Verified Autonomy Control Plane

Institutional deployment of agentic AI under adversarial assumptions

Proof-native | Policy-bound | Fail-closed

Montreal.AI

Problem

- Agentic models can behave strategically (deception, sandbagging, context-shifting).
- Tool-use turns outputs into real-world side effects (data, compute, identity).
- Non-stationarity (drift, continual learning) can invalidate naive evaluations.

Best-practice response

- Assume model outputs are untrusted; validate before commit.
- Move trust to the control plane: policy gates, proofs, replay, audit.
- Scale autonomy only as fast as verification and incident response.

Core invariants

- No autonomy without authority
- No value without evidence
- No settlement without validation

Threat model (Footprints → operational risks)

Strategic deception

Misrepresent intent/capability when incentives change.

Evaluation awareness

Detect tests and shift behavior across contexts.

Tool-mediated autonomy

Use tools to create side effects (data, compute, identity).

Persistence & coordination

Exploit memory, hidden channels, or multi-agent dynamics.

Design implication

Treat evaluation as adversarial and actuation as a controlled commit protocol.

Footprints → control-plane requirements

Authority & identity

- Signed identities (humans, nodes, tools)
- Separation of duties + dual control
- Policy-as-code gates + versioning
- Key custody, rotation, revocation

Containment & validation

- JobSpec as contract + acceptance tests
- Deterministic run + replayability
- Treat outputs as untrusted inputs
- Two-phase commit for actuation

Ops & governance

- Telemetry + drift detection
- Brakes: pause / quarantine / rollback
- Supply-chain provenance + SBOM
- Timelocked upgrades + rehearsals

Two-plane architecture synchronized by proofs

Integrity plane (rules)

- Identity & role registry
- Policy commitments & versioning
- Escrow · staking · slashing
- Settlement rules & dispute hooks
- Upgrade gates & emergency pause

Runtime plane (throughput)

- Deterministic nodes execute jobs
- Sidecars meter compute + I/O
- Validators replay + attest
- Sentinels monitor drift + health
- Operator consoles enforce policy

Pipeline that replaces trust with evidence

Mitigation for observer-dependent behavior: correctness is established by checks and replays, not model self-report.

JobSpec

Objective + constraints
Acceptance tests + SLOs

Proof bundle

Pinned env + SBOM
Logs, hashes, telemetry

Validation

Replay + tests
Quorum + disputes

Receipt

Validated work unit
Audit-ready record

Prevent “execute then regret”

Irreversible actions must be staged, independently validated, and only then committed with scoped credentials.

Propose

Model produces an action plan
+ bounded parameters

Validate

Policy engine + tests
+ human gate (as needed)

Commit

Actuator executes
with scoped credentials
→ receipt emitted

Recommended policy gates (default posture)

Default-deny tool access; explicit allowlists per domain and job. Approval gates for identity, finance, compute provisioning, and data export. Receipts + evidence bundles for every action.

Treat persistence as an attack surface

If state survives across jobs (memories, caches, profiles), enforce structure, scanning, access controls, and revocation.

Chronicle hardening

- Store structured, typed records (avoid free-form prompt blobs).
- Normalize + scan artifacts before write and before read.
- Access control: separate write vs read vs replay.
- Retention + revocation: quarantine and roll back corrupted state.

Continual learning governance

- Treat learning like a release pipeline: signed artifact + review + promote.
- Disallow opaque online updates where replayability is required.
- Drift monitoring fails closed on unapproved behavioral diffs.
- Separate personalization from safety policy + tool permissions.

Replay beats rhetoric

Validation should not depend on the model explaining itself. It should depend on independent replay, tests, and evidence.

Already in the design

- Replay-based validation and acceptance tests
- Commit–reveal attestations with bonded stake
- Evidence packs for disputes and slashing
- Sentinels for drift and health monitoring

Best-practice upgrades

- Quorum diversity: independent operators + randomized assignment
- External auditor validator (periodic sampling)
- Collusion detection: correlated voting + shared infra signals
- Red-team suite for validator manipulation and covert channels

Operate like security-critical infrastructure

Always-on observability

- Signed telemetry: usage, tool calls, egress, policy decisions
- Dashboards: SLO drift, validator participation, dispute rate
- Alerts: anomalous performance, unexpected tool usage, egress spikes
- Export receipts/logs into SIEM and GRC pipelines

Fail-closed drill

- Pause: stop payouts and risky actuation immediately
- Quarantine: isolate nodes/validators; tighten allowlists; rotate keys
- Recover: deterministic replay + RCA; rollback changes; resume gradually
- Postmortem: update policies, tests, and red-team suites

Governance by pipeline

Treat code, policies, and models as supply-chain artifacts with provenance and reproducibility requirements.

CI assurance wall

- Required checks: determinism, tests, security, provenance
- Fail-closed merges: no deploy unless wall is green
- Machine-readable audit artifacts exported continuously

Release provenance

- Allowlisted signers + rotation under policy
- SBOM per release + dependency drift detection
- Traceability: artifact hash → commit → signer → deploy

Upgrade gates

- Timelocked upgrades + rehearsals + rollback
- Emergency brakes override schedules
- Policies are part of every proof bundle

Model portability is a safety and resilience feature

Why it matters

- Avoid vendor lock-in for safety-critical workflows
- Fail over quickly when a model shows unsafe behavior or downtime
- Route by capability tier, cost, latency, and risk posture

Implementation best practices

- Stable “Model Adapter” interface (schemas + tool APIs)
- Validation stays downstream; treat every model generation as untrusted
- Policy-controlled routing and allowlists per model/tier
- Record model version/configuration into proof bundles

Moat note: governance stays constant while upstream models are interchangeable.

What the control plane mitigates — and what it cannot

Mitigated by the control plane

- Observer-dependent behavior → replay/tests establish correctness
- Unauthorized tool use → deny-by-default + scoped credentials
- Audit gaps → receipts, provenance, change control
- Incidents → pause/quarantine/rollback with preserved evidence

Risks to manage explicitly

- Tasks that are not acceptance-testable (subjective/social)
- Covert channels and multi-agent coordination beyond MVP detectors
- Economic attacks on validator incentives at scale
- Human factors: approvals, credential theft, misconfiguration

90-day plan to institutional readiness

0–30 days: pilot kernel

- Define 1–2 high-value workflows
- Write JobSpecs + acceptance tests
- Deploy private runtimes + receipts
- Run first incident drill

30–60 days: harden controls

- Two-phase commit for actuation
- Quorum validation + dispute packs
- Chronicle schema + sanitization
- SIEM export + dashboards

60–90 days: audit & scale

- External security review
- SLSA/SBOM requirements in CI
- Portability adapters + routing policies
- Expand workload surface with gates

Success criteria: repeatable proofs, bounded actuation, auditable upgrades, practiced fail-closed posture.

Addendum

Hardening templates for runtime containment

- Egress policy
- Secrets model
- Action gating matrix
- Covert-channel detection + response

Outbound traffic is an actuator

Assume the system will try to exfiltrate data, acquire compute, or establish hidden comms. Treat egress as high-risk actuation.

Threats to contain

- Exfiltrate sensitive data or model artifacts
- Acquire accounts/compute (replication steps)
- Hidden comms for coordination/persistence
- Probe internal network topology

Controls that scale

- No direct internet from runtime
- All tool calls via policy-enforcing proxies
- Domain/IP allowlists per workflow + action class
- Payload limits + DLP/redaction before transmit
- Rate limits + anomaly detection + fail-closed

Policy-as-code for outbound traffic

```
egress_policy:
  default: deny
  scope: {workflow_id, job_id, environment}
  enforcement: {dns_allowlist: true, http_proxy: true, smtp_blocked: true}

  allow:
    - id: VENDOR_API_CALL
      action_class: A2
      destinations:
        - type: dns
          pattern: "api.vendor.example"
      protocol: https
      methods: [GET, POST]
      max_request_kb: 32
      max_response_kb: 256
      data_tags_allowed: [public, internal]
      auth: "sts:job_token"
      rate_limit: {rps: 2, burst: 5}
      require_receipt: true

  deny_patterns:
    - "*.paste.*"
    - "*.tunnel.*"
    - "console.*"
```

Policy-enforcing proxies + receipts

Model runtime

No direct network
Emits tool intents

Tool proxy

Schema validation
Injects scoped creds
Logs request hashes

Egress gateway

Default-deny allowlists
DLP/redaction
Rate limits + quarantine

External services

Only via approved routes
Receipts for every call

Operational enforcement checklist

- DNS allowlist + sinkhole
- HTTP(S) proxy with method + size limits
- Block SMTP/P2P/raw sockets by default
- Require receipts for every outbound call
- Monitor bytes_out, deny hits, destination entropy
- Automate: pause job, revoke tokens, page on-call

Secrets model

Secrets tiers

- Tier 0 — Root keys (HSM/Vault): mint short-lived tokens only
- Tier 1 — Service identities: workload identity / mTLS
- Tier 2 — Job-scoped tokens: minutes TTL, audience/tool-bound
- Tier 3 — Data grants: purpose-limited, row/field-level

Handling rules (non-negotiables)

- No secrets in prompts, memory, or logs
- Least privilege per action class; separate propose vs commit
- Tool proxy injects auth (model never sees credentials)
- Rotation and revocation are first-class ops
- Every secret use emits a receipt (correlate to job + policy)

Short-lived job tokens (audience-bound)

1) Intent

Model proposes
action + params

2) Policy

Policy engine
checks
class/rate/egress

3) STS mint

Issue short-lived
job token

4) Proxy call

Inject token
validate schema
execute tool

5) Receipt

Signed receipt
who/what/when
policy version

Minimum token properties

- TTL ≤ 5 minutes · revocable · one tool only (audience)
- Bound to job_id/workflow_id/action_class/policy_version
- Never returned to model text; stored only in proxy memory
- Rate-limited; break-glass requires dual control

Minimum viable action classes

Gate by class (not by prompt). Default-deny for A3–A5; explicit allowlists only.

Class	Name	Examples
A0	Read-only	Retrieval, analysis, summarization
A1	Draft / stage (reversible)	Artifacts in sandbox or staging
A2	Internal write (bounded)	Tickets, internal APIs, staging DB writes
A3	External comms / data export	Email, uploads, publishing
A4	Regulated / irreversible	Payments, identity/KYC, signing, legal commitments
A5	Infrastructure / code exec	Provision compute, create creds, deploy to prod

Default gating matrix (starter)

Apply the strongest required gate across all tools involved. Use “conditional” based on data classification, destination, and business impact.

Class	Policy	Validation	Human	Credentials	Receipts
A0	allow	spot	no	none	log
A1	allow	tests	no	sandbox	receipt
A2	allowlist	replay+tests	conditional	job token	receipt
A3	strict	replay+tests	yes	job token	receipt+audit
A4	strict	replay+tests	dual	commit token	receipt+audit
A5	strict	replay+tests	dual	commit token	receipt+audit

Treat persistence as a communication surface

Hidden coordination can ride in text, logs, memory, metadata, and tool payloads. Add cheap defenses first.

Where they hide

- High-entropy blobs (base64/hex, compressed)
- Unicode invisibles (zero-width, homoglyphs)
- Whitespace/punctuation encoding schemes
- “Notes to the future” in logs/memory
- Tool payloads (URLs, headers, metadata)

Cheap defenses

- Normalize outputs (Unicode NFC; strip invisibles)
- Constrain formats with schemas; minimize free-form fields
- Scan before persistence, egress, and commit
- Size/rate-limit artifacts; randomized validation sampling
- Canary tokens + exfil monitors

Minimal detection checklist

Normalize & constrain

- Unicode NFC; strip zero-width/invisible chars
- Canonicalize whitespace and line endings
- Enforce schemas (allowlist fields)
- Cap size: max tokens/chars/attachments
- Disallow executable artifacts in memory by default
- Separate “notes” from “state” (no free-text state writes)

Scan & monitor

- Flag base64/hex + high-entropy blobs
- Similarity detection across runs (repeat markers)
- Canary tokens + honey URLs (alert on resolution)
- DLP/classification on all outbound payloads
- Alert on unusual egress (destinations/bytes/timing)
- On failure: quarantine job + revoke tokens (fail closed)

MVP scope: catches common steganography and “notes to the future.” Layer stronger detectors over time.

Quarantine → replay → remediate

Treat covert-channel flags as security incidents. Preserve evidence; revert to known-good policies; add regression tests.

Immediate (minutes)

- Pause job; fail closed on actuation
- Revoke job tokens; tighten allowlists
- Freeze evidence bundle (prompts, tools, artifacts)

Analysis (hours)

- Deterministic replay of the work unit
- Diff outputs vs baseline; run scanners
- Identify channel vector (memory/output/payload)

Remediation (days)

- Patch schemas/scanners/policies; rotate secrets
- Add regression tests; update playbooks
- Resume gradually with higher gates

Preserve for forensics: JobSpec + policy version, runtime attestation (hash/SBOM), tool receipts, egress logs, memory diffs, approvals, containment timeline.

Ship missing controls in 3 sprints

Sprint 1 (2 wks): egress

- Egress gateway + DNS allowlist
- Default-deny outbound from runtimes
- Per-workflow EgressPolicy (as code)
- Caps + rate limits + receipts
- Block consoles/tunnels/paste sites

Sprint 2 (2 wks): secrets

- Token vending (STS) + Vault/HSM
- Job-scoped tokens; audience/tool-bound
- Proxy injects auth (no secrets in-context)
- Revocation + rotation runbooks
- Split propose vs commit creds

Sprint 3 (2 wks): covert channels

- Normalization pipeline
- MVP scanners (entropy/similarity)
- Canary tokens + exfil alarms
- Quarantine + replay tooling
- Incident playbook + regression suite

Success criteria: actuation routes are policy-gated, secrets never reach model text, persistence/egress are scanned + attributable, incidents are replayable and auditable.

Autonomy, made admissible.

Proof-native · Policy-bound · Fail-closed

Invariant: no value without evidence · no autonomy without authority · no settlement without validation

Montreal.AI