CHECKLIST

SELinux Policy Controls Hardening Checklist for Red Hat Enterprise Linux

A practical checklist for validating SELinux policy controls, reviewing denials, and preparing Red Hat Enterprise Linux services for production with minimal risk and clear change control.

SELinux Policy Controls Hardening Checklist

Use this checklist to validate, adjust, and operationalize SELinux policy controls on Red Hat Enterprise Linux before production rollout.

1) Confirm the baseline

- [ ] Verify SELinux is enabled on the host.
- [ ] Verify the system is in Enforcing mode for production use.
- [ ] Confirm the target service is installed, updated, and configured as intended.
- [ ] Document the service owner and the system owner.
- [ ] Record the host purpose, exposed ports, and data handled by the service.

2) Review the service design

- [ ] Identify which files, directories, ports, and network destinations the service truly needs.
- [ ] Separate required access from convenient but nonessential access.

- [ ] Check whether the service can use standard paths and default ports.
- [ ] Identify any custom directories under /opt, /srv, or other nonstandard locations.
- [ ] Identify any custom listening ports or unusual inter-service connections.

3) Validate SELinux behavior under realistic load

- [ ] Start or restart the service in a representative environment.
- [ ] Exercise the normal application workflow, not just a startup test.
- [ ] Confirm expected reads, writes, and network connections succeed.
- [ ] Watch for unexplained failures that resemble permission problems or application bugs.
- [ ] Capture any blocked operations for review.

4) Investigate denials before changing policy

- [ ] Determine the exact process type involved.
- [ ] Identify the blocked object type, file path, port, or destination.
- [ ] Confirm whether the blocked action is part of intended service behavior.
- [ ] Distinguish a true policy gap from an application design problem.
- [ ] Re-test after each change rather than stacking multiple changes at once.

5) Prefer the least risky fix

Choose the narrowest fix that meets the service requirement:

- [ ] Relabel files or directories so they match the expected SELinux type.
- [ ] Assign the correct port label for a nondefault listening port.
- [ ] Use a documented SELinux boolean only when it is the supported way to enable a known integration.

- [ ] Create a narrowly scoped custom policy module only when the access is required and well understood.
- [ ] Avoid broad permissive exceptions or disabling SELinux for the service domain.

6) Avoid common anti-patterns

- [ ] Do not broaden file permissions just to bypass SELinux restrictions.
- [ ] Do not place service data in arbitrary shared directories without a labeling plan.
- [ ] Do not rely on permissive mode as proof that production enforcement will work.
- [ ] Do not allow the service to use arbitrary ports to avoid cleanup work.
- [ ] Do not keep temporary policy exceptions without a review date.

7) Check change impact before production

- [ ] Confirm the service still works with SELinux enforcing.
- [ ] Verify that no unexpected access was granted by the change.
- [ ] Confirm the service cannot read or write data outside its intended scope.
- [ ] Validate all required paths, ports, and integration points.
- [ ] Re-test after package updates, configuration changes, or deployment changes.

8) Operationalize review and ownership

- [ ] Document every SELinux change with a clear justification.
- [ ] Record the owner responsible for each exception or policy module.
- [ ] Store notes on what was changed, why it was changed, and when it should be reviewed.
- [ ] Treat SELinux policy changes like firewall or privilege changes.
- [ ] Include SELinux checks in standard deployment and incident runbooks.

9) Production readiness checklist

Before calling the service production-ready, confirm all of the following:

- [ ] The service runs successfully with SELinux enforcing.
- [ ] All required access is explicitly understood and justified.
- [ ] File labels are correct for custom content and data paths.
- [ ] Port labels are correct for nondefault listening ports.
- [ ] Any booleans used are reviewed and documented.
- [ ] Any custom policy modules are minimal, tested, and owned.
- [ ] No unnecessary permissive settings remain in place.
- [ ] The team knows how to interpret and respond to future denials.

10) Final go-live review

- [ ] Confirm the deployment matches the tested configuration.
- [ ] Confirm the service account is not being used to bypass policy design.
- [ ] Confirm the host is monitored for denial patterns and service failures.
- [ ] Confirm rollback steps are available if a policy adjustment causes disruption.
- [ ] Approve production use only after the above checks pass.

Quick decision guide

Use a narrow SELinux change when the access is clearly required, specific, and explainable.

Avoid a policy change when the service:

- [ ] Needs broad access to many unrelated files
- [ ] Uses arbitrary ports as a workaround

- [ ] Depends on shared directories without a labeling strategy
- [ ] Requires the policy to be weakened just to function

If the request cannot be described in one sentence using a specific file type, path, or port, the service design likely needs adjustment before production.