Security and sandboxing

Sandbox vs Tool Policy vs Elevated

OpenClaw has three related (but different) controls:

1. **Sandbox** (`agents.defaults.sandbox.*` / `agents.list[].sandbox.*`) decides **where tools run** (Docker vs host).
2. **Tool policy** (`tools.*` , `tools.sandbox.tools.*` , `agents.list[].tools.*`) decides **which tools are available/allowed**.
3. **Elevated** (`tools.elevated.*` , `agents.list[].tools.elevated.*`) is an **exec-only escape hatch** to run on the host when you're sandboxed.

Quick debug

Use the inspector to see what OpenClaw is *actually* doing:

```
openclaw sandbox explain
openclaw sandbox explain --session agent:main:main
openclaw sandbox explain --agent work
openclaw sandbox explain --json
```

It prints:

- effective sandbox mode/scope/workspace access
- whether the session is currently sandboxed (main vs non-main)
- effective sandbox tool allow/deny (and whether it came from agent/global/default)

elevated gates and fix-it key paths

Sandbox: where tools run

Sandboxing is controlled by `agents.defaults.sandbox.mode` :

- `"off"` : everything runs on the host.
- `"non-main"` : only non-main sessions are sandboxed (common “surprise” for groups/channels).
- `"all"` : everything is sandboxed.

See Sandboxing for the full matrix (scope, workspace mounts, images).

Bind mounts (security quick check)

- `docker.binds` *pierces* the sandbox filesystem: whatever you mount is visible inside the container with the mode you set (`:ro` or `:rw`).
- Default is read-write if you omit the mode; prefer `:ro` for source/secrets.
- `scope: "shared"` ignores per-agent binds (only global binds apply).
- Binding `/var/run/docker.sock` effectively hands host control to the sandbox; only do this intentionally.
- Workspace access (`workspaceAccess: "ro"` / `"rw"`) is independent of bind modes.

Tool policy: which tools exist/are callable

Two layers matter:

- **Tool profile**: `tools.profile` and `agents.list[].tools.profile` (base allowlist)

- **Provider tool profile**: `tools.byProvider[provider].profile` and `agents.list[].tools.byProvider[provider].profile`
- **Global/per-agent tool policy**: `tools.allow` / `tools.deny` and `agents.list[].tools.allow` / `agents.list[].tools.deny`
- **Provider tool policy**: `tools.byProvider[provider].allow/deny` and `agents.list[].tools.byProvider[provider].allow/deny`
- **Sandbox tool policy** (only applies when sandboxed): `tools.sandbox.tools.allow` / `tools.sandbox.tools.deny` and `agents.list[].tools.sandbox.tools.*`

Rules of thumb:

- `deny` always wins.
- If `allow` is non-empty, everything else is treated as blocked.
- Tool policy is the hard stop: `/exec` cannot override a denied `exec` tool.
- `/exec` only changes session defaults for authorized senders; it does not grant tool access. Provider tool keys accept either `provider` (e.g. `google-antigravity`) or `provider/model` (e.g. `openai/gpt-5.2`).

Tool groups (shorthands)

Tool policies (global, agent, sandbox) support `group:*` entries that expand to multiple tools:

```
{
  tools: {
    sandbox: {
      tools: {
        allow: ["group:runtime", "group:fs", "group:sessions", "group:memory"],
      },
    },
  },
}
```

Available groups:

- `group:runtime` : `exec` , `bash` , `process`
- `group:fs` : `read` , `write` , `edit` , `apply_patch`
- `group:sessions` : `sessions_list` , `sessions_history` , `sessions_send` , `sessions_spawn` , `session_status`
- `group:memory` : `memory_search` , `memory_get`
- `group:ui` : `browser` , `canvas`
- `group:automation` : `cron` , `gateway`
- `group:messaging` : `message`
- `group:nodes` : `nodes`
- `group:openclaw` : all built-in OpenClaw tools (excludes provider plugins)

Elevated: exec-only “run on host”

Elevated does **not** grant extra tools; it only affects `exec` .

- If you’re sandboxed, `/elevated on` (or `exec` with `elevated: true`) runs on the host (approvals may still apply).
- Use `/elevated full` to skip exec approvals for the session.

If you’re already running direct, elevated is effectively a no-op (still gated).

Elevated is **not** skill-scoped and does **not** override tool allow/deny.

`/exec` is separate from elevated. It only adjusts per-session exec defaults for authorized senders.

Gates:

- Enablement: `tools.elevated.enabled` (and optionally `agents.list[].tools.elevated.enabled`)
- Sender allowlists: `tools.elevated.allowFrom.<provider>` (and optionally `agents.list[].tools.elevated.allowFrom.<provider>`)

See **Elevated Mode**.

Common “sandbox jail” fixes

“Tool X blocked by sandbox tool policy”

Fix-it keys (pick one):

- Disable sandbox: `agents.defaults.sandbox.mode=off` (or per-agent `agents.list[].sandbox.mode=off`)
- Allow the tool inside sandbox:
  - remove it from `tools.sandbox.tools.deny` (or per-agent `agents.list[].tools.sandbox.tools.deny`)
  - or add it to `tools.sandbox.tools.allow` (or per-agent allow)

“I thought this was main, why is it sandboxed?”

In `"non-main"` mode, group/channel keys are *not* main. Use the main session key (shown by `sandbox explain`) or switch mode to `"off"` .

‹ Sandboxing Gateway Protocol ›