CHECKLIST

# Red Hat Linux SELinux and Firewall Hardening Checklist

A practical checklist for hardening Red Hat Linux systems by aligning SELinux enforcement with firewall rules, verifying the correct control layer, and validating changes before production use.

## Red Hat Linux SELinux and Firewall Hardening Checklist

Use this checklist to harden a Red Hat Linux system without over-opening the network or weakening SELinux containment. It is designed to help you decide whether a problem belongs to firewall exposure, SELinux policy, file labeling, or application design.

---

## 1) Confirm the service boundary

- [ ] Identify the service, daemon, or application being hardened.
- [ ] Confirm whether the service should be reachable from the network at all.
- [ ] Define the business or operational reason for any external access.
- [ ] Identify the exact client networks that should be allowed.
- [ ] Record whether the service should listen on localhost, a single interface, or multiple interfaces.

If the service should not be reachable externally:

- [ ] Do not create a permanent firewall exception.
- [ ] Use localhost testing, a temporary lab subnet, or a controlled jump path instead.

---

## 2) Verify the network exposure first

- [ ] Confirm the process is listening on the expected port.
- [ ] Confirm it is bound to the intended interface.
- [ ] Check whether the port number is standard or custom.
- [ ] Verify the firewall zone assigned to the active interface.
- [ ] Review active firewall rules for the service port, protocol, and source restrictions.
- [ ] Confirm the rule is scoped to the smallest necessary source network.
- [ ] Avoid broad allow rules such as any source, any zone, or all interfaces unless explicitly justified.

Ask:

- [ ] Can the client reach the host from the network?
- [ ] If not, is the block at the firewall, routing, or interface level?

---

## 3) Verify SELinux status and role

- [ ] Confirm SELinux is enabled and enforcing.
- [ ] Check that SELinux is not disabled for convenience during deployment.
- [ ] Identify the SELinux context of the process.
- [ ] Identify the SELinux context of any files, directories, or sockets the service uses.
- [ ] Check whether the application uses a standard path or custom path.
- [ ] Check whether the application uses a standard port or custom port.
- [ ] Determine whether the needed access is already supported by an existing SELinux boolean or label.

Remember:

- [ ] SELinux controls what a process can do after it is already running.

- [ ] SELinux does not replace firewall rules.
- [ ] A reachable service can still be blocked locally by SELinux.

---

## 4) Inspect denials before changing policy

- [ ] Review recent SELinux denial events.
- [ ] Confirm the denial matches the observed failure.
- [ ] Determine whether the issue is a mislabeled file, wrong path, wrong context, or actual policy gap.
- [ ] Verify that the application is running under the expected domain.
- [ ] Check whether the service is trying to access a resource it should not use.
- [ ] Do not change policy until the denial evidence is understood.

Use evidence to classify the issue:

- [ ] Network exposure issue: external client cannot connect.
- [ ] SELinux policy issue: process is denied a local action.
- [ ] Labeling issue: file or directory context is incorrect.
- [ ] Application issue: the software is using an unexpected path, port, or privilege.

---

## 5) Make the smallest safe change

- [ ] If the issue is network reachability, adjust the firewall first.
- [ ] If the issue is local access or binding behavior, adjust SELinux labels, booleans, or service context first.
- [ ] Change only one control layer at a time.
- [ ] Prefer a narrow exception over a broad exception.
- [ ] Prefer correcting labels or service configuration over disabling SELinux.

- [ ] Prefer a precise firewall rule over opening a port to all networks.

Do not:

- [ ] Disable SELinux to solve a deployment problem.
- [ ] Open a port broadly ?for testing? and leave it in production.
- [ ] Change firewall and SELinux settings in the same edit unless there is a documented, temporary emergency plan.

---

## 6) Validate common hardening scenarios

### If the service should be network-restricted

- [ ] Confirm only the approved subnet can reach the service.
- [ ] Confirm the service is not exposed on unintended interfaces.
- [ ] Confirm the firewall blocks the port from other networks.

### If the service needs a custom port

- [ ] Verify whether SELinux already allows that port for the service type.
- [ ] Verify the firewall allows only the intended source networks.
- [ ] Confirm the custom port is documented for operations and recovery.

### If the service writes to a custom directory

- [ ] Verify the directory label matches the service expectation.
- [ ] Verify the process domain is allowed to write there.
- [ ] Confirm the path is necessary and not just a workaround.

## If the service accesses unusual files or sockets

- [ ] Confirm the access is required for normal operation.
- [ ] Confirm the access is permitted by policy or label.
- [ ] Confirm the change does not expose unrelated files or capabilities.

---

## 7) Re-test after every change

- [ ] Re-test from the client side.
- [ ] Re-test on the host.
- [ ] Confirm the service is reachable only where intended.
- [ ] Confirm the service can perform only the required local actions.
- [ ] Verify there are no new denial events after the change.
- [ ] Verify the configuration survives service restart and system reboot if applicable.

---

## 8) Document the decision

- [ ] Record the service name and owner.
- [ ] Record the port, protocol, and source network allowed.
- [ ] Record the SELinux context, label, or boolean change made.
- [ ] Record why the change was necessary.
- [ ] Record why broader access was not used.
- [ ] Record any temporary exceptions and their expiration date.
- [ ] Record how the change was validated.

---

## 9) Quick decision guide

Use the firewall when:

- [ ] The problem is who can connect to the service.
- [ ] The service should not be reachable from certain networks.
- [ ] The issue is external exposure.

Use SELinux when:

- [ ] The service can start but cannot read, write, bind, or transition as expected.
- [ ] The issue is local process behavior.
- [ ] The access denial involves labels, contexts, or policy.

Use both layers together when:

- [ ] The service must be reachable only from approved sources and must remain tightly confined after connection.
- [ ] The service needs a precise network exception and a precise local permission.

---

## 10) Common mistakes to avoid

- [ ] Disabling SELinux instead of fixing the underlying issue.
- [ ] Allowing overly broad firewall access.
- [ ] Assuming one control can compensate for the other.
- [ ] Treating every denial as a reason to loosen policy.
- [ ] Forgetting to verify the client-side view after changes.
- [ ] Leaving temporary testing rules in production.

---

## Final hardening check

Before production use, confirm all of the following:

- [ ] Only required ports are open.
- [ ] Firewall rules are restricted to the intended source networks.
- [ ] SELinux is enabled and enforcing.
- [ ] File labels and service contexts are correct.
- [ ] No unnecessary policy exceptions were introduced.
- [ ] The service was tested from both the client and the host.
- [ ] The change is documented for future operations and audits.

---

## One-line rule of thumb

Firewall rules answer ?who can reach it,? while SELinux answers ?what can it do once it is reached.?

Use the narrowest safe change that matches the actual control layer.