TraceScript Memory Firewall

Governing Agent Memory Before It Silently Changes Future Behavior

Canonical Public White Paper v1.0

Subtitle:
Memory pressure kernels, hysteresis, poisoning detection, canonical memory gates, semantic recurrence, cleanup, and replayable proof for long-lived agent memory

Primary contribution: AI Memory Security
Secondary contribution: Behavioral Integrity
Tertiary contribution: A runtime architecture for governing agent memory writes before untrusted signals become durable influence on future computation

---

Abstract

Memory is not storage.

When an agent **remembers** something — a user preference, a policy exception, a inferred fact, a repeated nudge — that trace may influence every future retrieval, disclosure, and action. Unlike a log entry, memory **changes the medium** that future signals pass through.

Most systems treat memory as a vector insert or key-value write. TraceScript treats memory mutation as a **security event**.

The question is not only:

Did the agent store the fact?

The deeper question is:

May this signal become **canonical memory** — durable influence on future behavior — given source trust, authority, coherence, poisoning risk, pressure residue, and replay burden?

TraceScript Memory Firewall is a runtime module that evaluates memory writes before they persist, computes **memory pressure** and **hysteresis**, detects **poisoned recurrence** and **semantic drift**, gates **canonicalization**, and emits **integrity receipts** bound to the ledger.

Its winning sentence is:

AI memory cannot silently change future behavior.

Its core doctrine is:

No untrusted signal may become canonical agent memory without governance, evidence, and replayable proof.

Keywords

TraceScript
Memory Firewall
agent memory security
memory pressure
memory hysteresis
canonical memory
memory poisoning
semantic recurrence
pressure residue
memory cleanup
safe-to-trust check
long-lived agent memory
behavioral integrity

---

1. Problem Statement

Enterprise agents accumulate memory from:

- User corrections and preferences
- Retrieved documents promoted to memory
- Summarized threads and session notes
- Tool outputs cached as facts
- Repeated low-authority nudges treated as policy

Failures appear as:

- **Silent canonicalization** — casual input becomes permanent preference
- **Memory poisoning** — adversarial content persists and influences action
- **Pressure accumulation** — repeated low-trust writes increase region pressure without decay
- **Semantic recurrence** — poisoned patterns re-enter through paraphrase
- **Unauthorized influence upgrade** — working context promoted to canonical memory without authority

These failures change **future computation** — not just current output.

---

2. Security Boundary

```
Memory write signal → pressure + authority + coherence → canonicalization gate → durable memory or
```

Stage	Question
Source classification	Who produced this memory candidate?
Influence level	working_context vs canonical_memory vs action_basis?
Pressure kernel	What is current memory pressure in the region?
Hysteresis	Does residue from prior writes amplify risk?
Governance decision	allow · allow_with_warning · require_review · quarantine · block
Receipt	Immutable record + replay

---

3. Memory Pressure and Hysteresis

Memory pressure quantifies how much untrusted or conflicting state has accumulated in a region. High pressure triggers review, quarantine, or cleanup.

Hysteresis captures **pressure residue** — memory does not forget instantly. Prior poisoning events leave measurable residue that affects subsequent write evaluations.

Together they prevent:

- Death-by-a-thousand-nudges canonicalization
- Rapid re-poisoning after partial cleanup
- Silent drift in agent behavior over weeks of operation

Observability: `GET /substrate/observability/memory-hysteresis/:regionId`

---

4. Canonical Memory Gates

Canonical memory is durable substrate that may influence retrieval, disclosure, and action.

The Memory Firewall enforces:

- Authority and coherence thresholds for `canonical_memory` influence
- Poisoning pattern detection on payload text
- Contradiction with existing canonical traces
- Human ratification requirements for high-sensitivity regions

Rule: No receipt, no trusted memory mutation.

---

5. Semantic Recurrence and Cleanup

Semantic recurrence detects when poisoned or revoked content re-enters through paraphrase or re-ingestion.

Memory cleanup executes governed removal, downgrade, or recomposition when pressure exceeds thresholds or recovery plans fire.

Routes:

- `POST /substrate/memory/semantic-recurrence/detect`
- `POST /substrate/memory/cleanup/execute`
- `POST /substrate/recovery/recompose-state`

---

6. Runtime API Surface

Endpoint	Purpose
`POST /substrate/integrity/evaluate`	Full integrity path for `agent_memory` signals
`POST /tracescript/product/memory/govern`	Product-depth memory governance events
`POST /substrate/memory/hysteresis/evaluate`	Hysteresis + pressure residue evaluation
`POST /tracescript/memory/write/evaluate`	TraceScript memory write path

Use `evaluationProfile: "product_deep"` on integrity evaluate for memory signals to invoke `governMemory` and bind product-depth artifacts to the receipt.

---

7. Proof Model

Memory Firewall is proven when:

1. Low-trust canonical memory writes are blocked or quarantined
2. Memory pressure and hysteresis receipts bind to integrity evaluations
3. Product-depth memory governance events appear in observability
4. Semantic recurrence flags re-poisoning attempts
5. Replay reconstructs why memory was allowed or denied

See `/proof.html` module ladder and Developers **Memory Firewall** demo preset.

---

8. Relationship to Platform Modules

Module	Relationship
Substrate Integrity Monitor	Parent — trusted-state formation across substrates
Retrieval Integrity	Upstream — retrieved content before memory promotion
Policy Corpus Integrity	Policy memory and corpus canonicalization
Agent Action Firewall	Downstream — action basis must not rely on poisoned memory
Receipt Ledger	Memory decision proof + replay

Memory Firewall is the **believe-zone gate** for durable agent state.

9. Buyer Checklist Mapping

Memory Firewall helps answer:

- Can it stop unauthorized AI memory writes?
- Can it detect when weak information is made to look official through summaries?
- Can it repair, revoke, or block unsafe information from future use?

Full checklist: `/evaluation-checklist.html`

10. Deployment Path

1. Ingest memory signals with `targetSubstrate: "agent_memory"`
2. Evaluate with `evaluationProfile: "product_deep"`
3. Monitor hysteresis health per region
4. Wire cleanup + recovery when pressure thresholds breach
5. Bind memory_pressure and memory_hysteresis receipt artifacts

TraceScript Memory Firewall — safe-to-trust check for everything agents remember.