A Sovereign, Reproducible Trust Layer for LLM Systems: Two-Number Prompt-Injection Detection with Post-Quantum, Browser-Verifiable Audit Receipts

Hagen Schmidt
DESTILL.ai
Vienna, Austria
IP@destill.ai

***Abstract*—Prompt-injection guardrails are commonly reported with a single fused metric, evaluated on private test sets, and shipped as cloud services that exfiltrate the very prompts they screen. We present an on-premise, CPU-first trust layer for Large Language Model (LLM) systems that combines two genuinely separable concerns. First, AEGIS is a cascaded injection guardrail whose first layer is a pure-TypeScript Aho-Corasick finite-state automaton giving linear-time, sub-millisecond keyword detection (median 1.73 ms, 95th percentile 6.95 ms), backed by an on-premise Llama-Guard-3:1b content-safety stage. We report a deliberately two-number scorecard, in which injection detection and content safety are never fused, measured only on public corpora (hackerprompt, jailbreakhub, StrongREJECT, OR-Bench): injection detection of 97.7 percent true-positive rate at 4.0 percent false-positive rate (176 adversarial / 150 benign), and content safety with an F-measure of 0.814 at 79 percent true-positive rate and 10 percent false-positive rate. Second, LEDGER emits tamper-evident receipts signed with FIPS-204 ML-DSA-65 (real 3309-byte signatures, not stubs), anchored in an RFC-6962 transparency log with RFC-3161 timestamps, and exported as W3C Verifiable Credentials with scoped EU AI Act mappings; receipts verify offline in a web browser against a did:web key. Unit tests confirm detection of action-replay, metric-inflation, and signature forgery. Beyond these two subsystems, we apply the same demonstrate-not-assert discipline to a defense usually reported only by assertion: we measure whether a deception (honeypot) layer is statistically fingerprintable under four attacker models with a same-population control, finding template decoys trivially fingerprintable (held-out stylometry AUC 1.000) while a retrieval-from-real-deflection variant reduces this to 0.636 at pure-CPU serving, with a residual content tell we disclose rather than hide. We disclose measured gaps: narrative jailbreaks reach only 16 percent true-positive rate, key custody is software-only, and several auxiliary layers are unmeasured. We release the harness and a live evidence bundle for independent reproduction.**

***Index Terms*—prompt injection, LLM security, AI guardrails, post-quantum cryptography, verifiable credentials, reproducible evaluation, data sovereignty, EU AI Act**

I. Introduction

Generative LLMs are entering regulated, security-sensitive deployments faster than the mechanisms that are supposed to make them trustworthy. Three structural weaknesses recur in current guardrail practice.

Single-metric reporting. Injection defense and content safety are distinct problems—a perfectly benign-looking prompt can request harmful content, and a harmful instruction can be syntactically clean—yet they are routinely collapsed into one aggregate score, hiding which axis is weak.

Private test sets. Headline numbers are frequently produced on proprietary corpora, so no third party can reproduce them. In a safety context this is the opposite of what should be required.

Cloud screening. Many guardrails are cloud services: the prompt must leave the customer's premises to be screened, which is incompatible with data-residency obligations and adds a network round-trip to every request.

This paper describes a trust layer that addresses all three, built around a simple discipline: *claims must be demonstrable, not merely asserted*. Our contributions are:

1) A **two-number scorecard** that reports injection detection and content safety as separate axes, evaluated entirely on public corpora through a one-command reproducible harness (Section IV, V).
2) **AEGIS**, a sovereign, CPU-first cascade that blocks before GPU inference using a zero-dependency Aho-Corasick automaton for linear-time, sub-millisecond detection, degrading gracefully to CPU-only (Section III).
3) **LEDGER**, a tamper-evident audit substrate binding guardrail verdicts to FIPS-204 ML-DSA-65 post-quantum signatures over an RFC-6962 append-only Merkle log with RFC-3161 timestamps, exported as W3C Verifiable Credentials (Section VII).
4) A **zero-server, browser-verifiable evidence bundle**: an auditor offline-checks the signature and timestamp against a did:web key, never trusting the issuer's servers (Section VII).
5) An **adversarially honest evaluation**: measured failure modes are published as first-class results (Section VIII).
6) A **deception-indistinguishability methodology**: four attacker models with a same-population control that *measure* whether a honeypot decoy is fingerprintable—rather than asserting it is convincing—applied to a

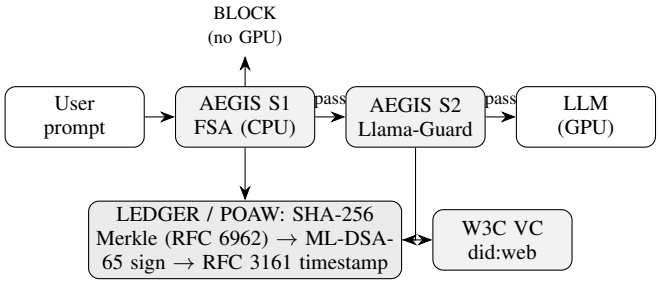

Fig. 1. CPU-first trust layer. AEGIS Stage 1 (linear-time automaton) and Stage 2 (on-premise content model) screen before any GPU inference; each verdict is bound into a post-quantum-signed, browser-verifiable receipt.

template and a retrieval-based generator (Section VI).

II. Background and Related Work

Cloud guardrails. Commercial moderation and injection-screening services require a network call per request and, by construction, receive the prompt being screened. AEGIS instead runs in the customer's process on commodity CPU.

Public injection corpora. We evaluate on openly available datasets: hackerprompt [8] and jailbreakhub [9] for injection attacks, StrongREJECT [10] and OR-Bench-toxic for harmful content, OR-Bench [11] for benign over-refusal prompts, and wildjailbreak as a hard narrative slice.

String matching. Layer 1 uses the classical Aho-Corasick automaton [1], which matches a fixed pattern set in time linear in the input length.

Transparency and timestamping. Our audit substrate uses an append-only Merkle log in the style of Certificate Transparency (RFC 6962) [2] and trusted timestamps per RFC 3161 [3].

Post-quantum signatures and credentials. Receipts are signed with ML-DSA-65 as standardized in FIPS 204 [4], exported as W3C Verifiable Credentials [5] and resolved through the did:web method [6]. Mappings target the record-keeping and logging duties of the EU AI Act [7].

Deception defenses. Honeypot guardrails serve a plausible decoy instead of a visible refusal so an attacker cannot tell a probe was detected [13]. Such work typically *asserts* the decoy is convincing; we instead measure its indistinguishability directly (Section VI).

III. System Architecture

The system is two decoupled subsystems sharing a common attestation format (Fig. 1).

A. Threat model

We assume an attacker who controls the natural-language input to the LLM (direct prompt injection) and may also plant adversarial text in retrieved or tool-returned content (indirect injection). The attacker's goals are to override the system prompt, exfiltrate hidden instructions or data, or coerce the model into producing harmful content. The attacker does *not* control the deployment host, the guardrail binary, or the issuer's signing key. Out of scope are model-weight extraction, side-channel attacks on the host, and adaptive attackers who can query the guardrail at scale to learn its decision boundary; the last is an important limitation we return to in Section VIII. A second adversary—a dishonest operator who wishes to *misrepresent* what the system did after the fact—is addressed by the audit substrate (Section VII): the threat there is tampering with recorded verdicts, which our cryptographic receipts are designed to make detectable.

B. AEGIS — injection and content guardrail

Stage 1 (injection). Threat patterns are pre-compiled at load time into a single Aho-Corasick automaton; input is scanned in one linear pass, and matches trigger regular-expression verification to suppress false positives. A cumulative severity threshold yields the block decision. An optional cascade adds orthogonal filters (Unicode normalization, entropy analysis, a semantic-intent classifier, conversation-drift tracking, and an authority detector) with an early-exit schedule so that clean prompts terminate after the cheapest layers. An optional, default-off *deception* mode can return a plausible decoy in place of a visible block; we measure its fingerprintability rather than assume it convincing (Section VI).

Stage 2 (content safety). A sovereign Llama-Guard-3:1b model runs on-premise via a local runtime (CPU or NPU), classifying the request against a hazard taxonomy. Crucially, Stage 2 is reported *separately* from Stage 1.

CPU-first. Both stages execute before any GPU inference, so a blocked prompt never reaches the model; the cascade always has a CPU fallback path.

C. LEDGER — the audit substrate

For each scanned action, LEDGER canonically serializes the event, hashes it (SHA-256), and incorporates the leaf into an append-only Merkle tree (RFC-6962 hashing, with inclusion and consistency proofs). A signed tree head is produced with ML-DSA-65 (FIPS 204). An RFC-3161 request over the receipt digest obtains an independent third-party timestamp. The receipt is exported as a W3C Verifiable Credential whose issuer key is published as a did:web document, and which carries an explicit, *scoped* EU AI Act mapping (Art. 12/19 logging; Annex IV record-keeping) together with a non-conformity disclaimer.

IV. The Two-Number Evaluation Methodology

We argue that a guardrail must report at least two numbers. Injection detection answers "is the user trying to subvert the system prompt or tooling?"; content safety answers "is the request itself harmful?". The two failure surfaces are orthogonal, so a single fused F-measure is uninformative and, worse, lets a vendor hide a weak axis behind a strong one. We therefore (i) evaluate the axes on separate corpora, (ii) report each with its own operating point and sample sizes, and (iii) recommend that deployers treat the two verdicts as *independent* decisions rather than a compounded gate.

All corpora are public. The benign set is OR-Bench (which deliberately contains over-refusal-bait prompts, stressing the false-positive rate). Every reported value is produced by an open harness and traced to a dated JSON manifest.

V. Experimental Results

Table I gives the headline two-number scorecard and Stage-1 latency. Table II gives the per-source breakdown and the aggregate at scale. We deliberately place limit results in the *same* tables as headline results.

TABLE I
TWO-NUMBER SCORECARD (PUBLIC CORPORA)

Axis	TPR	FPR	n (adv/benign)
Injection (Stage 1)	97.7%	4.0%	176 / 150
Content safety (Stage 2)	79%	10%	100 / 150
Content F-measure: 0.814 (Llama-Guard-3:1b, on-prem)			
Stage-1 latency: 1.73 ms median, 6.95 ms 95th percentile (CPU)			

Injection: hackerprompt+jailbreakhub vs OR-Bench; config NO_PHI+L3, dated 2026-06-08 (single canonical operating point; see Sec. VIII).

TABLE II
INJECTION DETECTION PER SOURCE, AND AGGREGATE AT SCALE

Source	TPR	n
hackerprompt	72%	—
jailbreakhub	85.5%	—
wildjailbreak (narrative)	16.0%	200
Aggregate (broad mix)	**47.6%**	676 / 1500

Aggregate FPR 3.8%. The aggregate is dominated by the hard narrative slice; standard injection performs substantially better.

The honest reading is explicit: on standard injection corpora the Stage-1 automaton is strong (72–85.5% per source, 97.7% at the tuned operating point), but on *narrative/roleplay* jailbreaks it is weak (16%), which pulls the broad-mix aggregate down to 47.6%. We report both rather than the flattering subset alone.

VI. Measuring Deception-Defense Indistinguishability

A deception layer is only as strong as its *indistinguishability*: if a decoy is fingerprintable, the attacker discounts it and the defense collapses to an expensive block. Honeypot guardrails are nonetheless usually reported by *asserting* the decoy is convincing [13]. We instead measure fingerprintability under four attacker models, with a same-population control that guards against a discriminator which would trivially separate any two text sources.

We compare two decoy generators against genuine on-premise model deflections: (i) a fixed template pool, and (ii) a retrieval variant that serves, at pure-CPU inference time, a real model deflection drawn from an offline-precomputed cache—so decoys lie on the model's own output manifold without a model in the serving path (offline generation, zero *marginal* energy). Table III reports D1, presence of an embedded provenance watermark a byte-inspecting attacker can detect;

Input: credential C, issuer DID d, timestamp token τ
$K \leftarrow \mathrm{resolveDidWeb}(d)$ {fetch published ML-DSA-65 key}
$m \leftarrow \mathrm{canonicalize}(C \setminus \{proof\})$
$s_{ok} \leftarrow \mathrm{MLDSA65.verify}(K, m, C.proof)$
$r \leftarrow \mathrm{merkleRoot}(C.actions)$
$t_{ok} \leftarrow \tau.granted \wedge \mathrm{refs}(\tau, \mathrm{SHA256}(m))$
return $s_{ok} \wedge (r = C.merkleRoot) \wedge t_{ok}$

Fig. 2. Offline verification of a LEDGER receipt against the live did:web key.

D2, the verbatim-collision onset under repeated probing (pool enumeration); and D4, a held-out term-frequency nearest-neighbour stylometry classifier (split by template, watermark stripped), validated by a same-population control that must score ≈ 0.5.

TABLE III
DECEPTION-DEFENSE INDISTINGUISHABILITY (LOWER = HARDER TO FINGERPRINT)

Attacker model	Template	Retrieval
D1 watermark (byte inspection)	100%	none
D2 verbatim-collision onset	$\sim$13 probes	$\sim$1250 (10^6 cache)
D4 stylometry AUC (held-out)	1.000	0.636
Same-population control AUC: 0.443 (≈ 0.5, method valid)		
Retrieval residual content-relevance tell: AUC 0.764		

The template pool is trivially fingerprintable: a 100% byte-watermark, a 40-string pool that repeats verbatim within $\sim$13 probes, and a held-out stylometry AUC of 1.000. The retrieval variant removes the stylometric tell substantially ($1.000 \rightarrow 0.636$ against a same-population control of 0.443) and pushes enumeration to $\sim$1250 probes for a 10^6-entry cache, but a residual content-relevance tell (0.764) remains: a cached deflection answers a *similar*, not the exact, prompt. The honest reading is that statistical indistinguishability at pure-CPU serving is *not* achieved—matching a model's output distribution provably requires either a model in the serving path or a finite cache that is eventually enumerable—so a deception layer should be deployed for its cost-and-attribution value, not as a claim that the attacker is reliably fooled. The same-population control (0.443) is essential: without it, a stylometry AUC near 1.0 could merely reflect a classifier separating any two sources rather than a genuine fingerprint.

VII. Cryptographic Audit Substrate Evaluation

The verification procedure run by the offline browser verifier is given in Fig. 2: it resolves the issuer's published key, recomputes the Merkle root, checks the post-quantum signature, and confirms the timestamp references the credential—contacting no issuer server beyond fetching the public DID document.

Table IV summarizes the audit-layer verification. Signatures are genuine ML-DSA-65 (3309-byte) values, not placeholders. Tampering with a recorded action changes the Merkle root; re-hashing the root without the private key is caught by signature verification; a forged signature from a different key fails. A live

evidence bundle—a real signed credential, its DID document, and an RFC-3161 token from a public timestamp authority—together with an offline HTML verifier lets any reviewer check the signature and timestamp *without contacting our servers*.

TABLE IV
AUDIT SUBSTRATE VERIFICATION

Property	Result
Signature realness	ML-DSA-65, 3309-byte (not stub)
Action-replay tamper	caught (verdict TAMPERED)
Metric-inflation tamper	caught (signature invalid)
Forged-key signature	rejected
RFC-3161 timestamp	granted; token references digest (live TSA)
Test coverage	96 unit assertions, 7 test files

VIII. LIMITATIONS AND DISCLOSED FAILURE MODES

We treat candid disclosure as a contribution, not an afterthought.

- **Narrative jailbreaks.** wildjailbreak reaches only 16% TPR; the broad-mix aggregate is 47.6% (Table II). This is a measured weakness, stated plainly—and, as we argue in Section X, it is the expected ceiling of any energy-bounded detector against an adaptive attacker, which is precisely why we pair detection with a non-repudiable audit substrate rather than treating the detection number as the whole defense.
- **Single canonical number.** Two operating points exist for the same methodology (77.4% on 2026-06-07; 97.7% on 2026-06-08) differing only in environment defaults. We publish the latter, dated, as the single canonical figure and retain a consistency record; both numbers must not circulate.
- **Key custody.** Keys live in process memory (software custody). Hardware (HSM/enclave) custody is roadmap; credentials must not claim it.
- **Timestamp trust.** RFC-3161 tokens are verified to be granted and to reference our digest, but full TSA certificate-chain validation is not yet performed.
- **Content sample size.** The content-safety F-measure (0.814) is on 250 prompts and depends on a running local model; it is a demonstration result.
- **Auxiliary layers.** Several cascade layers (drift, watermarking, coherence gating) are not separately measured; NPU paths run as CPU fallback in our measurements.
- **Deception indistinguishability.** The retrieval deception layer reduces but does not eliminate fingerprintability (held-out stylometry AUC 0.636; residual content-relevance tell 0.764); statistical indistinguishability at pure-CPU serving is not achieved (Section VI), so the layer is default-off and not relied upon for security.
- **Audit hygiene.** The log stores unsalted SHA-256 hashes of prompts (a rainbow-table risk for common strings), has no revocation mechanism, and is in-memory (a hosted append-only log is roadmap).
- **Retired claims.** Components that did not survive measurement (a stenographic "compression" module and a related metric) were removed and are documented as retired.

IX. REPRODUCIBILITY AND AVAILABILITY

All headline numbers are computed by an open harness over 100% public corpora—no NDA, no private test set. The two-number scorecard is regenerable with a single command, and every value traces to a dated JSON manifest. Cryptographic claims are covered by 96 unit assertions runnable with a standard test runner. A live, browser-verifiable evidence bundle (a real ML-DSA-65-signed credential, a DID document, and an RFC-3161 token from a public TSA) is published so that any third party can verify the signature and timestamp offline. We publish one canonical, dated number per claim and a consistency ledger documenting reconciliations and retired claims.

X. DISCUSSION

Trustworthiness as infrastructure, not paperwork. The properties that make a guardrail's output *verifiable*—signed receipts, an append-only transparency log, published issuer keys—are the same properties that let downstream parties build on that output without re-checking it. A regulator, an auditor, or a partner system can accept a verdict because the evidence is independently checkable, not because the vendor asserts it. This reframes compliance work from a static document exercise into a runtime, machine-verifiable artifact, which is what makes it scale to the speed at which LLM agents now operate. Concretely, the record-keeping and logging duties of the EU AI Act (Art. 12, Art. 19, Annex IV) are today discharged as human-authored documents; a signed, timestamped, offline-verifiable receipt per action discharges the same duty as a machine-checkable artifact. We claim a *mapping* to these duties, not conformity—conformity is a legal determination, made against harmonized standards and, for high-risk systems, a notified body, that we do not and cannot assert from a technical artifact alone. That boundary is deliberate: verifiable evidence is a precondition for conformity, not a substitute for it.

Why per-attack detection is the wrong target. Our 16 percent narrative result (Table II) is not an anomaly to be engineered away but the expected ceiling of any *energy-bounded* detector. Recent work shows that an adaptive attacker who moves second—probing a committed guardrail until it yields—bypasses twelve recent defenses, the majority of which had originally reported near-zero attack-success rates, at attack-success rates above 90 percent for most [12]. A defender who commits a fixed decision boundary and is then probed is, in game-theoretic terms, the leader in a Stackelberg game whose follower best-responds; with query access the boundary is learnable. Chasing a single per-attack detection number is therefore a race the defender loses by construction, and inflating that number on private corpora merely hides the loss. We argue the deployer's true objective is not the per-

attack detection rate $P(success)$ but the attacker's expected value over a *repeated* campaign,

$$EV = P(success) \cdot V_{success} - C_{attempt} - P(attributed) \cdot \Pi_{attribution},$$

where V is the value of a success, C the cost of an attempt, and Π the penalty when an attempt is attributed. A detector touches only the first term. The trust layer of this paper recruits the others at CPU cost: a cheap automaton blocks the easy majority before any accelerator spend (C); an optional deception layer aims to lower the value of a success (V), which Section VI measures as only *partially* achievable at pure-CPU serving (and so is not relied upon); and—decisively—the audit substrate makes a successful bypass *non-repudiably attributable after the fact* ($P(attributed) \cdot \Pi$), which is the term cloud detectors omit because they ship no portable, post-quantum receipt. This is the same architecture biological immunity settled on: a cheap innate barrier plus durable memory that wins the *repeated* encounter rather than guaranteeing first-contact sterilisation. We are careful not to overstate it: the framing is strongest against campaign and repeat attackers whose actions are traceable, and weakest against a one-shot, anonymous, high-value extraction where Π is unenforceable; and we do not report a fused EV figure—the attacker's value function is not ours to measure—reporting instead the separable mechanism terms, consistent with our two-number discipline.

Deployment guidance. Because Stage 1 is a CPU automaton that runs before any GPU inference, it doubles as an economic filter: traffic blocked at Stage 1 never consumes accelerator time. We recommend treating the two-number verdicts as *independent* gates—an injection block and a content block are different policy decisions—rather than fusing them into one threshold, which would re-introduce the very ambiguity the two-number methodology removes.

Sovereignty. On-premise, CPU-only screening is not merely a performance choice; for organizations under data-residency constraints (healthcare, finance, public administration) it is a precondition for using a guardrail at all, since cloud screening would require exporting the prompt. The audit substrate complements this by producing portable, offline-verifiable evidence that never depends on the issuer's continued availability.

Security claims must be reproducible. We stress that for security tooling specifically, a headline number that cannot be independently reproduced is a liability rather than an asset: it cannot be trusted by the parties who most need to. Publishing the harness, the public corpora, and the measured failure modes is therefore not a courtesy but a requirement we impose on ourselves.

A content-provenance companion. The same demonstrate-not-assert discipline underpins a companion layer, FORTRESS [14], which embeds invisible, pipeline-surviving provenance marks in AI-generated media for EU AI Act Article 50 transparency and reports a multi-number scorecard with a null-controlled false-attribution guard (measured false attribution 0/12 against un-watermarked and foreign images). Its provenance attributions can be anchored as leaves in the same LEDGER transparency log described here, so that generation-time logging (Art. 12/19) and output marking (Art. 50) are covered by one post-quantum, browser-verifiable substrate; the two systems thus form a sovereign trust ecosystem spanning model output through content distribution.

Future work. Closing the narrative-jailbreak gap likely requires a trained classifier rather than an automaton; hardware key custody (enclave/HSM) would remove the software-custody caveat; full TSA certificate-chain validation and a hosted, externally-witnessed transparency log would strengthen the audit guarantees; and a larger, multi-thousand-prompt content-safety benchmark would replace the present demonstration-scale result. Following the expected-value framing of Section X, we also plan to measure the cost-shaping mechanism terms directly and separately—the rate at which a discovered attack stops transferring once the decision boundary is perturbed, the number of repetitions before a discovered attack is neutralised, and the fraction of decisions carrying a verifiable receipt—reported as a vector rather than a single fused number; these mechanism measurements are not yet available and we make no quantitative claim about them here.

XI. Conclusion

We have described a sovereign, CPU-first trust layer that pairs a reproducible two-number injection/content guardrail with a post-quantum, browser-verifiable audit substrate, and we have reported its strengths and its measured weaknesses with equal prominence. The design embodies a single principle—trustworthiness should be *demonstrated, not asserted*—which we believe is the right foundation for deploying LLM systems in regulated European settings. That principle also reframes the objective: because per-attack detection is a race an energy-bounded defender provably loses, the durable contribution is not a detection score but a verdict whose evidence is independently checkable, which is what turns a successful attack from an invisible event into an attributable, penalisable one. We invite collaborators and independent re-evaluation; the harness and the live verifier are the invitation.

Acknowledgment

We thank the maintainers of the public injection and safety corpora, whose openness makes reproducible security evaluation possible.

References

[1] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," Commun. ACM, vol. 18, no. 6, pp. 333–340, 1975.

[2] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, Internet Engineering Task Force, 2013.

[3] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)," RFC 3161, 2001.

[4] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard," FIPS 204, 2024.

[5] World Wide Web Consortium, "Verifiable Credentials Data Model v2.0," W3C Recommendation, 2025.

[6] World Wide Web Consortium, "Decentralized Identifiers (DIDs) v1.0," W3C Recommendation, 2022.

[7] European Parliament and Council, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act)," Official Journal of the European Union, 2024.
[8] S. Schulhoff et al., "Ignore this title and HackAPrompt: exposing systemic vulnerabilities of LLMs through a global prompt hacking competition," in Proc. EMNLP, 2023.
[9] X. Shen, Z. Chen, M. Backes, Y. Shen, and Y. Zhang, " 'Do anything now': characterizing and evaluating in-the-wild jailbreak prompts on large language models," in Proc. ACM CCS, 2024.
[10] A. Souly et al., "A StrongREJECT for empty jailbreaks," 2024, arXiv:2402.10260.
[11] J. Cui, W.-L. Chiang, I. Stoica, and C.-J. Hsieh, "OR-Bench: an over-refusal benchmark for large language models," 2024, arXiv:2405.20947.
[12] M. Nasr, N. Carlini, C. Sitawarin, S. V. Schulhoff, J. Hayes, M. Ilie, J. Pluto, S. Song, H. Chaudhari, I. Shumailov, A. Thakurta, K. Y. Xiao, A. Terzis, and F. Tramèr, "The attacker moves second: stronger adaptive attacks bypass defenses against LLM jailbreaks and prompt injections," 2025, arXiv:2510.09023.
[13] C. Wu, Y. Wang, and Y. Liao, "Active honeypot guardrail system: probing and confirming multi-turn LLM jailbreaks," 2025, arXiv:2510.15017.
[14] H. Schmidt, "FORTRESS: a sovereign, reproducible provenance layer for AI-era content with a null-controlled false-attribution guard," DESTILL.ai / FORTRESS, 2026, companion manuscript.