Homericly — Access Control Policy

Live online: https://homericly.com/access-control — always the canonical version.

Owner: Founder / Engineering Lead — roofing@homericly.com **Monitored security channel:** security@homericly.com (group address, always watched) **Last reviewed:** 2026-06-05 **Review cadence:** Quarterly. Access reviews recorded in the audit log.

1. Purpose

Define who can access what across Homericly systems, how those grants are issued, reviewed, and revoked.

2. Roles

Application-layer (in-app)

Defined in `lib/auth/roles.ts`. Two surfaces:

- **Staff surface** (Homericly employees / operators of a fork):

`admin`, `operator`, `designer`, `editor`, `engineer`, `auditor` (read-only), plus Content Ops sub-roles (`content_strategist | writer | editor | designer | publisher`).

- **Portal surface** (contractor's own team using their fork):

`owner`, `manager`, `staff`, `read_only`.

Every server action calls `requireStaff(<capability>)` or `requireOps(<capability>)`. Capabilities are coarse, action-named, namespaced (`staff.manage_users`, `staff.issue_refund`, `staff.override_client_setting`, etc.). No action checks role names directly — only capabilities.

Tenant isolation (cross-org)

Postgres Row-Level Security on every per-org table, keyed on `current_setting('app.current_org')`. A request that hasn't established the org context (`setOrgContext(orgId)`) cannot read rows belonging to that org. Cross-tenant reads are not architecturally possible from a request-scoped DB session.

Infrastructure-layer (admin tools)

The following services are gated by MFA and a tight allowlist of human operators:

Service	Who has access	MFA required
Vercel (`roofing@homericly.com` team)	Founder + named delegate	Yes
Supabase	Founder + named delegate	Yes
GitHub (`Homericly-Roofing` org)	Founder + named delegate	Yes

Service	Who has access	MFA required
AWS (SES, IAM)	Founder only	Yes
Stripe (platform billing)	Founder only	Yes
Plaid	Founder only	Yes
Telnyx	Founder only	Yes
Cloudflare / DNS	Founder only	Yes
Hetzner (render box)	Founder only	Yes

3. Authentication

- **Staff** authenticate via Google SSO (Better Auth). Email/password

is disabled. Allowlist `STAFF_EMAILS` env var gates which Google identities can hold a staff role at all.

- **Portal users** authenticate via Google SSO or email magic-link

(NextAuth). Each portal user is bound to exactly one organization; changing org requires a separate invite.

- **Server-to-server** (broker ■ fork, scheduler ■ fork, Powerhouse ■

fork) uses HMAC-SHA256 over canonical envelopes plus a freshness window. Secrets rotate independently per channel.

4. MFA

- All admin/infrastructure accounts in section 2 above require MFA.

Where the vendor supports phishing-resistant MFA (passkeys, hardware keys), it is enabled.

- Portal users inherit MFA from their Google identity when signing in

via SSO. Magic-link sign-in is single-factor; sensitive portal surfaces (Connect Bank, Wallet Settings) gate behind a re-auth prompt.

5. Provisioning

- New staff: founder invites via Google SSO + grants role through

staff UI (`staff.manage_roles` capability).

- New portal users: invited by their org's `owner` via the team panel.
- New service / API key: founder issues from the vendor portal, stores

in Vercel env, never in code.

6. De-provisioning

- Staff departure → Google account revoked + Vercel/Supabase/GitHub

membership revoked the same day. The audit log records the event.

- Portal user removal → org owner revokes from team panel; session is

invalidated within 5 minutes.

- Vendor key rotation → key revoked at vendor portal + Vercel env

updated + production redeployed in a single change.

7. Periodic access review

- **Quarterly** review of:
- Active staff roles vs. who actually needs them
- Active portal-user accounts per org (samples)
- Vendor API keys + scopes
- Open Composio connected accounts per org
- Each review committed to `docs/access-reviews/YYYY-QN.md` and stamped

to the audit log.

8. Audit

- Every privileged action passes through `record()` in `lib/audit/index.ts`

and lands in the immutable `audit_log` table.

- Audit log is retained indefinitely. See Data Retention & Deletion

Policy for the regulatory rationale.

9. Enforcement

Violations (sharing credentials, disabling MFA, leaking keys) are grounds for revocation of all access. Suspected compromises follow the Incident Response section of the Information Security Policy.