening context, see Hardening Red Hat Linux with SELinux and Firewall Rules.

Prerequisites

Before you change any boolean, confirm the following:

- You have root access or equivalent privilege.
- SELinux is enabled and enforcing or permissive on the host.
- You know which service or path is being affected.
- You have confirmed the denial is plausibly policy-related, not a misconfiguration in the application itself.

Useful commands for the initial check:

Expected output is usually Enforcing, Permissive, or Disabled for getenforce, and a more detailed policy status from sestatus.

Find the boolean you need

Start by listing booleans that are relevant to your service or subsystem. The getsebool command shows current boolean state.

A typical result looks like this:

The exact boolean names depend on what policy package is installed and what service is involved. Use names tied to the application you are troubleshooting, not generic guesses.

If you already know the boolean name, query it directly:

Expected output:

If the boolean you need does not appear, that is a useful signal. It may mean the denial is not a boolean problem, or the relevant policy package is not installed.

Change a boolean temporarily first

The safest operational pattern is to test the change in memory first. Use setsebool without persistence to change behavior until the next reboot.

Then verify the state:

Expected output:

This temporary approach is useful because it limits blast radius. If the change is too broad, you can revert it immediately without leaving a persistent policy exception behind.

Make the change persistent only after validation

If the temporary change solves the problem and you have confirmed it is acceptable, make it persistent with -P.

Important operational detail: -P writes the change to policy configuration so it survives reboot. That is the version you use only after validation, not as the first move.

Verify the persisted state:

You should still see --> on after the change.

Validate the effect in the real service path

A boolean is only useful if the application now works as intended and the access granted is no broader than you expected.

Validation should include both service behavior and SELinux evidence:

- Retry the exact action that was failing.
- Confirm the application can complete the intended task.
- Check for new denials after the test.

If you are validating web or network-facing services, it is often useful to test the application path directly and then review SELinux logs for any remaining AVC messages. Denials that continue after a boolean change can indicate a separate policy constraint or a different service context.

A quick log check may look like this:

or, on systems without ausearch in your workflow, review the relevant audit logs with your standard logging tools.

If the service still fails, do not keep adding booleans blindly. Re-evaluate whether you are dealing with labeling, port access, or an unrelated application error.

Common operational patterns

Some booleans are frequently used because they gate common service behaviors. The exact names and effects vary by policy and package, but the workflow stays the same: inspect, test temporarily, validate, then persist if needed.

Examples of typical patterns include:

- Allowing a web server to make outbound network connections.
- Allowing a service to read files from a user home directory.

- Allowing a file-sharing service broader export behavior.
- Allowing specific named services to interact with local resources that are normally restricted.

Before enabling any of these, ask two practical questions:

- Is this access required for the application to function?
- Does the access create an unacceptable security expansion?

If the answer to the second question is yes, a boolean may be the wrong fix even if it makes the service work.

Roll back or disable a boolean

If validation fails or the change is no longer needed, revert it immediately.

To disable a temporary or persistent boolean:

To make the rollback persistent:

Then confirm the value:

Expected output:

If you changed the persistent state and want to verify what is configured for boot, recheck after a restart as part of maintenance validation.

Check what changed before production use

Before promoting a boolean change to production, verify these points:

- The denial really matched a boolean-controlled behavior.
- The temporary change solved the exact problem.
- No additional AVC denials appeared during testing.
- The access granted is the minimum needed.
- The rollback command is documented and available to operators.

This is especially important on systems that also depend on strict network exposure control. SELinux can permit behavior that firewall policy still blocks, and vice versa. If you are hardening SSH exposure as part of the same change set, coordinate policy with service-level controls using How to Harden Red Hat Enterprise Linux SSH Access.

Practical decision rule

Use this rule of thumb when deciding whether to configure a boolean:

- If the denial is specific, documented, and expected, a boolean is reasonable.
- If the denial comes from bad labels, wrong paths, or a misplaced service context, fix the underlying configuration instead.
- If you cannot explain exactly why the access is needed, do not persist the boolean.

That keeps SELinux working as a guardrail rather than treating booleans as a shortcut around policy.

Example workflow you can reuse

Here is a concise workflow for a common service issue:

Use that pattern as an operational checklist, not as a substitute for diagnosis. The key is to change only what you can justify and verify.

Final take

Configuring SELinux booleans on Red Hat Linux is a controlled way to resolve specific policy blocks without disabling SELinux. The safe workflow is to identify the denial, test a temporary change, validate the real service path, and persist the boolean only when the access is necessary and acceptable. If the change feels vague or the logs point to labeling instead of policy, stop and fix the underlying issue first.

Use this guidance together with Red Hat vulnerability patching and CentOS SELinux troubleshooting to connect the workflow with related operational context already available on the site.