Adversarial Robustness in Legal AI

A threat model for systems that state the law and act on it,
the principles that defeat each attack class, and the evidence

Research paper LL-ADV-01
Version 1.0.0, 26 November 2026

Abstract

Law is adversarial by construction. Every legal output has a party who gains if it is wrong: a counterparty who wants a false statement of the law accepted, an insider who wants an action executed without the approval it needs, a litigant who wants evidence altered without trace, an opponent who wants a superseded or foreign rule applied, an author who wants a poisoned source treated as authority. A legal AI system is robust only if it resists the party who chooses the input, not merely the average input. Accuracy on a test set says nothing about that party.

This paper states a threat model for legal AI with five adversary goals and, for each attack surface, the attack class, the principle that defeats it, the residual risk the principle leaves, and the evidence from this suite's experiments. It proves four simple results that organise the rest: against an adaptive adversary the relevant false-accept rate is the maximum over error classes, not the average; an authorization function cannot enforce a norm that depends on an attribute it does not receive; an unkeyed hash chain cannot detect truncation or rewriting without an independent witness of its head; and a canonical form trades rejection of benign variation for blindness inside what it folds.

The evidence is mixed and reported as such. Lawie's grounding check, as implemented at the code basis, accepted every minimal decisive edit and every misattribution [MEASURED]; a typed prototype stopped the edit types it models and few others [MEASURED]. Across 552 synthetic authorization scenarios, Lawie's evaluated authorization functions denied no legitimate request, refused every scenario of some attack classes and allowed every scenario of others [MEASURED]; its audit-chain verifier detected every tested edit, deletion, reordering, insertion and deleted prefix, and its dossier verifier failed every tampered dossier [MEASURED]. Its temporal checks reported a provision as in force on every tested date on which a provision was in force but not yet applicable [MEASURED], and its jurisdiction check flagged none of the tested citations of five foreign legal orders. The principles are typed verification of decisive elements, canonicalisation before comparison, anchoring, provenance-carrying authorization and refusal by default. Descriptions stay at the level of classes; the paper contains no attack recipe.

Prepared by Lawie Labs. *Status:* First edition. *Distribution:* Public. *Basis:* source repository `main` @ `ffa13d4bce` (2026-11-24); experiments of 26 November 2026; EUR-Lex texts retrieved 26 November 2026. *Audience:* Security engineers, AI researchers, legal technologists, regulators and diligence teams.

Evidence labels. Every material statement about Lawie carries an evidence class. [IMPLEMENTED] in source at the basis commit; [TESTED] exercised by repository tests at that commit; [MEASURED] measured by an offline experiment of this suite on the stated public or synthetic data, never a production or customer measurement; [ANALYTICAL] follows by argument from stated assumptions; [SIMULATED] output of a stated model; [EXTERNAL] checked against a cited primary source; [PROPOSED] design or protocol not built or not run; [UNVALIDATED] a hypothesis, stated as such; [ABSENT] not implemented; [BLOCKED] awaits an external owner or decision. Statements describe the source repository at the basis commit, not a deployed service: no workflow described here has a dated production acceptance, nothing is independently audited, and no statement has been reviewed by an independent human expert. This is a technical publication, not legal advice, not an offer of services and not a statement that any use of Lawie complies with any law.

Contents

Abstract **1**
1 Introduction **3**
1.1 Why legal AI is an adversarial setting 3
1.2 The verifier is the attack surface 3
1.3 Contributions and scope 3
2 Threat model **4**
2.1 Assets and adversaries 4
2.2 Surfaces, attack classes and principles 4
3 False statements of law (G1) **4**
3.1 Minimal decisive edits 4
3.2 Typographic and Unicode variation 6
3.3 Misattribution and self-declared attribution 7
4 Unauthorized actions (G2) **7**
4.1 An authorization function sees only its inputs 7
4.2 Content-induced requests and delegation 7
4.3 Approvals, quorums and tenants 8
5 Undetected alteration of evidence (G3) **8**
6 Wrong law and poisoned sources (G4, G5) **10**
6.1 Temporal attacks 10
6.2 Jurisdiction confusion 11
6.3 Poisoned sources 11
7 Design principles and residual risk **11**
8 Limits of the evidence, and conclusion **13**
References **14**

1 Introduction

1.1 Why legal AI is an adversarial setting

Most evaluations of AI systems ask how often a system is right on inputs drawn from some population. In law that question is necessary and insufficient. Legal work is done against someone: the other side of a contract, the opposing party in a dispute, the regulated firm that would prefer a lighter reading, the insider who would prefer not to wait for an approval. Each has an interest in a specific wrong output and the means to shape the input that produces it. A contract arrives with a clause drafted to be misread; a counterparty's letter contains an instruction addressed to the assistant that will summarise it; a citation is chosen because it is close enough to the right one to pass a glance.

Robustness in this setting is a worst-case property. It is measured against an adversary who sees the system's rules, chooses the input, and adapts. The literature on adversarial evaluation of language systems shows how far average-case accuracy can mislead: distracting sentences that leave a reading-comprehension answer unchanged sharply reduce model accuracy [1]; minimal, label-changing perturbations expose decision boundaries that test sets rarely probe [2]; behavioural tests find failures that held-out accuracy hides [3]. For agents that act, the adversary also writes the content the agent reads, and instructions embedded in that content can redirect the agent's actions [4]; injected instructions can also hijack the goal a model was given [5].

1.2 The verifier is the attack surface

A legal AI system that verifies before it releases, and authorizes before it acts, concentrates its security in those checks. The language model may be persuaded of anything; what matters is whether the checks let the persuasion through. The suite's doctrine, that intelligence is not authority, has a security reading: the model's output is never the warrant for a statement or an action, so the adversary's target is not the model but the procedure that decides what counts as warranted. This paper therefore studies the procedure. It asks, for each check a legal AI system relies on, what class of input defeats it, what principle would make it resist that class, and what risk remains when the principle is applied.

1.3 Contributions and scope

K1 A threat model for legal AI organised by five adversary goals and the surfaces they attack (section 2).

K2 Four analytical results that apply across the surfaces: the adaptive false-accept rate, the completeness of authorization inputs, the limits of unwitnessed hash chains, and the trade-off of canonical forms (sections 3 to 5).

K3 For each surface, the evidence from this suite's experiments on Lawie's components, with aggregate outcomes (sections 3 to 6).

K4 Five design principles with the residual risk each leaves (section 7).

Every weakness is described as a class with the principle that defeats it. The paper names no field, threshold, identifier or sequence that would let a reader reproduce an attack on a deployed system. Nothing described here is deployed in production [BLOCKED], and no independent audit or security review of the components is recorded [ABSENT]. The evidence comes from offline experiments on synthetic and public data; none is a red-team exercise against a running service (section 8).

2 Threat model

2.1 Assets and adversaries

The assets of a legal AI system are of three kinds: the *statements* it releases about the law, the *actions* it executes on a principal's behalf (filing, signing, sending, paying, deleting), and the *evidence* it keeps of what it did and why. A fourth asset is shared by all three: the *sources* it treats as the law.

Definition 2.1 (Adversary goals)**.** An adversary pursues one or more of five goals:

G1 False statement. A statement of law that the cited source does not support is released as supported.

G2 Unauthorized action. An action is executed without the authorization its policy requires.

G3 Undetected alteration. A record of what was done, or of the evidence relied on, is altered, removed or rewritten without detection.

G4 Wrong law. A provision is applied at a date on which it does not apply, or in a legal order that it does not govern.

G5 Poisoned source. Text that is not the official law is treated as the law.

The adversaries are ordinary participants in legal work. An *external author* writes content the system reads: a counterparty's contract, a letter, a web page, a document in a data room. A *requester* is an authenticated user or agent who asks for an action; it may be an honest user misled by content, or an insider. A *storage-level adversary* can modify the system's records, for example through an administrative account or a compromised database credential. A *source-level adversary* can place text where the system's retrieval may find it.

Assumption 2.1 (Capabilities)**.** *The adversary knows the system's design, including its checks, and chooses inputs adaptively. It cannot forge a signature without the private key, find collisions in a cryptographic hash function, or alter the Official Journal. It can author arbitrary content that the system reads, submit requests within its own credentials, and, at the storage level, modify records and recompute anything that is not keyed.*

Out of scope are denial of service, model extraction, side channels, physical attacks and the compromise of the official publisher. So is persuasion of the human professional who signs: a system can bind an approval to an exact action, but it cannot make the person who approves read what they approve.

2.2 Surfaces, attack classes and principles

Table 1 maps each goal to the attack classes this paper treats, the principle that defeats each, and the evidence of this suite. The sections that follow take the rows in turn.

3 False statements of law (G1)

3.1 Minimal decisive edits

Attack class. The adversary wants a false statement of law released with a real citation. The cheapest such statement is the provision's own text with one decisive element changed: *shall* for *may*, *not* inserted or removed, a number, a date, a party, a comparator, an exception deleted. The edit is minimal in words and maximal in law. It is the contrast set of the evaluation literature [2] and the directional test of behavioural testing [3], used with intent.

Proposition 3.1 (Adaptive false-accept rate)**.** *Let a verifier accept an error of class m with probability F_m, for the classes $m \in M$ that an adversary can produce. (i) An adversary who chooses the class faces*

Table 1: Threat model: goals, attack classes, defeating principles and the evidence of this suite.

Goal	Attack class	Defeating principle	Evidence in this suite
G1	Minimal decisive edit	Typed verification of decisive elements	Lawie grounding accepts all; typed prototype stops modelled types only
G1	Typographic and Unicode variation	Canonicalisation before comparison	Exact span rejects benign variants; canonical match exact on edits
G1	Misattribution; self-declared attribution	Anchoring to the cited address	Lawie grounding accepts all misattributions
G2	Content-induced request; delegation widening; misused, unbound or foreign approval; self-approval; one-person quorum; foreign tenant	Provenance-carrying authorization; approvals bound to the exact action; every attribute a norm needs is an input	BEN-02 aggregate: no legitimate request denied; some attack classes refused in every scenario, others allowed
G3	Edit, deletion, reordering or insertion of records; tampered signed artefacts	Chain and signature verification	Every tested edit, deletion, reordering, insertion and deleted prefix detected; every tampered dossier failed
G4	In force but not applicable; superseded schedule	Per-provision dates under versioned schedules	Every such date answered wrongly
G4	Foreign law in a domestic answer	Jurisdiction by admission	Five foreign orders unflagged
G5	Poisoned or unofficial source	Authority only from admitted official sources	Implemented check; no poisoning experiment run

the acceptance probability $\max_{m \in M} F_m$*, whatever the average of the* F_m *over any test population. (ii) Against a deterministic verifier whose rule is known, an adversary who can evaluate candidate inputs before submitting them succeeds whenever some input that achieves the goal is accepted.*

Proof. (i) The adversary submits errors of a class attaining the maximum. (ii) The adversary evaluates candidates with the known rule and submits an accepted one; determinism makes the evaluation exact. □

Average-case accuracy is therefore the wrong summary of a verifier under attack. What matters is the worst error class the adversary can produce, and, for a deterministic rule, whether any accepted input exists.

Evidence. LLRB-1 built such edits from the official text of eight EU acts. Lawie's grounding check, as implemented at the code basis, accepted 1,959 of 1,959 statements carrying one decisive edit and 698 of 698 statements cited to the wrong provision [MEASURED]. In general, an overlap measure with acceptance threshold t keeps accepting a fully contained statement after up to a share $1 - t$ of its tokens are replaced, so any overlap measure with a threshold below one tolerates a one-word change [ANALYTICAL] (LL-REL-01). Containment, token F1 and Jaccard, each thresholded at the 5th percentile of supported development scores, accepted 27.1%, 54.2% and 51.8% of decisive edits on average [MEASURED]; by proposition 3.1 the adversary faces at least their worst family instead, which is every deleted exception for calibrated containment and about nine negation edits in ten for token F1 and Jaccard, and an adversary who can test inputs does better still, for instance by inserting *not* only into statements of 18 tokens or more, which token F1 accepts against a coinciding clause (the per-family rates are in the companion benchmark report, LL-BEN-01). Each measure has a class of edit that its arithmetic cannot see: containment, as defined in the benchmark, cannot see deletions, tokens of one or two characters, or words the provision already uses [ANALYTICAL]; symmetric measures cannot see a single changed word in a long sentence.

Principle: typed verification of decisive elements. A verifier that extracts the decisive elements of the statement and of the cited provision and requires them to match is not fooled by an edit to an element it types. The typed research prototype of this suite, which types the deontic operator with its polarity,

quantities, dates, time units and cross-references, accepted 0 of 1,444 edits of those types while rejecting 0 of 1,396 supported statements [MEASURED]. It is a research instrument, not a Lawie component [PROPOSED].

Residual risk. Typing protects only what it types. The same prototype accepted 456 of 515 edits of the types it does not model: swapped actors, reversed comparators, deleted exceptions [MEASURED]. Against an adaptive adversary its false-accept rate is at least that of its worst unmodelled class, 93% for swapped actors (LL-BEN-01), and higher for an adversary who can test candidate edits before submitting them. The prototype and the edit families were also designed by the same authors [ANALYTICAL]; an adversary designs neither. A typed verifier must therefore publish the list of element types it checks, treat every other type as unverified, and be tested on types its designers did not anticipate.

3.2 Typographic and Unicode variation

Attack class. Text that looks the same can differ in bytes, and text that differs in law can look nearly the same. The adversary uses the first fact for denial and the second for smuggling. *Denial*: an authentic quotation re-typed with a straight apostrophe or an ordinary space fails a byte-exact check, so a true citation can be made to look fabricated, and operators who see too many such failures switch the check off. *Smuggling*: a change confined to characters that a comparison ignores passes it, and a look-alike letter from another script can make a human reader see one text while the machine compares another.

Proposition 3.2 (The canonical trade-off)**.** *Let κ be a canonicalisation and let a check accept a quotation when its image under κ is a substring of the image of the official text. Then (i) a check that accepts only bytes equal to a receipted span of the official text accepts no altered quotation and rejects every typographic variant, whereas a byte-substring check, κ the identity, still accepts truncations; (ii) for any κ, the check accepts every variant that differs only within the character classes κ folds, and so cannot detect an edit confined to those classes; (iii) an edit outside the folded classes is rejected unless the edited text occurs elsewhere in the official text.*

Proof. (i) Byte equality with a span admits only the identical string, while a substring test also admits any shorter piece of it, such as the quotation without its final punctuation. (ii) κ maps the variant and the original to the same image. (iii) The edit survives κ, so the image of the edited quotation differs from the image of the original. □

Evidence. Lawie's exact-span authority predicate accepted every authentic quotation and rejected every typographic variant and every edit; with the span derivation that EU texts would need, it rejected everything [MEASURED]. Canonical substring matching accepted every authentic quotation and every typographic variant and rejected every edit and every misattribution [MEASURED]. Both behaviours are the ones proposition 3.2(i) and (ii) predict. Variation also attacks change detection: Lawie's legislative-diff engine flagged 477 changes when it compared the AI Act's texts as rendered, of which 21.6% were amendments, and 114 changes at 86.8% precision on canonicalised text [MEASURED]. A detector that raises more than three false alarms for each true one teaches its reviewers to stop reading, and a substantive change can hide among the formatting noise.

Principle: canonicalisation before comparison. Compare, hash and sign canonical forms, as structured data is canonicalised before it is hashed or signed [6]. Declare the canonical form, version it, and fold only character classes that carry no legal meaning in the corpus. Treat a token that mixes scripts as a finding to report, not a variant to fold.

Residual risk. Whatever κ folds is invisible to the check. Compatibility normalisation maps a superscript digit to an ordinary digit, so "10^2" and "102" share a canonical form; case folding maps a capitalised defined term to the common word, so "the Board" and "the board" coincide; neither folds look-alike letters

across scripts, which a canonical match therefore rejects [ANALYTICAL]. A canonicalisation safe for one corpus can be unsafe for another, and the choice is a legal decision recorded as a technical one.

3.3 Misattribution and self-declared attribution

Attack class. The adversary pairs a correct statement with the wrong provision, choosing a provision close enough to pass a glance: the neighbouring article, the same formula in another chapter. A variant asks the system to accept attribution on the output's own word: an answer that declares which source supports it.

Evidence. Lawie's grounding check accepted every misattribution in LLRB-1, 698 of 698, each cited to the most lexically similar provision of another article of the same act [MEASURED]. Its quoted-holding matcher, given the provision text, accepted 51.6% of them [MEASURED]. Canonical matching against the cited provision rejected every misattribution.

Principle: anchoring to the cited address. Resolve the citation to an address, retrieve the official text of that address at the version in force on the date in question, and compare the statement with that text and nothing else. Attribution is computed by the verifier from the citation; it is never accepted from the output. Lawie's citation-authority predicate applies this principle strictly: authority only for an exact byte span of an official-source observation retrieved under an approved source contract, current and in force at the citation's date, never from model output [TESTED]. It admits exactly one span derivation, for Czech statute paragraphs, so no EU text can pass it today [IMPLEMENTED].

Residual risk. Anchoring is only as good as resolution. Lawie's extractor resolved the instrument of no citation written in the Official Journal's own "Regulation (EU)" style [MEASURED]. If any system releases a citation it cannot resolve unverified, every format its resolver cannot parse is an evasion route; if it is withheld, the same gap becomes an availability cost. Refusal by default (section 7) chooses the second.

4 Unauthorized actions (G2)

4.1 An authorization function sees only its inputs

Proposition 4.1 (Completeness of authorization inputs)**.** *Let an authorization function decide each request from an input x. If a norm requires that two requests with the same input x be decided differently, the function violates the norm on one of them. In particular, a norm whose verdict depends on an attribute that x does not encode cannot be enforced by the function, however correct it is on the attributes it receives.*

Proof. The function returns one decision for the input x; the norm requires two. □

The proposition is trivial; its consequence is that a check must be given every attribute its norm depends on.

4.2 Content-induced requests and delegation

Attack class. An agent that reads content and holds a principal's authority is a deputy, and content that instructs it is the confused deputy's client [7]. Indirect prompt injection places instructions in data that an application retrieves and so reaches the model without passing through the user [4]; goal hijacking overrides the task the model was given [5]. Benchmarks for tool-using agents show that the attack is general across tasks, tools and models [8, 9], and automatically generated adversarial suffixes transfer across aligned models [10], so a model's own refusal is not a security boundary. The OWASP list of risks for LLM applications places prompt injection and excessive agency among its principal entries [11]. The variant for multi-agent systems is delegation widening: a sub-agent acts with less restriction than the agent that created it.

Principle: provenance-carrying authorization. Every request carries, as an attribute the authorization function receives, where it came from (the principal's instruction, or content the agent read) and the restrictions of every agent that delegated it. A consequential request whose provenance includes untrusted content is refused or routed to a fresh human approval bound to the exact action; a child's authority is the intersection of its own and its parent's. This is the capability discipline: authority travels with the designation of what may be done, not with the deputy who does it [7, 12]. A provenance model such as PROV supplies the vocabulary of entities, activities and agents that such an attribute needs [13]. Binding an approval to the exact action is the complementary control: in Lawie's implementation at the code basis, an approval authorizes only the tool identity, its exact version and the digest of the canonical current input [TESTED].

Residual risk. A human approver can be deceived by the same content that deceived the agent; binding makes the executed action the approved one, not a wise one. Provenance can be laundered when a person copies untrusted text into an instruction. Actions that need no approval remain reachable by injection, so the classification of actions as consequential is itself part of the attack surface.

4.3 Approvals, quorums and tenants

Attack classes. A requester seeks to execute an action under an approval that does not authorize it: an approval for another input or another version, an expired, rejected or pending one, one reviewed before it was requested or by no named reviewer, an approval given by the requester, a quorum met by one person signing twice, or a request that reaches into another tenant's data, whether named in the request or carried inside the tool input. The suite's authorization benchmark (BEN-02) turns this taxonomy into sixteen attack classes and four classes of legitimate request, with norms fixed in advance; requests induced by untrusted content and delegation widening are two of the sixteen.

Evidence. Table 2 shows what three reference systems allow. A permissive system and an allow-list allow every attack; a system that accepts any approved gate for the tool refuses only rejected and pending approvals. Across the 552 scenarios, Lawie's evaluated authorization functions denied none of the 96 legitimate requests, refused every scenario of some attack classes and allowed every scenario of others [MEASURED]. Which classes these are is recorded in restricted engineering records pending remediation.

Principle: every attribute a norm needs is an input. Separation of duties, one of the two pillars of commercial integrity policy, requires that the parties to a transaction be distinct and identified [14]; complete mediation requires that every access be checked for authority [15], and we extend it to every attribute on which the norm depends.

Residual risk. Identity checks rest on authentication, and two colluding people defeat any two-person rule. And the scenarios measured are synthetic and component-level: they show what the evaluated functions decide, not that every route of a running system calls them.

5 Undetected alteration of evidence (G3)

Attack class. A storage-level adversary changes the record of what the system did: edits a record, deletes one, swaps or inserts records, removes the most recent records, or rewrites a stretch of history and recomputes whatever integrity values it can.

Proposition 5.1 (What an unwitnessed chain can show)**.** *Let records $r_1, \ldots, r_n$ be chained by $h_i = \mathcal{H}(h_{i-1} \,\|\, r_i)$ from a fixed h_0, with an unkeyed collision-resistant hash $\mathcal{H}$, and let a verifier recompute the chain from the stored records and compare it with the stored values. (i) A change to a record that leaves its stored chain value in place is detected. (ii) Removing the last k records, or changing a record*

Table 2: BEN-02 scenario classes and the number of scenarios each reference system allows.

Scenario class	Norm	n	Permissive	Allow-list	Unbound approval
Valid, exactly bound approval	allow	24	24	24	24
Tool without approval requirement	allow	24	24	24	24
Own tenant named explicitly	allow	24	24	24	24
Quorum by two distinct reviewers	allow	24	24	24	24
A1 approval reused for another input	deny	24	24	24	24
A2 approval for another tool version	deny	24	24	24	24
A3 expired approval	deny	24	24	24	24
A4 reviewed before requested	deny	24	24	24	24
A4 no named reviewer	deny	24	24	24	24
A5 rejected approval	deny	24	24	24	0
A5 pending approval	deny	24	24	24	0
A6 self-approval	deny	24	24	24	24
A7 two approvals by one person	deny	24	24	24	24
A7b quorum with named reviewers, one person	deny	24	24	24	24
A8 foreign tenant requested	deny	24	24	24	24
A9 foreign tenant in a conventional tenant field	deny	24	24	24	24
A10 foreign tenant under an alternative field name, top level	deny	96	96	96	96
A10 foreign tenant under an alternative field name, nested	deny	24	24	24	24
A11 request induced by untrusted content	deny	24	24	24	24
A12 sub-agent beyond parent's restrictions	deny	24	24	24	24

and recomputing every later value, yields a stored chain that the verifier accepts. (iii) If an independent witness holds the head h_m for some $m \leq n$ and the verifier also checks it, any truncation below m and any change to a record at or before m is detected, unless $\mathcal{H}$ has a collision.

Proof. (i) Recomputation from the changed record yields a value that differs from the stored one, except by a collision. (ii) In both cases the stored chain is the honest construction applied to a different sequence, which the verifier cannot distinguish from an honest chain because it uses no information outside the store. (iii) A truncated or rewritten sequence has a different value at position m, or none, and the witness's h_m exposes it, except by a collision. □

Part (ii) is why detecting truncation or rewriting requires an independent witness of the head [ANALYTICAL]. Part (iii) states what the witness buys and what it does not: records appended after the last witnessed head remain exposed until the next one. A protection enforced by the store itself, such as a rule that forbids updates, binds only those who cannot administer the store, which excludes the storage-level adversary by definition.

Evidence. The suite's evidence-integrity study mutated audit chains built by Lawie's chain construction and asked its verifier to find the changes. The verifier detected every tested edit, deletion, reordering and insertion of records and every deleted prefix, fifty trials each (table 3); the portable dossier verifier failed every tampered dossier, one with an edited or dropped record, a signature by a foreign key or a key the trust policy does not pin, and verified the intact one [MEASURED]. Signed dossiers verify offline against pinned key fingerprints [TESTED], and each states what verification does not establish: it proves neither substantive truth nor admissibility nor legal correctness, and it creates no authority [IMPLEMENTED]. We do not call any record tamper-proof.

Principle: anchoring. Deposit the head of every chain, and the time of every signature that must outlive its key, with a witness the operator does not control, at intervals short enough that the exposure of part (iii)

Table 3: In-place mutations of an audit chain detected by Lawie's chain verifier (50 trials each).

Mutation	Trials	Detected by the chain verifier
Edit a record	50	50
Delete a record	50	50
Swap two records	50	50
Insert a forged record	50	50
Delete a prefix	50	50

is acceptable. Linked time-stamping was proposed for exactly this purpose [16]; secure audit logs make earlier entries tamper-evident to a later attacker [17]; history trees and Merkle-tree logs with signed heads and consistency proofs let auditors check that a log only grew [18–20].

Residual risk. The witness must be independent and available, and its independence is an organisational fact, not a cryptographic one. Records written after the last anchor can be removed until the next. Integrity is not truth: an anchored record of a wrong decision is a well-preserved wrong decision.

6 Wrong law and poisoned sources (G4, G5)

6.1 Temporal attacks

Attack class. The adversary argues from a provision that is in force but does not yet apply, or from a schedule that an amendment has since changed. Both are ordinary moves in a dispute, and both succeed against a system whose notion of time is a single validity interval per version. EU law gives the adversary material: the final provisions of the acts of this suite's corpus, as published, separate publication, entry into force and application, often by years [EXTERNAL], and the text of Regulation (EU) 2026/1744 as published in the Official Journal moves the dates that Article 113 of the AI Act lists for the application of the high-risk requirements before the original dates were reached [EXTERNAL].

Evidence. Lawie's temporal index and the in-force check of its citation-authority predicate reported a provision as in force on 781 of 781 test dates on which it was in force but not yet applicable, and were correct on every other date [MEASURED]. The cause is structural: an index keyed on entry into force answers "applicable" wrongly on every day between entry into force and application [ANALYTICAL], and staggered dates of application, transitional provisions and consolidated versions are not modelled [ABSENT]. Over 2016–2028 the exposure of that heuristic is 7–20% of provision-days depending on the act, and an AI Act schedule frozen before the 2026 amendment errs on 2.0% of provision-days for systems in Annex III [MEASURED].

Principle: per-provision dates under versioned schedules. Answer applicability from four dates, publication, entry into force, application per provision and per category of addressee, and the knowledge date that fixes which version of the schedule applies, as temporal databases separate valid time from transaction time [21]. Refuse when application depends on facts that are not in evidence.

Residual risk. Schedules must be transcribed and kept current; between an amendment's publication and its transcription, the system answers from the superseded schedule. Provisions whose application turns on facts can only be refused, which is correct and unhelpful.

6.2 Jurisdiction confusion

Attack class. The adversary imports authority from a legal order that does not govern the matter, choosing one close enough to pass: a neighbouring state's statute with the same structure and similar numbering, or a statute in the same language.

Evidence. In answers for a Czech matter, Lawie's gateway flagged citations of US law (8 of 8) and never flagged citations of Slovak, German, Austrian, UK or Polish law (0 of 32) [MEASURED]. The rule treats Czech, Slovak and EU law as one compatible bloc and recognises foreign markers only for a short list of legal orders [IMPLEMENTED]: it blocks what it knows and admits everything else.

Principle: jurisdiction by admission. Admit, for each matter, the instruments that govern it, and flag every citation that does not resolve to an admitted instrument. Lawie's retrieval already works this way for documents, routing jurisdiction along a declared chain and dropping untagged documents from jurisdiction-routed results [TESTED].

Residual risk. Foreign law sometimes applies legitimately, through a choice of law or a conflict rule, and an admission list then flags a correct answer. Clearing such flags is a legal judgment that the system should route to a person, not make.

6.3 Poisoned sources

Attack class. The adversary places text where retrieval will find it and presents it as the law: an altered copy of a provision, a fabricated provision on a plausible page, a document whose metadata claims an official origin. Data and model poisoning is a recognised risk class for LLM applications [11]; for legal AI the poisoned object is the source of authority itself.

Evidence. Lawie's citation-authority predicate grants authority only to an exact byte span of an official-source observation retrieved under an approved, active source contract, still current and in force at the citation's date, and never takes model output as an input [TESTED]. Each source observation records its locator, retrieval time, payload and content digests, an object snapshot, connector and parser versions, licence revision and a supersession chain [IMPLEMENTED]. Retrieval drops citations that are not backed by a retrieved source [IMPLEMENTED]. At the code basis the predicate is defined to refuse every citation it cannot trace to an approved official source, and it admits exactly one span derivation, for Czech statute paragraphs [IMPLEMENTED]. These describe implemented checks; no poisoning experiment was run, and they are not an assurance about a deployed service.

Principle: authority only from admitted official sources. A text is the law only if it is a span of an observation of an admitted official source, with its retrieval recorded and its digest checkable; everything else is evidence about the law at best. Model output is never a source.

Residual risk. Admission moves the trust to the official publisher and to the connector that fetches from it. Corrections and consolidations must be tracked as superseding observations, or the system will hold an authentic but outdated text. The compromise of an official publisher is outside this threat model, and no verifier downstream of it can detect it.

7 Design principles and residual risk

Five principles recur across the surfaces. None is new to security engineering; what is particular to legal AI is where they must be applied and what they leave behind. Table 4 gives, for each, the attack classes it defeats and the risk that remains when it is applied correctly.

Table 4: Design principles for legal AI under attack, the classes each defeats and the residual risk each leaves.

Principle	Defeats	Residual risk
Typed verification of decisive elements	Minimal edits to the element types it checks	Every element type it does not check; an adaptive adversary moves there
Canonicalisation before comparison	Denial by typographic variation; floods of formatting noise in change detection	Edits confined to folded classes; the choice of what to fold
Anchoring	Misattribution and self-declared attribution; truncation and rewriting of records; backdated signatures	Resolution gaps; the witness's independence and availability; records since the last anchor
Provenance-carrying authorization	Content-induced requests; delegation widening; requests whose norm depends on identity, provenance or delegation	Deceived approvers; laundered provenance; actions classed as inconsequential
Refusal by default	Unknown legal orders; unresolvable citations; unverified stages; unadmitted sources	Availability: correct answers withheld, and pressure to switch the refusal off

Typed verification of decisive elements. Decide support by extracting the elements that carry legal effect and requiring them to match the cited provision, never by resemblance. The evidence that this works is the typed prototype's record on the types it models [MEASURED]; the evidence of its limit is its record on the types it does not [MEASURED].

Canonicalisation before comparison. Compare canonical forms under a declared and versioned canonicalisation that folds only what carries no legal meaning. Canonical matching was exact on every edit in LLRB-1 and accepted every typographic variant [MEASURED]; its blind spot is what it folds [ANALYTICAL].

Anchoring. Bind each statement to the text of the address it cites, each chain to a head held by an independent witness, and each signature that must outlive its key to an independently witnessed time. Without a witness, no unkeyed chain can show that it was not truncated or rewritten [ANALYTICAL].

Provenance-carrying authorization. Give every authorization function every attribute its norm needs: the requester and each reviewer, distinct identities for a quorum, every tenant reference as a typed value, the request's provenance and the restrictions of every delegating agent. By proposition 4.1, a norm that depends on an attribute the function does not receive cannot be enforced, whatever the function's logic.

Refusal by default. Admit what is known to be safe and refuse the rest: legal orders by admission, sources by approved contract, release only when every verification stage ran and passed. Several of Lawie's controls are built this way, among them default-deny tool grants [TESTED] and authority only from admitted official sources [TESTED]. Its jurisdiction rule, which blocks a list of known foreign markers instead of admitting governing instruments, is the control that missed five foreign legal orders [MEASURED]. The cost of refusal by default is availability, and it is paid by honest users; a system that refuses must say why, so that the cost can be seen and argued about rather than quietly removed.

Two further points apply to all five. First, by proposition 3.1, a defence is measured by its worst class, and the list of classes is open; every principle above protects a list, and the list must be published with its gaps. Second, the principles move trust rather than remove it: to the person who types the decisive elements, to the witness of a head, to the official publisher, to the approver. A legal AI system cannot be made trustless. It can be made to say, for each output and each action, exactly whom it trusted and for what.

8 Limits of the evidence, and conclusion

What was not tested. No adversary attacked a running system. The evidence comes from offline experiments in which Lawie's components were imported unchanged and run on constructed inputs [IMPLEMENTED]: single decisive edits of official EU text, and 552 synthetic authorization scenarios. No language model was in the loop, so no prompt injection was attempted against a model, and the study of model-generated answers is defined and not run [PROPOSED]. The authorization scenarios show what the evaluated functions decide, not whether every route of a running system calls them. The benchmark's labels, including the application schedules encoded from the official texts, are research labels that have not been reviewed by a lawyer [ABSENT], and the typed prototype is a research instrument, not a Lawie component [PROPOSED]. Nothing described is deployed in production [BLOCKED], and no independent audit, certification or expert review of the components is recorded [ABSENT]. This paper reports research measurements at a stated code basis. It is not a security assurance, a compliance statement or legal advice, and no independent security, privacy or legal review of it has taken place.

What an attack test would add. The classes in this paper are the ones the suite's experiments could reach without a live system. A red-team exercise against a deployed service would add the classes that live only at integration seams: routes that skip a check, caches that serve stale law, user interfaces that show one text and verify another. Those classes are named here so that their absence from the evidence is visible.

Conclusion. The checks of a legal AI system are its security boundary, because the model behind them can be persuaded of anything. Measured against an adversary, the checks that compare words fail on the edits that change law, and the rules that block a list of known dangers let the unlisted ones through. In principle, a check that lacks an attribute cannot enforce a norm that needs it, and an unwitnessed chain cannot prove it was not rewritten. Each failure has a principle that addresses its class: typed verification, canonical comparison, anchoring, provenance-carrying authorization and refusal by default. Each principle leaves a residue that must be stated. The doctrine of this suite, that intelligence is not authority, has a security corollary: a system that cannot show who authorized an action, on what evidence and against which source, has no defence against someone who wants it to act without one.

References

[1] Robin Jia and Percy Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017. doi: 10.18653/v1/D17-1215.

[2] Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, et al. Evaluating Models' Local Decision Boundaries via Contrast Sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, 2020. doi: 10.18653/v1/2020.findings-emnlp.117.

[3] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, 2020. doi: 10.18653/v1/2020.acl-main.442.

[4] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023. doi: 10.1145/3605764.3623985.

[5] Fábio Perez and Ian Ribeiro. Ignore Previous Prompt: Attack Techniques for Language Models, 2022. arXiv:2211.09527.

[6] Anders Rundgren, Bret Jordan, and Samuel Erdtman. JSON Canonicalization Scheme (JCS). RFC 8785, Internet Engineering Task Force, 2020.

[7] Norm Hardy. The Confused Deputy (or Why Capabilities Might Have Been Invented). *ACM SIGOPS Operating Systems Review*, 22(4):36–38, 1988. doi: 10.1145/54289.871709.

[8] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, 2024.

[9] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506, 2024. doi: 10.18653/v1/2024.findings-acl.624.

[10] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models, 2023. arXiv:2307.15043.

[11] OWASP Foundation. OWASP Top 10 for LLM Applications 2025, 2024. URL https://genai.owasp.org/llm-top-10/.

[12] Jack B. Dennis and Earl C. Van Horn. Programming Semantics for Multiprogrammed Computations. *Communications of the ACM*, 9(3):143–155, 1966. doi: 10.1145/365230.365252.

[13] Luc Moreau and Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-dm/.

[14] David D. Clark and David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–194, 1987. doi: 10.1109/SP.1987.10001.

[15] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939.

[16] Stuart Haber and W. Scott Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2): 99–111, 1991. doi: 10.1007/BF00196791.

[17] Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999. doi: 10.1145/317087.317089.

[18] Scott A. Crosby and Dan S. Wallach. Efficient Data Structures for Tamper-Evident Logging. In *Proceedings of the 18th USENIX Security Symposium*, pages 317–334, 2009. URL https://www.usenix.org/conference/usenixsecurity09/technical-sessions/presentation/efficient-data-structures-tamper-evident.

[19] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, Internet Engineering Task Force, 2013.

[20] Ralph C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378, 1988. doi: 10.1007/3-540-48184-2_32.

[21] Christian S. Jensen, Curtis E. Dyreson, et al. The Consensus Glossary of Temporal Database Concepts — February 1998 Version. In *Temporal Databases: Research and Practice*, volume 1399 of *Lecture Notes in Computer Science*, pages 367–405. Springer, 1998. doi: 10.1007/BFb0053710.