TUTORIAL

How to Configure SELinux Booleans in Red Hat Enterprise Linux

SELinux booleans let you adjust specific policy behaviors without writing custom policy modules. This tutorial shows how to identify the right boolean, change it safely, persist the setting, validate the result, and avoid common mistakes in Red Hat Enterprise Linux.

Operating Systems - July 14, 2026

Why SELinux booleans matter

SELinux booleans are the safest first place to adjust policy behavior when a service needs a limited exception. In Red Hat Enterprise Linux, they let you enable or disable specific policy paths without disabling SELinux, loading custom modules, or weakening the entire host.

After reading this tutorial, you will be able to identify the boolean you need, decide whether it is the right control to change, apply it temporarily or persistently, verify the resulting state, and confirm the service still behaves as expected before production use.

Before you change anything

Goal

Confirm that a boolean is the right fix and that you are not masking a deeper access-control problem.

Action

Review the denial or operational symptom first. If you are troubleshooting a service, check whether the issue matches a known SELinux policy restriction rather than a file permission, port binding, or service configuration problem.

Useful commands:

If SELinux is disabled, booleans are not the right tool. If it is enforcing or permissive, a boolean may be appropriate when the required behavior is already represented in policy but currently turned off.

If you are hardening a host rather than fixing a denial, read How to Harden Red Hat Systems with SELinux Policies first so you can decide whether a boolean change aligns with least-privilege requirements.

Expected output

You know the current SELinux mode and have a clear reason to change a boolean instead of weakening the policy more broadly.

Validation

Confirm the service or task still fails or is blocked with SELinux active before changing policy. If possible, inspect audit logs for AVC denials related to the service.

Common failure

Applying a boolean because a service is failing for unrelated reasons. That creates unnecessary policy changes and makes later troubleshooting harder.

Stop here if

- SELinux is disabled and you plan to rely on booleans as a workaround.
- You cannot explain the behavior change you are trying to achieve.
- The issue is actually a missing package, incorrect context, wrong port, or application bug.

Identify the boolean you need

Goal

Find the specific policy switch that controls the behavior you want.

Action

List booleans and inspect the available descriptions. The names are often service-specific, so read the description rather than guessing from the name alone.

To narrow the list, search for a keyword that matches the service or behavior, such as httpd, ftp, nis, tftp, or domain.

If you need detail on a specific boolean, use semanage boolean -l when available:

Expected output

You should have the exact boolean name and a brief understanding of what it controls.

Validation

Read the boolean description and confirm that turning it on or off matches the desired behavior. For example, a boolean that permits one directory or protocol extension is not a general bypass for the service.

Common failure

Using a boolean because the name looks familiar without checking the description. SELinux booleans are precise; the wrong one may do nothing useful or open an unintended capability.

Decide whether the change should be temporary or persistent

Goal

Choose the right scope for the change before you apply it.

Action

Use a temporary change if you are testing whether the boolean resolves the problem. Use a persistent change only after you confirm the setting is correct and safe.

A temporary change applies until the next reboot:

A persistent change survives reboot:

This distinction matters operationally. Temporary changes are ideal for validation and incident response. Persistent changes are what you want after change control review and verification.

Expected output

The boolean changes state immediately if the policy supports it, and the service can be retested without restarting the host.

Validation

Check the current value after the change:

A typical output looks like this:

Common failure

Using -P before validating the behavior. If the boolean is the wrong one, you have now written an unnecessary persistent change into policy configuration.

Apply the boolean safely

Goal

Enable or disable the targeted SELinux behavior with minimal operational risk.

Action

Set the boolean only for the service and behavior you identified. A few practical examples are below.

Enable Apache HTTPD to make outbound network connections:

Allow Apache to read home directories when the policy requires it:

Allow FTP access to home directories when needed by the service design:

If you are unsure whether the setting should be on or off, test with a temporary change first and record the observed behavior before making it persistent.

Expected output

The command completes without errors, and the boolean value reflects the requested state.

Validation

Immediately recheck the setting and, if applicable, retest the application path that previously failed.

Common failure

- Misspelling the boolean name.
- Changing a boolean that exists but is unrelated to the service path you are testing.
- Forgetting that a persistent change may need a service restart for the application to pick up new access patterns, even though the SELinux setting itself changed immediately.

Verify the effect in a real workflow

Goal

Confirm the boolean actually resolves the access issue and does not create a broader problem.

Action

Reproduce the original operation after the change. For a web service, that might mean reloading a page, connecting to a backend, or reading the affected path. For a daemon, it might mean repeating the action that triggered the denial.

Check logs for new AVC denials during the test window:

If the denial is gone and the application works, the boolean likely addressed the specific policy restriction.

If you need to compare with current SELinux behavior more broadly, the troubleshooting workflow in How to Harden Red Hat Systems with SELinux Policies can help you distinguish a boolean issue from a policy-context or labeling issue.

Expected output

The original access problem disappears, and no new AVC denials appear for the same operation.

Validation

Look for three things together:

- The boolean shows the expected on/off state.
- The application or service action succeeds.
- The audit log no longer records the same denial.

Common failure

The service still fails because the root cause is file labeling, port labeling, or a different policy restriction. In that case, changing more booleans is usually the wrong next move.

Confirm persistence and document the change

Goal

Make sure the setting will survive reboot and that the operational record is complete.

Action

If you used setsebool -P, confirm that the boolean remains set after a reboot or at least after a policy reload cycle in your maintenance window.

Review the current persistent state with semanage boolean -l and verify that your intended value is reflected in the installed configuration.

Document the reason for the change, the system or service it affects, the exact boolean, the expected outcome, and the rollback command.

Expected output

The boolean remains set across restarts, and you have an operational record for future maintenance or audits.

Validation

After reboot or maintenance restart, confirm the boolean again:

Then repeat the service test to ensure the behavior still matches expectations.

Common failure

Assuming a change is persistent because the service worked once. Without verification after reboot, you may not discover a reverted setting until the next outage.

Roll back or tighten the setting when needed

Goal

Return to a safer state if the boolean is no longer justified or if the test failed.

Action

Disable the boolean using the same command pattern you used to enable it:

If you used a temporary change only, a reboot may also clear it, but do not rely on reboot alone as a rollback plan. Always confirm the final state explicitly.

Expected output

The boolean returns to the off state and the service follows the stricter policy again.

Validation

Recheck the boolean and confirm the application no longer relies on the relaxed behavior.

Common failure

Leaving a permissive boolean enabled after the original issue has been resolved elsewhere. Over time, that creates policy drift and weakens the purpose of SELinux enforcement.

Operational checklist for production use

Before you promote a boolean change into production, confirm the following:

- You identified the exact SELinux-controlled behavior, not just a generic service failure.
- The boolean description matches the required action.
- You tested the change temporarily before making it persistent.
- The service works after the change and the same AVC denials no longer appear.
- The persistent state survives reboot or is otherwise controlled through configuration management.
- The change has an owner, reason, and rollback plan.

A boolean is usually the right tool when you need a narrow, documented policy exception. It is not a substitute for fixing labeling, service design, or application behavior. If you treat booleans as precise controls instead of broad workarounds, you can keep SELinux enforcing while still supporting real operational needs.

Use this guidance together with BitLocker TPM and PIN protection to connect the workflow with related operational context already available on the site.

Use this guidance together with harden Ubuntu with AppArmor and UFW and Windows 10 Local Security Policy to connect the workflow with related operational context already available on the site.