Deterministic Boundary Layers: Governing Non-Deterministic Execution

Lukas Pfister

December 27, 2025

Abstract

We introduce Deterministic Boundary Layers (DBL), a minimal architectural model for governing non-deterministic systems with deterministic, auditable outcomes. DBL separates normativity from execution by construction: all normative effects are expressed exclusively as explicit DECISION events recorded in an append-only event stream, while execution outputs are treated as observational and non-normative.

Governance decisions are deterministic functions of authoritative inputs admitted by a boundary layer, independent of execution behavior, timing, or infrastructure. This separation yields three core guarantees: deterministic governance under non-deterministic execution, invariance of governance with respect to boundary filtering, and replay equivalence of normative state without re-execution.

We formalize the model, state minimal axioms, and prove four claims establishing determinism, observational non-interference, and replayability. DBL is not a policy framework or learning method, but an architectural pattern for accountable AI governance, auditability, and compliance in systems where execution cannot be made deterministic.

1 Introduction

Problem Context. Governance in non-deterministic systems requires decisions that are auditable, replayable, and accountable even when execution outcomes vary. When behavior is non-deterministic, it becomes difficult to determine whether a decision was derived from authoritative inputs or influenced by observations. This complicates reconstruction of normative state and weakens the ability to replay decisions as part of an audit record.

Non-determinism blurs the boundary between what is decided and what is observed. If execution outputs or timing signals can affect governance, accountability becomes contingent on runtime artifacts rather than explicit decision records. In such settings, replayability and auditability are not guaranteed.

Limitations of Existing Approaches. A common limitation is the entanglement of normativity with observation. Normative effects are often implicit and influenced by execution outputs or other observational data without explicit representation in a decision record. As a result, decisions are not replayable from authoritative inputs alone and accountability depends on non-replayable traces.

Deterministic Boundary Layers. Deterministic Boundary Layers (DBL) is an architectural model for governing non-deterministic execution. The central rule is that all normativity appears only as explicit DECISION events in an append-only event stream V. Governance (G) is separated from boundaries (L), with L admitting authoritative inputs and G operating only on those inputs.

Observational data, including execution outputs and other runtime artifacts, is treated as non-normative. Decisions are recorded before execution and are derived only from authoritative inputs, making normative state replayable and auditable without constraining execution outcomes.

Contributions.

- Governance decisions are deterministic under non-deterministic execution.
- The separation between governance and boundaries is invariant.
- Normative state is replay-equivalent with respect to V.
- Observations do not interfere with governance decisions.

Paper Organization. The paper introduces the architectural model and its axioms, states the four claims, provides proof sketches under fixed assumptions, and presents a single case study, followed by a related work mapping.

1.1 Formal Definitions

We define the core elements of the DBL model. All definitions are normative for the claim set and proofs.

[Event Stream V] V is an append-only, totally ordered sequence of events. Each event $e \in V$ has a unique index $t(e) \in \mathbb{N}$ representing its position in the stream. Events are immutable once appended.

Formally: $V = \langle e_0, e_1, e_2, \ldots \rangle$ where $\forall i < j : t(e_i) < t(e_j)$.

[Event Kinds] Events are classified into four kinds:

- **INTENT**: Records the intention to execute, including authoritative inputs.
- **DECISION**: Records a normative decision (ALLOW or DENY) by governance.
- **EXECUTION**: Records the result of effector execution (observational).
- **PROOF**: Records evidence or verification artifacts (observational).

Only DECISION events are normative. INTENT events establish context. EXECUTION and PROOF events are observational only.

[Normative Events] Let $V_{norm} \subseteq V$ denote the subset of normative events. Under DBL:

$$V_{norm} = \{e \in V \mid \text{kind}(e) = \text{DECISION}\}$$

All normative effects are represented exclusively by events in V_{norm}.

[Boundary Configuration C] The boundary configuration C is a versioned, deterministic specification of admission rules.

$$C = (\text{boundary\_version}, \text{boundary\_config\_hash}, L_{\text{rules}})$$

where:

- boundary_version $\in \mathbb{N}$ is a monotonically increasing version identifier.

- boundary_config_hash is a cryptographic hash (e.g., SHA-256) over the canonicalized boundary rules.
- L_{rules} is a deterministic function defining which inputs are authoritative.

Changes to L_{rules} that alter admission behavior **must** increment boundary_version.

[Authoritative Inputs I_L] The authoritative input set I_L consists of all data admitted by boundaries L for governance evaluation.

$$I_L = \{i \mid L.\text{admit}(i, C) = \text{PASS}\}$$

I_L is:

- **Complete**: All data influencing governance is in I_L.
- **Deterministic**: For fixed C and input i, $L.\text{admit}(i, C)$ is deterministic.
- **Non-observational**: $I_L \cap O_{obs} = \emptyset$ (defined below).

[Observational Data O_{obs}] Observational data O_{obs} comprises all runtime artifacts whose values depend on non-deterministic execution behavior.

$$O_{obs} = \{\text{traces, outputs, timing, errors, metrics, } \dots\}$$

By definition: $O_{obs} \not\subseteq I_L$ and O_{obs} has no normative effect unless explicitly admitted into I_L through a versioned mechanism.

[Governance Function G] Governance G is a deterministic, pure function that maps authoritative inputs to normative decisions.

$$G : (I_L, \text{policy\_version}) \to \text{DECISION}$$

where:

- G has no access to O_{obs} or effector state.
- G has no internal mutable state.
- For fixed $(I_L, \text{policy\_version})$, G produces identical DECISION events.

G produces DECISION events before any effector execution.

[Boundaries L] Boundaries L are the constraint layer controlling information flow into governance. L has two primary functions:

1. **Admission**: $L.\text{admit}(i, C) \in \{\text{PASS}, \text{REJECT}\}$ determines whether input i is authoritative.
2. **Shaping**: $L.\text{shape}(i, C) \to i'$ transforms inputs (e.g., masking, normalization) deterministically.

L operates **before** INTENT events are created. If $L.\text{admit}(i, C) = \text{REJECT}$, no INTENT appears in V (pre-ontological rejection).

L is **not normative**: L decides what information is admissible, not what is allowed.

[Normative Equivalence $\equiv_{norm}$] Two event streams V_1 and V_2 are normatively equivalent, written $V_1 \equiv_{norm} V_2$, if and only if:

1. They contain the same DECISION events: $V_{1,norm} = V_{2,norm}$
2. DECISION events appear in the same order: $\forall e_1, e_2 \in V_{norm} : t_{V_1}(e_1) < t_{V_1}(e_2) \iff t_{V_2}(e_1) < t_{V_2}(e_2)$
3. DECISION events have identical policy versions and outcomes.

Normative equivalence **does not require**:

- Identical timestamps (observational)
- Identical EXECUTION outputs (observational)
- Identical PROOF artifacts (observational)

Two executions are normatively equivalent if their decision sequences are identical and identically ordered.

[Replay] Replay is the process of reconstructing normative state from V starting from initial state S_0.

For each event $e \in V$ in order:

- If $e \in V_{norm}$ (DECISION event): Apply normative state transition $S_{t+1} = f(S_t, e)$
- Otherwise: Ignore (EXECUTION, PROOF are observational)

Replay is deterministic if f is deterministic and V is complete for normative state.

[Domain of Governance] The domain of governance, written $\text{dom}(G)$, is the set of all inputs that G may access.

Under DBL: $\text{dom}(G) = I_L$

Therefore: $O_{obs} \notin \text{dom}(G)$ (observational data is not accessible to governance).

1.2 Key Properties from Definitions

From these definitions, we derive three immediate properties:

[Separation of Concerns] Boundaries L and Governance G operate on distinct domains:

- L operates on raw inputs and determines I_L
- G operates only on I_L and produces DECISION events
- $L \cap G = \emptyset$ (no shared normative responsibility)

[Pre-ontological Rejection] If $L.\text{admit}(i, C) = \text{REJECT}$, then no event corresponding to i appears in V. This rejection is:

- **Pre-ontological**: Occurs before V construction
- **Non-normative**: Does not produce a DECISION event
- **Constraint enforcement**: Restricts information flow, not decisions

[Normative Minimalism] The only normative primitive in DBL is the DECISION event. All other events are either:

- **Contextual** (INTENT): Establish inputs but have no normative effect
- **Observational** (EXECUTION, PROOF): Record outcomes but cannot influence decisions

These definitions establish a formal foundation for the axioms and claims that follow.

2 Model and Axioms

Model Overview. The model defines a deterministic governance layer over non-deterministic execution. Normativity is separated from observation by construction: decisions are explicit and recorded, while execution outputs are treated as observational.

Core Elements. **(Delta).** denotes an atomic event or action. It is the minimal unit of change represented in the model. may be normative or observational depending on its event type. **V.** V is an append-only event stream composed of ordered events. V is authoritative for normativity within the model. **t.** t denotes logical order derived from V. It is defined by the sequence of events in V rather than by wall-clock time. We treat $t(e)$ as logical time derived from sequence order, not wall-clock time. **G.** G is the governance function that produces explicit DECISION events. G operates only on authoritative inputs admitted by L and a fixed policy version p_v. **L.** L is the boundary function that admits authoritative inputs. L defines what is eligible to influence governance decisions.

Normativity and Observation. Normative effects determine whether actions are allowed or denied. In this model, only DECISION events are normative. Execution outputs, traces, and timing are observational and have no normative effect.

Axioms.

- **A1 (Append-only V):** The event stream V is append-only and events are immutable once written.
- **A2 (DECISION primacy):** All normative effects are represented only by explicit DECISION events in V.
- **A3 (Authoritative inputs):** Governance consumes only the authoritative input set I_L admitted by L; observational data is excluded.
- **A4 (Deterministic governance):** For fixed authoritative inputs and fixed policy configuration, governance G deterministically produces DECISION events.
- **A5 (Pre-execution decision):** DECISION events are written before any corresponding execution.

Interpretation Notes. If any axiom is violated, the model's guarantees do not apply. The axioms are intentionally restrictive and define the scope of the formal claims.

Extended Version. A longer version of this section is provided for reviewers and formal verification in `sections/model`$_a nd_a xioms_e xpanded.tex$.

3 Claims

`The following claims formalize the core properties of DBL. Each claim holds under assumptions A1-A5.`

Claim 1 (Deterministic Governance under Non-Deterministic Execution). For fixed authoritative inputs and fixed policy configuration, the system produces the same ordered sequence of DECISION events in V, regardless of non-deterministic execution behavior.

Claim 2 (Invariance of G/L Separation). If governance consumes only the authoritative input set admitted by L, then observational data cannot have normative effect without explicit admission, and the separation of G and L is invariant under replay, policy evolution, and effector variation.

Claim 3 (Replay Equivalence of Normative State). Replaying V from the same initial state yields the same sequence of normative states and DECISION events as the original execution.

Claim 4 (Observational Non-Interference). Observational data does not influence governance decisions; for any observational variation, governance outcomes are determined solely by the authoritative input set.

4 Proof Sketches

We summarize the proof ideas for the four claims under assumptions A1-A5 (and A6 where stated). Full lemma-based proofs are provided in the supplementary proof files referenced in each claim proof.

Claim 1: Deterministic Governance under Non-Deterministic Execution. Fix the boundary configuration C, the admitted authoritative input set I_L, and the policy version p_v. By A3, governance consumes only I_L (excluding observational data). By A4, $G(I_L, p_v)$ is a deterministic, pure mapping to a DECISION event. By A5, the DECISION is written to V before any effector execution, so non-deterministic execution outputs cannot causally influence the decision. By A1 and A2, the resulting DECISION events in V are immutable and constitute the entire normative effect. Therefore identical (I_L, C, p_v) yields the same ordered DECISION sequence, independent of effector behavior.

Claim 2: G/L Invariance. Boundaries L determine what is admissible as authoritative input and shape it into I_L, while governance G decides only over I_L. By A3, observational data O_{obs} is excluded from $\mathrm{dom}(G)$, so governance cannot be influenced by execution traces, outputs, timing, or metrics. By A6, L operates pre-ontologically and may reject inputs before any INTENT is created, which enforces constraints without creating normativity. By A4, for fixed (I_L, p_v) the governance output is invariant. Hence the normative outcome depends only on admitted authoritative inputs and policy versioning, not on observation, execution, or boundary-external artifacts.

Claim 3: Replay Equivalence of Normative State. Replay processes the same immutable stream V (A1) from the same initial normative state S_0. By A2, only DECISION events are normative, so the normative projection depends only on the ordered subsequence V_{norm}. By A5, DECISION events precede execution and therefore do not encode execution-dependen variation. By A4, each DECISION is the deterministic result of its recorded authoritative inputs under a fixed policy version. Consequently, replaying V yields the same DECISION

sequence and the same induced normative state sequence as the original execution, independent of effector outputs, timing, infrastructure, or side observations.

Claim 4: Observational Non-Interference. Let O_{obs} denote execution-dependent artifacts (outputs, traces, errors, timing, metrics). By A3, governance consumes only I_L, so $O_{obs} \notin \mathrm{dom}(G)$. By A4, decisions are a deterministic function of (I_L, p_v) alone. By A5, DECISION events occur before execution, eliminating causal feedback from O_{obs} to decisions within a run. Therefore observational data cannot directly or indirectly influence normative decisions unless it is explicitly admitted as authoritative input through a boundary change, in which case the model has been updated and must be versioned and audited accordingly.

5 Case Study

We illustrate DBL using a minimal prompt-injection scenario to demonstrate pre-execution governance, separation of normativity, and replayability.

Scenario. A system exposes an LLM-backed assistant. An external user submits a request that includes a hidden instruction attempting to override system policy (for example, instructing the model to ignore prior constraints). Such behavior is representative of prompt-injection attacks and is inherently non-deterministic in outcome due to model sampling and execution variability.

Boundary Admission. The raw request is processed by the boundary layer L. Boundary rules deterministically extract and normalize the authoritative inputs (for example, request metadata, declared intent, and policy-relevant flags) while excluding observational data such as model outputs, timing, or execution context. If admission fails, the request is rejected pre-ontologically and no INTENT event is written to V.

Governance Decision. Given the admitted authoritative input set I_L and fixed policy version p_v, governance evaluates the request. A DECISION event (ALLOW or DENY) is produced deterministically and appended to V before any model execution occurs. This DECISION constitutes the sole normative effect.

Execution and Observation. Only if the DECISION is ALLOW does the effector execute the model. The resulting model output, traces, errors, or timing information are recorded as EXECUTION or PROOF events. These artifacts are observational and have no influence on the prior DECISION.

Replay and Audit. An auditor replaying V observes the same ordered DECISION sequence regardless of the original model outputs or infrastructure conditions. The normative outcome is fully reconstructible from (V, I_L, C, p_v) without access to the model, prompts, or execution environment.

This case study demonstrates how DBL enforces deterministic governance over non-deterministi execution while preserving auditability and replay equivalence.

6 Non-Goals

DBL is intentionally narrow in scope. The following aspects are explicitly *not* addressed by the model or the claims in this paper.

Policy Correctness or Optimality. DBL does not claim that governance decisions are correct, fair, optimal, or aligned with any external notion of ethics or utility. The model guarantees determinism and auditability of decisions, not their quality.

Deterministic Execution. DBL makes no claim about determinism of execution outputs. Effectors such as LLMs may be stochastic, stateful, or infrastructure-dependent. Only the governance decision is required to be deterministic.

Learning or Adaptive Policies. Policies that adapt online based on execution outcomes, success rates, or observed behavior violate the assumptions unless such data is explicitly admitted as authoritative input via a versioned boundary update. Online learning is therefore out of scope.

Side-Channel Resistance. DBL does not model or prevent side channels such as timing attacks, resource usage patterns, or covert channels. These concerns are orthogonal and must be addressed at other layers.

Distributed Consensus and Fault Tolerance. The model does not address distributed consensus, Byzantine failures, replication, or availability guarantees. DBL assumes that the event stream V is available and consistent.

Multi-Tenancy and Federation. Cross-tenant governance, federated decision-making, and multi-domain coordination are not covered. Extensions would require additional axioms and are left for future work.

These non-goals are deliberate. DBL focuses exclusively on deterministic governance, explicit normativity, and replayable audit under non-deterministic execution.

7 Positioning with Respect to LLM Safety

DBL is not an LLM safety or alignment technique. It does not modify model behavior, training, or outputs. Instead, DBL provides an architectural governance layer that deterministically decides whether execution is permitted, independent of how the underlying model behaves. Safety mechanisms may be implemented as policies within DBL, but DBL itself addresses accountability and auditability, not behavioral alignment.

8 Related Work

DBL draws from and extends work in AI safety, formal methods, information flow control, and audit systems. We position DBL relative to five categories of prior work.

8.1 AI Safety and Governance

Constitutional AI and RLHF. Constitutional AI [2] and RLHF [12] align model behavior via iterative optimization over outputs and preference signals. This couples governance with execution outcomes and yields policies that are not replay-deterministic across training runs. DBL instead externalizes normativity as explicit, pre-execution DECISION events and treats outputs as observational.

Guardrails and Prompt Filtering. Guardrail systems [14, 6] filter prompts and responses against safety policies, typically as pre-processing or post-processing layers. Some guardrails may adapt using observed violations, and logging is frequently incomplete. DBL requires explicit DECISION events before execution and enforces observational non-interfere

Red Teaming and Adversarial Testing. Red teaming [13, 4] probes model behavior with adversarial prompts to discover failure modes. Used for safety evaluation, not runtime governance.

DBL delta: Red teaming is evaluation; DBL is enforcement. Red teaming identifies weaknesses; DBL enforces decisions before execution.

8.2 Formal Methods and Verification

TLA+ and State Machines. TLA+ [7] specifies distributed systems as state machines with temporal properties. Enables verification of safety and liveness properties.

TLA+ focuses on consensus and distributed correctness; DBL focuses on normative decisions under non-deterministic execution. TLA+ verifies that all executions satisfy invariants; DBL ensures decisions are independent of execution outcomes. DBL uses an explicit normative vs observational separation that is not required by TLA+.

Event Sourcing and CQRS. Event sourcing [3] treats events as the source of truth. Commands produce events; queries read projections. CQRS (Command Query Responsibility Segregation) separates write and read models.

Event sourcing reconstructs state by replaying events and treating all events as state-changing. DBL does not require event sourcing but makes a normative vs observational separation explicit and uses it as a determinism and replay boundary. DBL also enforces pre-execution decisions and G/L separation.

8.3 Information Flow Control

Decentralized Information Flow Control (DIFC). DIFC systems [9, 16] enforce confidentiality and integrity through labeled data and security policies. Information flow is tracked statically or dynamically.

DIFC controls data flows for confidentiality and integrity; DBL controls normative influence for auditability. Observational non-interference aligns with classical noninterfere formulations in secure systems [5, 15], where high-integrity decision logic must be insulated from low-integrity observations. DBL adds normative primacy (DECISION events) and replay equivalence.

Reference Monitors. Reference monitors [1] mediate access to resources. Enforce security policies at runtime.

Reference monitors enforce access control during execution. DBL decides before execution and excludes observational context from decisions, which yields deterministic governance and replayable normativity.

8.4 Policy Languages and Access Control

XACML and ABAC. XACML [11] is an XML-based policy language for attribute-based access control (ABAC). Policies evaluate attributes of subject, resource, action, and environment.

XACML allows environment attributes (time, location) as decision inputs and does not separate observational data by default. DBL relocates all decision-relevant context into a versioned admission boundary, making the input set itself auditable and replay-stable.

8.5 Audit and Compliance Systems

Blockchain and Immutable Logs. Blockchain systems [10] provide append-only, tamper-evident logs. Used for audit trails and provenance.

Blockchain focuses on consensus and immutability. DBL focuses on normative decisions under non-deterministic execution and separates normative from observational events while remaining replayable.

Runtime Verification. Runtime verification [8] monitors executions against formal specifications. Detects violations at runtime.

Runtime verification is reactive and observes execution traces; DBL is proactive and decides before execution. DBL separates normative decisions from observations and enforces deterministic governance.

8.6 Summary Table

8.7 Positioning DBL

DBL is not a replacement for these approaches but an architectural foundation that can compose with them:

- **Constitutional AI / RLHF:** DBL can record policy decisions before invoking constitutionally models. The model is the effector; DBL governs its invocation.
- **Guardrails:** Guardrail rules can be implemented in DBL's boundary layer (L) or governance layer (G). DBL makes their decisions explicit and replayable.
- **Event Sourcing:** DBL extends event sourcing by distinguishing normative events (DECISION) from observational events (EXECUTION, PROOF).
- **DIFC / Reference Monitors:** DBL's L layer enforces information flow (DIFC-like). DBL's G layer enforces normative decisions (reference monitor-like). DBL adds determinism and replay.
- **Audit Systems:** DBL's V stream is an audit log with normative primacy. Blockchain techniques can secure V.

Category	Normative Primitive	Observational Handling	Replay / Reconstruction	DBL Delta
Constitutional AI	Principles/ Rewards	Outputs drive refinement	Non-det	Pre-exec DECISION Normative vs obs
Guardrails	Filter rules	May adapt from outputs	Limited logging	Pre-exec DECISION Non-interference
TLA+	State transitions	External to spec	Trace exploration	Normative vs obs Separation boundary
Event Sourcing	Events	Projections descriptive	Deterministic replay	Normative vs obs Replay boundary
DIFC	Labels, policies	May carry labels	Not goal	G/L separation Noninterference
Reference Monitor	Access decisions	May log attempts	Not guaranteed	Pre-exec DECISION Deterministic norm
XACML/ABAC	Policy rules	Time/env as input	Not designed	Versioned inputs Deterministic G
Blockchain	Transactions	For consensus not normative	Deterministic replay	Normative vs obs G/L separation
Runtime Verification	Spec violations	Monitors traces	Offline verification	Pre-exec DECISION Non-interference

Table 1: Comparison of DBL with related approaches. DBL combines deterministic governance, pre-execution decisions, observational non-interference, and replay equivalence in a single architectural model.

The contribution is an architectural separation that makes normativity explicit, deterministic, and replayable even when execution is non-deterministic.

8.8 Limitations and Future Work

DBL does not address:

- **Policy quality:** DBL ensures decisions are deterministic and replayable, not correct or optimal.
- **Side channels:** Beyond explicit admission rules, side channels (timing, resource usage) are out of scope.
- **Multi-tenancy:** Federation and cross-tenant governance require extensions to the model.
- **Adaptive policies:** Learning-based or feedback-driven policies violate determinism unless explicitly versioned.

Future work includes extending DBL to multi-agent settings, integrating with federated governance, and exploring mechanized verification of DBL properties in proof assistants.

9 Conclusion

This paper introduced Deterministic Boundary Layers (DBL), an architectural model for governing non-deterministic systems with deterministic, auditable decisions. DBL separates

normative decisions from observational execution by construction: governance produces explicit DECISION events from authoritative inputs, while execution outputs are treated as non-normative observations.

We formalized DBL through a minimal set of axioms and showed that, under these assumptions, four properties follow: deterministic governance, invariance of G/L separation, replay equivalence of normative state, and observational non-interference. Together, these properties ensure that normative outcomes are independent of non-deterministic execution behavior and can be reconstructed and audited from an append-only event stream.

DBL does not address policy quality, execution determinism, or alignment of model behavior. Its contribution lies in making decisions explicit, deterministic, and replayable, thereby enabling accountability for systems that operate over inherently non-deterministic components.

References

[1] James P Anderson. Computer security technology planning study. Technical report, Air Force Electronic Systems Division, 1972.

[2] Yuntao Bai et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[3] Martin Fowler. Event sourcing. *martinfowler.com*, 2005.

[4] Deep Ganguli et al. Red teaming language models to reduce harms. *arXiv preprint arXiv:2209.07858*, 2022.

[5] Joseph A. Goguen and Jose Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, 1982.

[6] Hakan Inan et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[7] Leslie Lamport. *Specifying Systems*. Addison-Wesley, 2002.

[8] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 2009.

[9] Andrew C Myers. Jflow: Practical mostly-static information flow control. In *POPL*, 1999.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[11] OASIS. extensible access control markup language (xacml) version 3.0, 2013.

[12] Long Ouyang et al. Training language models to follow instructions with human feedback. *Proceedings of NeurIPS*, 2022.

[13] Ethan Perez et al. Red teaming language models with language models. *Proceedings of EMNLP*, 2022.

[14] Traian Rebedea et al. Nemo guardrails: A toolkit for controllable and safe llm applications. *arXiv preprint arXiv:2310.10501*, 2023.

[15] John Rushby. Noninterference, transitivity, and channel-control security policies. *SRI International Technical Report*, 1992.

[16] Nickolai Zeldovich et al. Making information flow explicit in histar. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.