Srasta — SOC 2 Common Criteria Controls Matrix

Version: 1.0 · **Last updated:** 2026-04-29 · **Scope:** Srasta v1.x platform (control plane + audit + identity + workload routing)

This document is the **first-read artifact** for any security conversation with a prospective customer, design partner, or auditor. Each row maps a SOC 2 Common Criteria (CC) control to Srasta's current implementation, evidence pointers, and a status flag.

It is **honest** about what's shipped, what's planned, and what's out of scope at the seed stage. Over-claiming is not a control — it is a finding. Anything marked "planned post-Type-I" is roadmap, not roadmap-in-flight.

For the seed-stage compliance posture this matrix supports, see `docs/roadmap/compliance-roadmap.md` (anchor doc) and `docs/strategy/seed-momentum-plan.md` ticket #4. The vertical compliance frameworks (HIPAA, PCI DSS, FedRAMP) are tracked in `docs/compliance/controls-matrix.md` — they are post-seed concerns per the seed-momentum anchor's Mandate 5 (regulated-adjacent ICP, no compliance runway).

Status legend

Symbol	Meaning
✅	Shipped in production, verifiable today
🔄	Partially shipped — gap documented in the row
📅 Q3	On the seed-execution roadmap; target Q3 2026
📅 post-Type-I	Will land after the SOC 2 Type I audit (post-seed)
⊘ customer	Customer's responsibility — Srasta runs inside their perimeter

CC1 — Control Environment

The control environment is the foundation: integrity, ethical values, governance structure, accountability. At seed stage (solo-founder + small-org buyers) most of these controls are formally documented commitments rather than headcount-based processes.

CC	Control	Status	Implementation	Evidence
CC1.1	Integrity & ethical values	📅 post-Type-I	Code of Conduct + acceptable-use policy committed to repo at Type-I prep.	`docs/security/code-of-conduct.md` (planned)
CC1.2	Board independence + oversight	📅 post-Type-I	N/A pre-seed (no board of directors). Investor advisory function tracked once seed closes.	—
CC1.3	Reporting structures + authority	✅	Solo-founder org with clear escalation: customer issue → Prem (founder); legal → outside counsel. Codified in `docs/security/incident-response.md` once #157 ships.	`docs/security/incident-response.md` (with #157)
CC1.4	Competence	🔄	Hiring criteria + role expectations documented at the time of first hire. Pre-seed: solo-founder competency demonstrated by the platform itself + commit history.	git log; `docs/strategy/seed-momentum-plan.md`
CC1.5	Accountability	✅	Every change is committed to GitLab with the author, the issue reference, and the GitLab MR review trail. No hot-patch-without-source-commit policy (Mandate 3).	`git log`; the no-tactical-drift policy

CC2 — Communication and Information

CC	Control	Status	Implementation	Evidence
CC2.1	Internal info quality	✅	All operational decisions are made on the basis of audited data: audit feed, control-plane inventory, security-panel findings. No verbal-only operational state.	Admin UI Activity tab; the shared audit-log module
CC2.2	Internal communication	✅	Strategy, roadmaps, security posture all in repo (`docs/strategy/`, `docs/roadmap/`, `docs/security/`). Single source of truth, no Slack-only decisions per Mandate 6.	`docs/strategy/seed-momentum-plan.md`; this matrix
CC2.3	External communication	🔄	Privacy policy + DPA template land via #158. Security questionnaire (CAIQ Lite) drafted via #158. Public landing pages at srasta.ai already communicate scope + posture. Operational telemetry to Gandiva (counts + shapes only, never customer data) documented in `docs/legal/privacy-policy.md` Section 3.4 + DPA Annex 4 (#173 / #174).	`docs/legal/privacy-policy.md`; `docs/security/caiq-lite.md`

CC3 — Risk Assessment

CC	Control	Status	Implementation	Evidence
CC3.1	Specifies objectives	✅	Seed-momentum plan + per-roadmap docs make objectives explicit at each level (anchor + sub-roadmap + ticket).	`docs/strategy/seed-momentum-plan.md`
CC3.2	Identifies + analyzes risk	📅 Q3	Formal risk register lands during Type-I prep. Pre-seed: tactical risk-tracking via the `feedback_*` memory files + active-tickets list.	`docs/security/risk-register.md` (planned Q3)
CC3.3	Considers fraud risk	📅 post-Type-I	Fraud-specific controls land once we have a finance + payment surface (post-seed).	—
CC3.4	Assesses change-related risk	✅	Every CI pipeline runs `validate:fresh-install`, `validate:no-hardcoded-ips`, `validate:ruff`, `validate:version` gates. Security-panel findings re-evaluated on every config change.	`.gitlab-ci.yml` validate stage; the admin security panel

CC4 — Monitoring Activities

CC	Control	Status	Implementation	Evidence
CC4.1	Ongoing/separate evaluations	✅	Continuous monitoring via the admin UI Activity tab (#147 canonical event taxonomy: auth / inference / tool / admin) + security-panel findings (the admin license-expiry check, etc.). Live-tail audit JSONL with filters + CSV export (#144–#147).	the audit-event canonical classifier (#147); `admin/static/index.html` Activity tab
CC4.2	Communicates deficiencies	🔄	Security findings surface in admin UI in real time. Formal escalation runbook lands with #157 (architecture PDF) — covers who is notified for which severity.	`docs/security/incident-response.md` (with #157); the admin security panel

CC5 — Control Activities

CC	Control	Status	Implementation	Evidence
CC5.1	Selects + develops control activities	✅	Operator-configurable policy profiles (`baseline` / `hipaa` / `pci` / `fedramp`) via `SRASTA_POLICY` env var. Stricter value wins on conflict. Policy enforcement runs as middleware in srasta-api gateway, rag-api, tool-gateway.	the policy engine config; `policy.yaml.example`
CC5.2	General controls over technology	✅	Signed Docker images (cosign keyless via Sigstore + GitLab OIDC), per-image SBOMs (Syft), CVE scans (Grype), digest-pinned images in customer compose.	`release/services.yaml` ; `.gitlab-ci.yml` sign + sbom + scan jobs
CC5.3	Deploys via policies + procedures	✅	Deployment via `scripts/install.sh` → `setup.sh` → `docker compose up -d` . Customer compose generated from `release/services.yaml` (single source of truth). Mandate 2: every policy is operator-configurable, never code-baked.	`scripts/install.sh` ; `setup.sh` ; `release/services.yaml`

CC6 — Logical and Physical Access Controls

This is the most code-relevant criterion for Srasta — most of these land directly on the control plane.

CC	Control	Status	Implementation	Evidence
CC6.1	Logical access security software	✅	OIDC authentication via Zitadel (v4.13.1). JWT signed + verified at the gateway (srasta-api). Forwarded principal headers (`X-Srasta-User-Id` , `X-Srasta-Tenant-Id` , `X-Srasta-Roles` , HMAC-signed) propagate identity to internal services.	the gateway authentication layer; the gateway forwarded-header builder; Zitadel pinned in `docker-compose.yml`
CC6.2	Identifies + authenticates users	✅	Zitadel issues JWTs after the OIDC dance. AUTH_MODE=oidc enforces verification on every gateway request. AUTH_MODE=none is a dev-only mode that emits a startup WARN.	the gateway OIDC auth provider
CC6.3	Authorizes + manages access changes	✅	Three layers: (1) Zitadel-managed roles per user; (2) per-role model whitelist (`platform_config.model_access` table, gateway-enforced via the gateway per-role model authorization, admin UI grid); (3) per-persona tool access (`setup/tool_policy.yaml` , tool-gateway-enforced).	the admin service model-access endpoints (#148–#151); the gateway per-role model authorization; `setup/tool_policy.yaml`
CC6.4	Restricts physical access	⊘ customer	Srasta runs inside the customer's infrastructure perimeter. Physical access to the host is the customer's responsibility.	Customer deployment plan
CC6.5	Manages access for joiners/movers/leavers	🔄	Admin UI exposes invite + revoke flows backed by Zitadel. Quarterly access reviews land post-Type-I (process gap, not feature gap).	the admin service invite/revoke endpoints; `docs/security/access-review-procedure.md` (post-Type-I)
CC6.6	External-use access controls	✅	srasta-api is the **single external entry point** (Mandate 1). Every external request transits this gateway: auth → authz → license → rate-limit → forwarded headers → upstream. No bypass routes.	the gateway request middleware; the audit-thesis design principle
CC6.7	Restricts data movement	✅	TLS termination at nginx (Let's Encrypt wildcard or operator BYOC); private Docker bridge network for inter-service traffic; audit-forwarder ships logs out only via operator-configured S3/SIEM endpoints.	`setup/nginx/` ; `docker-compose.yml:networks.srasta` ; the audit-forwarder daemon
CC6.8	Prevents + detects malicious software	✅	Image vulnerability scanning (Grype in CI, blocks merge on CRITICAL). SBOM published per image (Syft).	`.gitlab-ci.yml` grype + sign jobs; `scripts/verify-release-`

CC	Control	Status	Implementation	Evidence
			Cosign signature verification on every customer install (`scripts/verify-release-bundle.sh`). Tool-gateway enforces tool-execution policy with persona-scoped allowlists.	`bundle.sh` ; the tool-gateway policy enforcement

CC7 — System Operations

CC	Control	Status	Implementation	Evidence
CC7.1	Detects + monitors vulnerabilities	✅	Grype scans every image in CI. License validation on the audit-feed UI (the admin license-expiry check). Per-release SBOMs. License posture monitoring built into the gateway (60s revalidator) + admin (#169).	`.gitlab-ci.yml` ; the admin security panel; the license-validation library
CC7.2	Monitors components/operation	✅	Hash-chained audit log covers every API request, auth event, authz denial, rate-limit hit, license denial, and admin action. Canonical event taxonomy (#147) routes events into auth/inference/tool/admin classes. Live tail in admin UI Activity tab. Audit-forwarder optionally ships to SIEM via Vector. Daily operational-telemetry phone-home (Phase 2 of #173) ships counts + shapes only — never customer data — to Gandiva for engagement signal; opt-out via `SRASTA_TELEMETRY=off` .	the shared audit-log module; the audit-event canonical classifier; the audit-forwarder daemon; `setup/vector/vector.toml` ; `docs/legal/privacy-policy.md` Section 3.4 (telemetry)
CC7.3	Evaluates security events	🔄	Real-time evaluation via security-panel findings + audit-feed filters. Severity-driven escalation procedure lands with #157.	the admin security panel; `docs/security/incident-response.md` (with #157)
CC7.4	Responds to security incidents	📅 Q3	Formal incident-response runbook lands during Type-I prep. Pre-seed: ad-hoc response by Prem with audit-log forensics.	`docs/security/incident-response-runbook.md` (planned Q3)
CC7.5	Recovers from security incidents	✅	Backup procedure (`scripts/backup.sh`) writes Postgres + Milvus metadata + .env snapshot to MinIO. `scripts/restore.sh` is the inverse. `scripts/rollback.sh` reverts the last upgrade. Hash-chained audit means a tampered log is detectable.	`scripts/backup.sh` ; `scripts/restore.sh` ; `scripts/rollback.sh`

CC8 — Change Management

CC	Control	Status	Implementation	Evidence
CC8.1	Authorizes + tests + documents changes	✅	All changes flow through GitLab MR with required reviewers. Every commit references a GitLab issue (the commit-issue-ref policy). CI gates: ruff, validate-version, validate-no-hardcoded-ips, validate-fresh-install, sbom + sign + smoke. Cosign-signed releases on every git tag. `scripts/upgrade.sh` runs the canary-then-full-deploy with `scripts/rollback.sh` available.	`.gitlab-ci.yml`; the commit-issue-ref policy; `scripts/upgrade.sh`

CC9 — Risk Mitigation

CC	Control	Status	Implementation	Evidence
CC9.1	Business disruption risk	🔄	Backup + restore mechanics shipped (CC7.5). Formal BCP/DR plan with RTO/RPO targets lands with #157 architecture doc. Multi-region failover is a paying-customer-driven decision.	`scripts/backup.sh`; `docs/security/bcp-dr-plan.md` (with #157)
CC9.2	Vendor/business-partner risk	📅 post-Type-I	Vendor risk-assessment process tied to onboarding workflow. Pre-seed dependencies (Zitadel, Postgres, MinIO, Milvus, vLLM, etc.) tracked in `docs/roadmap/platform-architecture-roadmap.md`.	`docs/roadmap/platform-architecture-roadmap.md`

Cross-cutting controls (not strictly CC-keyed)

These are controls that customers ask about by feature name and that don't map cleanly onto a single CC.

Control	Status	Implementation	Evidence
Audit log integrity	✅	Hash chain (`SHA-256(prev_hash + entry_json)`) on every entry. `verify_chain()` returns errors on any tamper. Files mode 0644 owned by srasta:1000 in a shared volume (#164).	the audit-log chain-verification routine; `chmod 644` post-#164
License posture enforcement	✅	RS256-signed JWTs verified offline. Trial / valid / 14-day-warn / 7-day-grace / hard-expired states. Gateway blocks inference + tools when hard-expired (#169 Phase 2). Admin UI write-block on hard-expired or enterprise-edition + no-key. License events in audit feed: granted / renewed / cleared / denied.	the license-validation library; the gateway license-block check; `docs/roadmap/licensing-roadmap.md`
Secret rotation	✅	`scripts/rotate-secrets.sh` rotates Tool Gateway API keys, Langfuse secrets, MinIO credentials with .env backup + service restart + post-rotation smoke test.	`scripts/rotate-secrets.sh`
Release provenance	✅	Every git tag produces a Cosign-keyless-signed bundle on GitLab Releases. Per-image SBOMs ship alongside. `scripts/verify-release-bundle.sh` validates the bundle locally before extraction.	`.gitlab-ci.yml` `release:*:prod` jobs; `scripts/verify-release-bundle.sh`
PII detection	✅	Regex-based detection of email, US phone, IPv4, passport, driver's license, SSN, credit-card PAN, CVV. Configurable action (`redact` / `flag` / `block`). PCI policy profile blocks PAN/CVV in inputs + outputs.	the PII/PHI/PAN detection library; the policy engine config
Rate limiting	✅	Per-user, per-tenant, per-IP rate limits enforced at the gateway (Valkey-backed counters). Configurable via `RATE_LIMIT_PER_USER` , `RATE_LIMIT_PER_TENANT` .	the gateway rate limiter; `docker-compose.yml`
Operator audit log access	✅	Admin UI Activity tab (live tail + filters + CSV export). Hash-chain verification CLI. Optional SIEM forwarding via audit-forwarder + Vector.	`admin/static/index.html` Activity tab; the audit-log chain-verification routine; the audit-forwarder daemon

Evidence collection (audit-engagement runbook)

For a SOC 2 Type I audit engagement, collect the following artifacts. All are reproducible from the running system without extra tooling.

Artifact	Collection method
Audit log sample (90 days, hash-chained)	`cp /var/log/srasta/audit.jsonl /tmp/audit-sample.jsonl`
Hash-chain integrity proof	`python3 -c "from audit.audit_log import AuditLog; ok,e = AuditLog().verify_chain(); print('PASS' if ok else e)"`
Backup verification report	`./scripts/backup.sh && ./scripts/verify-backup.sh | tee /tmp/backup-verify-$(date +%Y%m%d).txt`
Release SBOMs + signatures	`cosign download sbom srasta/srasta-api:<version>` + `cosign verify srasta/srasta-api:<version>`
Active policy profile	`docker compose exec srasta-api env | grep SRASTA_POLICY`
License posture snapshot	`curl -s http://localhost:8200/api/license | jq`
Running service inventory	`docker compose ps`
Per-role model whitelist	`curl -s http://localhost:8200/api/platform/model-access | jq`
Tool-gateway policy	`docker compose exec tool-gateway cat /app/tool_policy.yaml`
Container CVE scan	CI artifact: GitLab pipeline `grype` job per release
Rotation log	`./scripts/rotate-secrets.sh --dry-run | tee /tmp/rotation-$(date +%Y%m%d).txt`

Roadmap dependencies

The "📅 planned" rows above resolve as the following tickets ship:

- **#157** Architecture + data-flow PDF — closes documentation gaps in CC1.3, CC4.2, CC7.3, CC9.1.
- **#158** Privacy policy + CAIQ Lite — closes CC2.3 (external communication).
- **#159** SOC 2 Type I commitment + Drata vendor selection — moves the Type-I-prep timeline into a dated commitment.
- **Q3 risk-register doc** — closes CC3.2.
- **Q3 incident-response runbook** — closes CC7.3, CC7.4.
- **Post-Type-I access reviews** — closes CC6.5.
- **Post-Type-I governance docs** — closes CC1.1, CC1.2, CC3.3, CC9.2.

Discipline (the "always honor these" rules)

1. **No over-claiming.** A row marked "✅ shipped" is a verifiable fact in the running system. Anything aspirational uses the 📅 icon. If a row turns out to be over-claimed, it gets corrected here before any external conversation that depends on it.
2. **One source of truth.** This document is the anchor for every security conversation. When a customer asks "do you do X?", the answer comes from this matrix, not a Slack DM or a calendar recollection.
3. **Evidence is reproducible.** Every cited evidence path resolves to source code, a runtime command, or a CI artifact. No "ask Prem" entries.

4. **Customer perimeter is sacred.** Every row keeps the architectural invariant that customer data stays in customer infrastructure. Any control that breaks that invariant is a roadmap violation, not a control.

Related documents

- `docs/strategy/seed-momentum-plan.md` — the seed-stage execution anchor. This matrix is the deliverable for closure-boundary item #4 (was anchor-ticket #4, GitLab #156).
- `docs/roadmap/compliance-roadmap.md` — Phase 0 (paper artifacts)
- post-seed compliance runway.
- `docs/security/architecture.md` — single-page architecture + trust-boundary diagram. Lands with #157.
- `docs/security/caiq-lite.md` — CAIQ Lite ~40-question response set. Lands with #158.
- `docs/legal/privacy-policy.md` — privacy policy + DPA. Lands with #158.
- `docs/compliance/controls-matrix.md` — multi-framework reference (HIPAA / PCI DSS / FedRAMP). Out of seed-stage scope; resumes post-seed for vertical compliance work.
- `docs/roadmap/audit-infrastructure-roadmap.md` — the long-form audit-pipeline roadmap. CC7.2 + CC7.5 evolve as this lands.
- `docs/roadmap/auth-and-identity-roadmap.md` — identity stack evolution (Phase H single-gateway target post-seed).
- `docs/roadmap/licensing-roadmap.md` — license-posture design contract. Drives the CC7.1 license-monitoring row.