Make trust boundaries visible.

Identify where a request changes identity, authority or data ownership as it moves through a system.

Boundary review map

Boundary	Question to resolve
User to application	How is the person authenticated and authorised?
Application to data	Which records may this request access?
Service to service	What authority does the service identity receive?
Model to tool	Which proposed actions may the application execute?

Apply controls where authority changes

A prompt, interface or network location alone does not establish permission. Enforce the decision at the component that owns the protected action or record.

Test the negative path

Try an expired identity, a different organisation and a revoked permission. Confirm that a cached or indirect path cannot retain access.

Locate the boundaries in a real request

A trust boundary exists where an operation moves between different identities, permissions or control arrangements. Follow a representative request through the browser, application, data stores and external providers. Identify which component authenticates the caller and which component authorises the specific action.

Do not assume that access to one system grants authority in another. A user allowed to view an order may not be allowed to change its payment details. A background integration may operate with a service identity whose scope needs separate control. Mark these distinctions in the architecture and keep them visible in interface contracts. The boundary should be enforced where the authoritative action occurs.

Treat external content as input

Documents, messages and retrieved passages can contain instructions that were not written by the application owner. An AI workflow should not treat those instructions as authority to reveal data or call a tool. Keep tool permissions and business constraints outside the prompt, and validate proposed actions against the current user and record state.

The same principle applies beyond AI. Validate uploaded files, API payloads and redirects at the relevant boundary. Decide which information can be passed to an external provider and what is recorded in logs. Use representative tests to check that restricted input cannot expand an operation's scope. A diagram becomes more useful when each boundary has a corresponding control and test.

Review privileges and change paths

Identify who can alter roles, credentials, prompts, tools and deployment configuration. Administrative access can change the effectiveness of controls even when the application itself is unchanged. Use appropriate separation of responsibilities and keep reviewable records of consequential changes.

Test permission removal as well as permission grant. Consider cached results, active sessions and queued work created before a role changed. Record the expected behaviour and any delay before removal takes effect. Revisit the model when new integrations or user groups are introduced. The aim is an explicit, testable account of authority across the workflow, rather than a broad statement that the application uses authentication or encryption.

Is an internal network automatically trusted?

No. Internal components still need explicit identities and scoped authority appropriate to their responsibilities.

Read the current resource online

https://cobnex-review.rgcsekaraa.workers.dev/trust-hub/