DEON: Runtime Guards for Policy-Safe Agentic Service Composition under Attack

Xuan-Bach Le[1] ✉*, Phat T. Tran-Truong[1], Duy Nguyen[2], and Ngoc Khanh Tran Nguyen[3]

[1] Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Vietnam
{lexuanbach,phatttt}@hcmut.edu.vn
[2] University of Economics Ho Chi Minh City, Vietnam
duynguyen.726202321440@st.ueh.edu.vn
[3] University of Science, VNU-HCM, Vietnam
22120378@student.hcmus.edu.vn

Abstract. Large language models increasingly orchestrate services, but attacker-controlled inputs can induce access-control violations, data leakage, omitted mandatory steps, and quota overruns. Prompt-based defences reduce these failures without eliminating them. We place two runtime checks between an untrusted proposer and service execution. Our *deontic policy automaton* checks each proposed action against explicit permissions, prohibitions, and obligations. Under correct catalogue labels, complete mediation, and an adequate policy, no action violating the policy's safety fragment executes. Terminally pending obligations are detected but cannot be forced. A mislabelled action falls outside this guarantee. Our conformal guard handles residual violations, and we calibrate its detector threshold on labelled data. The pass-through bound holds only while deployment violation scores stay exchangeable with calibration scores, and adaptive low-scoring rewrites break it and are admitted. On DEON-BENCH, the automaton agrees with ground truth on **100%** of 900 traces across six violation classes. For content egress that the policy cannot identify, the calibrated layer admits **8.9%** of leaks against a **10%** target, rising to **32.8%** under adaptive obfuscation. Across six proposer models, our guard reduces attack success for policy-expressible injections from **87–100%** to **0%**, with microsecond-scale monitoring overhead.

Keywords: Agentic service composition · Runtime verification · Conformal prediction · Prompt injection · Deontic logic.

1 Introduction

In agentic service composition, a language model reads a goal and an API catalogue, then selects service calls step by step [24]. The ecosystem may encode

* X.-B. Le and P. T. Tran-Truong contributed equally to this work.
✉ Corresponding author: Xuan-Bach Le, lexuanbach@hcmut.edu.vn.
Artifact and extended version: https://github.com/lexuanbach/deon.

access rules as executable policy [23,15], but the model also reads attacker-controlled content, and indirect prompt injection can therefore alter the actions it proposes [7,10]. Policy violation is a prominent LLM risk [17]. In our experiments, six proposers select a policy-violating action on **87–100%** of attacked tasks. Prompt-based defences reduce such violations without eliminating them. Static plan verification [2] applies to a fixed plan, whereas observations change during execution, and selective prediction [6,1] does not represent service policy.

XACML [14], usage control [18], and OPA/Rego gateways [15] provide useful enforcement, but an agentic setting also needs sequential state and a decision path that trusts no proposer-supplied metadata beyond the action identity and arguments. We provide both with two checks before every action. Our *deontic policy automaton* (DPA) checks explicit permissions, prohibitions, and obligations against the executed prefix. A *conformal execution guard* calibrates a threshold for the residual class the policy cannot express, targeting $\mathbb{P}[\text{admit} \mid \text{violation}] \leq \alpha$. These guarantees require correct action labels, complete mediation, an adequate policy, and exchangeable calibration data (A1–A4, §4), and an under-labelled action thus falls outside Thm. 1 and its measured 0%.

We evaluate whether the DPA matches the trace semantics and the conformal layer meets its target (**RQ1**), whether Deon cuts attack success across proposer sizes (**RQ2**), and how the calibrated layer handles policy-inexpressible violations under exchangeable and adaptive inputs (**RQ3**). The extended version adds hardening, calibration sensitivity, and dependent steps (RQ4–RQ6).

2 The Deon Model

Rules over a service catalogue $\mathcal{A}$ are naturally *deontic* [28]. Prohibitions subsume access control [23], lattice information flow [4], and quotas, while obligations are a *liveness* property a blocking monitor [11] can only *detect*. Conformal prediction [27,1] gives any score a distribution-free guarantee under *exchangeability*, with a reject option [6] and a training-conditional variant [26].

Definition 1 (Action and context). *An* action *a is a service invocation with the fields it reads/writes, an optional egress* sink *with a clearance level, and an optional metered* resource. *A* monitor state *$q = (\sigma, u, \Omega, \lambda)$ records the propositions σ a trace has established (e.g. `authenticated`), usage counters u, pending obligations Ω, and a field→lattice labelling λ.*

Definition 2 (Deontic policy and its automaton). *A* deontic policy *$\Pi = (\mathsf{P}, \mathsf{F}, \mathsf{O})$ maps each action to* permission preconditions P*, comprises* prohibitions F *(forbidden operations, quotas, and a flow rule admitting a field labelled ℓ only to a sink clearance c with $\ell \sqsubseteq c$), and maps triggers to* obligations O *that must occur later. The prototype uses the total chain `public`<`internal`<`pii`<`secret`. Obligations are Boolean eventualities. Repeated identical triggers coalesce. A trace $\pi = a_1 \cdots a_k$* satisfies *Π iff no a_i is prohibited or violates a precondition in its prefix state and every triggered obligation is discharged before the end. The DPA*

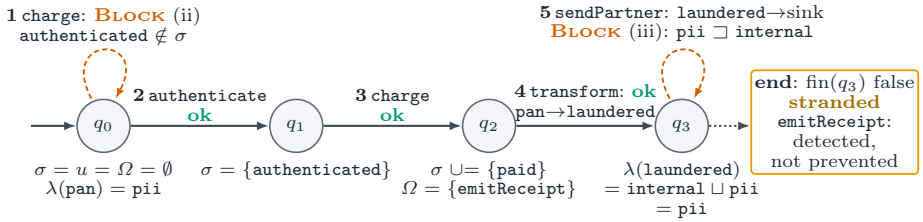

Fig. 1. A DPA run (Def. 2) under the policy of §7. Refusals are self-loops (q unchanged).

for Π is the monitor $M_\Pi = (\mathcal{Q}, q_0, \delta, \mathrm{fin})$. From state q, a candidate a has transition $\delta(q, a) = (ok, q')$, where ok is true iff (i) a is not forbidden, (ii) $\mathsf{P}(a) \subseteq \sigma$, (iii) a routes no over-cleared field to its sink, and (iv) a's resource is within quota. When ok, the successor q' adds a's granted propositions, increments its counter, discharges a from the old Ω, and then registers $\mathsf{O}(a)$. A self-trigger requires a later occurrence. The predicate $\mathrm{fin}(q)$ *holds iff $\Omega = \emptyset$. M_Π* admits π *iff every step is ok and* fin *holds at the end.*

Against *laundering* (copy a high field, egress the copy), each field w that a writes takes the lattice join $\lambda'(w) = \lambda_0(w) \sqcup \bigsqcup_{r \in R(a)} \lambda(r)$ of its default tag and a's read set $R(a)$. Clause (iii) then blocks a laundered field like a direct flow, at field-tag granularity. Because transitions are bounded lookups reading only q and Π, the DPA is a linear-time online monitor.

Example. In Fig. 1 (policy of §7), `charge` requires `authenticated`, grants `paid`, and obliges a later `emitReceipt`. Field `pan` is `pii`, and untagged written fields default to $\lambda_0 =$ `internal`. Trace state decides three verdicts. The same `charge` fails clause (ii) at step 1 and passes at step 3, once step 2 puts `authenticated` in σ. Because the join tags step 4's copy of `pan` as `pii`, its egress to an `internal` sink fails clause (iii) at step 5, though its default tag passes. At the end $\Omega \neq \emptyset$ and fin fails, flagging a stranded obligation the monitor cannot prevent.

An attacker may still disguise a violation as a benign-looking action whose admissibility depends on context the policy cannot name. We model it as a latent label $y \in \{0, 1\}$ seen via a noisy score $\mathsf{s}(a, q) \in \mathbb{R}$. The verdict is BLOCK if $\delta(q, a)$ is not *ok*, ABSTAIN if the DPA permits a and $\mathsf{s}(a, q) > \tau$, and ADMIT otherwise.

3 Architecture and Integration

Fig. 2 separates the proposer, guard, and executor. Only the proposer interprets attacker-influenced text as instructions. The guard may score candidate content as untrusted data. Only the executor causes side effects, accepting actions solely through the guard, whose monitor tracks q over the executed prefix. Failed calls leave q unchanged, and idempotent retries re-enter the guard. Current service schemas lack the labels of Def. 1. OpenAPI [16] does not express them, and

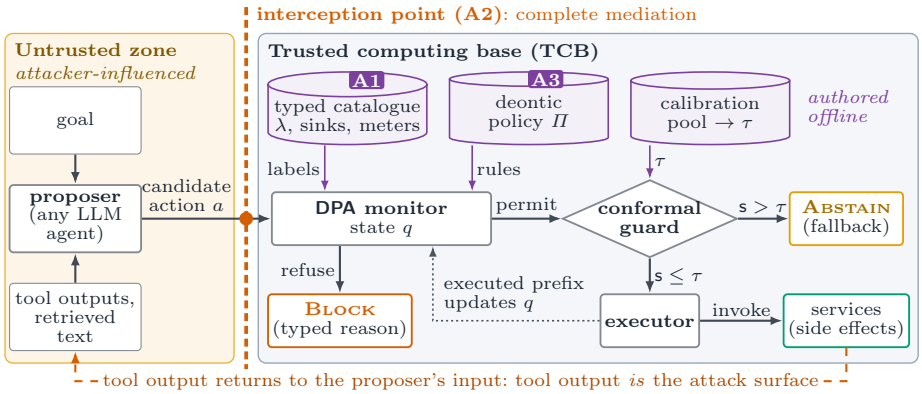

Fig. 2. Deon reference architecture. Every effectful action crosses the interception point (A2) once, before any side effect. The bottom loop is the threat [7].

MCP's optional read-only and destructive hints [13] are advisory metadata supplied by the server. Proposer-supplied labels would undermine the flow rules. For MCP, the guard can sit at the client's dispatch point, in a proxy server that re-exports downstream tools, or at a network gateway. Our prototype mirrors the first. The second is recommended but untested, and the third cannot mediate local stdio traffic. Labels are keyed by server, tool, and argument path.

4 Threat Model and Deployment Assumptions

We assume a content-level adversary who knows the defence and can write any text the agent reads [7], steer it to any catalogue action, and adapt content to evade s. It cannot modify trusted policy, labels, calibration, guard state, or code, or bypass mediation. Implicit/timing channels, aliasing, and supply-chain compromise are out of scope. Probability-zero and measured-zero claims require A1–A3. Conformal claims also require A4.

A1 (trusted, complete action labelling). Every reachable action arrives at the guard carrying correct type labels: the tagged fields it reads and writes, the sink it targets and that sink's clearance, and the meter it charges. The operator admitting a service authors them out of band, as trusted configuration (§3). *Over-labelling* makes the monitor refuse a benign action, naming the failed clause. This loud, attributable failure reduces availability. A missing catalogue entry is visible and defaults to Abstain. *Under-labelling* of a present entry is silent. Clause (iii) cannot fire, the action is permitted, and nothing is reported. The action leaves Thm. 1's class, and the 0% of §7 holds only over what A1 covers. RQ3 instead measures content semantics outside the policy language, an under-labelling analogue that estimates no production labelling-error rate.

A2 (complete mediation), A3 (policy adequacy), A4 (exchangeable calibration). A2 requires every effectful action to reach a serialized atomic check–execute–commit transition. The proposer has no executor credentials. A3 requires Π to represent the operator's intended forbidden, required, and metered behavior. A4 requires exchangeability of calibration and deployment violation scores. DEON-BENCH satisfies A1–A3 by construction. Its monitor accuracy and guarded ASR measure the guard under those assumptions.

5 The DEON Guard

At each step the proposer suggests an action from the goal and attacker-influenced observations. The DPA refuses it (BLOCK, with a typed reason, q unchanged) or permits it, and the conformal guard ABSTAINS if $\mathsf{s}(a, q) > \tau$ and otherwise executes it, advancing q. If fin fails, the loop ABSTAINS or falls back to a recovery ladder (restricted retry $\rightarrow$ rollback $\rightarrow$ escalation) that never runs an unverified action. The threshold is set once, offline. With $v_{(1)} \leq \cdots \leq v_{(n_v)}$ the scores of a labelled sample's n_v violations, define $v_{(0)} = -\infty$ and set $\tau = v_{(k)}$ for $k = \lfloor \alpha(n_v + 1) \rfloor$, $0 < \alpha, \delta < 1$. Rank zero or an empty violation pool abstains on all finite scores. For ties, independent $U \sim \mathrm{Unif}(0, 1)$ values augment calibration and test scores and pairs are ranked lexicographically. For a training-conditional guarantee in the probably approximately correct (PAC) sense, we take the largest k' with $\mathbb{P}[\mathrm{Bin}(n_v, \alpha) \geq k'] \geq 1 - \delta$ [26]. The policy's supported *deny* fragment maps to Rego/OPA [15] over trusted attributes supplied by the DPA wrapper. Deny conditions in this fragment can be reused. Attribute construction, counters, taint, prefix state, and obligations remain wrapper-specific.

6 Theoretical Analysis

Theorem 1 (DPA soundness). *Under A1 (correct type labels on every reachable action), A2 (no effectful action bypasses the guard), and A3 (Π is the intended policy), M_Π realises the deontic trace semantics of Π.* (Safety) *No executed action is prohibited or violates a precondition, an explicit or laundered flow rule, or a quota in its prefix state, and blocking* enforces *all of these.* (Obligations) *The monitor* detects *at end of trace whether every triggered obligation was discharged, but it cannot* enforce *one online. A trigger that runs and then abstains therefore leaves an unpreventable stranded-obligation violation.*

Theorem 2 (Conformal admission bound). *Let $\tau = v_{(k)}$, $k = \lfloor \alpha(n_v + 1) \rfloor$ (§5).* (i) Marginal. *For a fresh DPA-permitted residual violation whose score S is exchangeable with the calibration scores for that population and untied (or randomized as in §5), $\mathbb{P}[\textit{admit} \mid \textit{residual violation}] = \mathbb{P}[S \leq \tau] \leq \alpha$. This per-action bound does not bound a trace's chance of at least one leak or the reverse conditional $\mathbb{P}[\textit{violation} \mid \textit{admit}]$.* (ii) PAC (exact). *If, more strongly, the calibration violation scores are i.i.d. draws from the continuous law F of deployment violation scores, then with the largest k' such that $\mathbb{P}[\mathrm{Bin}(n_v, \alpha) \geq k'] \geq 1 - \delta$ and $\tau = v_{(k')}$, the realised pass-through satisfies $\mathbb{P}_{\mathrm{cal}}[F(\tau) \leq \alpha] \geq 1 - \delta$.*

Corollary 1 (Attack-success bounds). (a) *Under A1–A3, for* any *proposer and adversary, no DPA-covered violation executes. The probability is exactly* 0 *for every adversary, and it is a per-deployment guarantee that holds whenever the deployment satisfies A1–A3.* (b) *A latent violation with calibration-exchangeable score executes with probability* $\leq \alpha$ only *under A4. An adaptive adversary breaks exchangeability, letting pass-through approach* 1.

Intuition (proofs in the extended version). Each clause of δ checks its counterpart in Π in the prefix state. Thm. 2 follows the standard exchangeability argument, with a binomial tail for PAC. Cor. 1(b) needs no trusted labels but depends on A4, since any violation scoring $\leq \tau$ is admitted, including an adaptive rewrite.

7 Evaluation

Deon-Bench is ground-truth-labelled because every generator knows what it planted. Its *monitor suite* has 900 traces (450 compliant, 450 with one planted violation) over a fixed payment/PII policy with prohibited operations, `pii`/`internal`/`public` tags, preconditions, two obligations, and a quota. The 450 odd-indexed traces draw their violation class uniformly from *prohibited*, *sequencing*, *flow*, *quota*, *obligation*, and *laundering*, the last caught only by the join (Fig. 1, steps 4–5), with **84/65/81/83/70/67** per class at the released seed. *Latent-risk pools* model what the policy cannot name. Of n=4000 calibration draws, the $n_v \approx 1200$ violations set τ, and a fresh n=20,000 evaluation comes from the *same* generator, over 300 trials per α. These pools are exchangeable by construction. Three attack suites sit increasingly far from our design. A mixed stream of 4000 proposed actions per trial (35% explicit, the rest latent) is the only joint test of both layers. A *proposer ladder* spans eleven operations but describes only the eight benign ones to the proposer. The banking suite of *AgentDojo* [3] adds 76 non-degenerate user×injection pairs. A *content-egress* task scores real text with a real, imperfect detector, and its *adaptive* pool re-emits the leaks obfuscated. We compare *Unguarded*, *DPA-only*, *fixed-threshold* ($\mathtt{s} \leq 0.5$, the *w/o calibration* ablation), and Deon over six local Ollama proposers (2B–47B), plus AgentDojo's shipped defences and three detectors. Table 1 reports 30+30 ladder runs per model: five benign and five attack tasks, six repetitions each, temperature 0.7, and no model retry.

RQ1: monitor correctness and calibration. The DPA matches ground truth on **100.0%** of the monitor suite at **100%** recall in all six classes, laundering included, and detects the stranded obligations it cannot prevent (Thm. 1). Over $\alpha \in \{0.02, \ldots, 0.30\}$, *marginal* calibration's mean tracks α (0.020–0.299) at **≈100%** utility. Yet only **0.51**–**0.56** of the 300 draws per α hold pass-through $\leq \alpha$, meaning about half of all calibrations miss the target. *Exact-PAC*, the deployable mode, holds it on **93**–**96%** against a $1 - \delta$=0.95 target (Fig. 3a). Across detector quality, Deon holds pass-through at 0.098–0.100 as utility rises $0.76 \to 1.00$, whereas a fixed cutoff swings from a **0.158** leak to over-blocking.

Table 1. RQ2: six-model proposer ladder (↑/↓: higher/lower is better).

		Benign success (%) ↑		ASR (%) ↓	
Proposer	size	unguarded	DEON	unguarded	**DEON**
`gemma2:2b`	2B	100	100	100	**0**
`llama3.2:3b`	3B	97	100	100	**0**
`qwen2.5-coder:7b`	7B	100	100	87	**0**
`qwen2.5-coder:32b`	32B	100	100	100	**0**
`command-r:35b`	35B	100	100	100	**0**
`mixtral:8x7b`	47B	100	100	100	**0**

RQ2: the DPA blocks policy-expressible injections. Across six proposers (Table 1), DEON cuts attack success from **87–100%** to **0%**. All five injections are policy-expressible, and the DPA blocks them with certainty (Cor. 1(a)). Guarded success counts a task once its reference subsequence executes. The 97-to-100 change thus reflects the guard halting a later extraneous violation, without added capability or retry. Because admission never reads the proposer, the 0% is identical at 2B and 47B and reflects the guard given A1, independent of model safety. On the mixed stream, executed violations fall from 100% unguarded to **3.6**% under DEON. The DPA alone leaves 35.7%, the class A3 misses. On AgentDojo, our closest external test of A1 and A3 (undefended: 50.0% utility, 11.8% ASR), prompt-level defences cut attack success only to 9–10%, and the two baselines at 0% drop utility to 31.2% and 25.0%. With a hand-authored counterparty-provenance policy, DEON reaches **0**% at **37.5**% utility, abstaining only on the 2 of 16 benign tasks that take a critical value from an untrusted file.

RQ3: policy-inexpressible content egress. A *content-dependent egress* the DPA *permits* leaks iff its free text is sensitive. Because the DPA reads only labels, it is blind here (pass-through **1.00**, Fig. 3b), and a logistic detector over marker mass, digit density, `@`-count, and token length supplies the signal. Over 30 seeds, exchangeable attacks are bounded at $\mathbf{0.089} \leq \alpha{=}0.10$ at **100**% utility. An obfuscating adaptive adversary breaks exchangeability and drives pass-through to **0.328**, as Cor. 1(b) predicts. Hardening (RQ4, extended version) narrows the gap without closing it (Fig. 3c). A max-ensemble of three detectors keying on different signals holds $\mathbf{0.097} \approx \alpha$ against the obfuscator that drives the lexical detector alone to **0.835**, yet a naturalised payload defeats all three at **1.00**.

In stress tests (RQ5, RQ6, extended version), calibration choices cost only utility. Under benign group-mixture shift, Mondrian thresholds cut the marginal worst group from **0.441** ($4.4 \times \alpha$) to **0.091** at utility $0.95 \to 0.71$, and label noise $\rho{:}\,0 \to 0.3$ drives pass-through $0.085 \to 0.000$ as utility falls $1.00 \to 0.17$. Exceptions are exchangeability and tied scores, where a deterministic $\leq$ leaks **0.343** and the executable augmented-score tie-break has observed mean **0.100** (three-decimal rounding). Exact-PAC needs of order 10^2 *labelled* violations (rank $k'{=}0$ at $n_v{=}20$), and the in-process Python automaton costs 0.76 μs per action

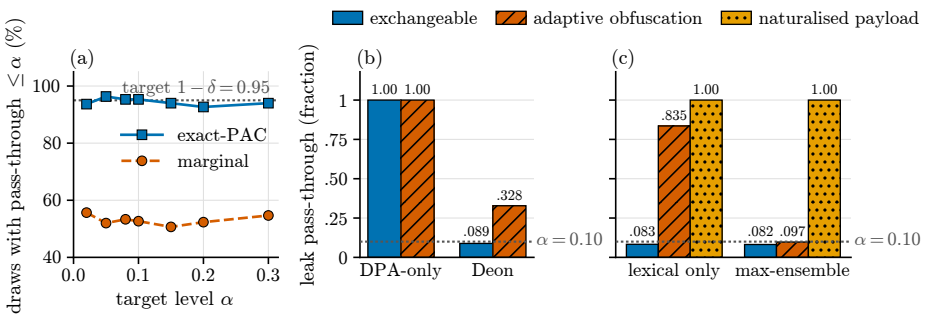

Fig. 3. (a) RQ1: draws holding pass-through $\leq \alpha$. (b) RQ3: leak pass-through. (c) Hardening: lexical detector vs. three-detector ensemble. Dotted in (b), (c): $\alpha = 0.10$.

(host-dependent timing), excluding detector, OPA, proxy, and network costs (R^2=0.99 over lengths 2–12, five batches of 20,000 traces per length).

8 Related Work

XACML [14] is the reference architecture for declarative runtime authorization, and OPA/Rego [15] a cloud-native policy engine. XACML expresses attribute-based rules [8], while Sandhu et al. [23] formalize role-based access control (RBAC). Because XACML requests keep no trace state, q must live outside the decision point. Its *Obligations* are directives returned with the current decision, whereas `charge` creates a liveness obligation for a later `emitReceipt` that may never be proposed (Fig. 1). XACML also trusts request attributes. A1 instead keeps security labels in the trusted catalogue, since attacker-controlled text can steer the proposer.

UCON$_{ABC}$ [18] is the closest classical relative. It provides attribute *mutability* and *continuity* of enforcement, and distributed usage control [21] adds data-flow tracking. The DPA's counters, obligation lifecycle, and per-action gate build on these ideas, specialized as a concrete automaton with an accepting condition, a lattice join as a transition effect, and a decision that excludes requester-supplied metadata. Security-by-contract [5] matches a contract shipped with code against the platform policy and monitors when the match fails, most recently on a UCON/XACML architecture by Rasori et al. [22]. Deon shares this separation of advance evidence and runtime monitoring. Because an LLM proposer ships no verifiable behavioural contract, the trusted description moves to the platform as A1's typed catalogue label. In all these lines, as in runtime verification [11] and information-flow control [4], the engine decides on the attributes it is given. Default behavior is configuration-specific. Deon adds a calibrated residual layer to this trace-monitor interface. The cited conformal formulations [1,6,26] do not encode the service policy or executed-prefix state used here.

Agentic service composition. The service-computing line DEON serves builds compositions *with* LLMs. *Compositio Prompto* embeds them in an automated service-computing architecture [20], NL2TOSCA compiles requirements into TOSCA topologies [12], and MCPybarra generates MCP services [19]. Each plans and then executes, trusting the model's output at execution time. DEON removes that assumption and complements this line, since any of them can be the proposer of Fig. 2. Trace-based audits estimate how reliably an agent completes its tasks after the run [25]. DEON acts before each step and stops the action. QoS and trust prediction [9] brings trust into service *selection*. It does not decide what may execute now and binds no adversary.

9 Discussion and Conclusion

An agent's proposed action carries no execution authority by itself. We enforce that distinction with a deterministic policy layer under A1–A3 and calibrate the residual layer under A4. The synthetic harness satisfies A1–A3 by construction. Production labelling and distributed mediation therefore define the deployment boundary. The adaptive experiment quantifies scorer evasion outside exchangeability, and the proposer study covers six local open-weights models. Within this scope, DEON blocks policy-expressible violations and abstains on residual scores above τ.

Acknowledgments. We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

1. Angelopoulos, A.N., Bates, S., Fisch, A., Lei, L., Schuster, T.: Conformal risk control. In: ICLR (2024)
2. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI. pp. 854–860 (2013)
3. Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., Tramèr, F.: AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In: NeurIPS (2024)
4. Denning, D.E.: A lattice model of secure information flow. Commun. ACM **19**(5), 236–243 (1976)
5. Dragoni, N., Massacci, F., Naliuka, K., Siahaan, I.: Security-by-contract: Toward a semantics for digital signatures on mobile code. In: EuroPKI. LNCS, vol. 4582, pp. 297–312. Springer (2007). https://doi.org/10.1007/978-3-540-73408-6_21
6. Geifman, Y., El-Yaniv, R.: Selective classification for deep neural networks. In: NeurIPS (2017)
7. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M.: Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In: AISec (2023)
8. Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (ABAC) definition and considerations. Tech. Rep. SP 800-162, National Institute of Standards and Technology (2014). https://doi.org/10.6028/NIST.SP.800-162

9. Kumar, S., Chattopadhyay, S.: TRQP: Trust-aware real-time QoS prediction framework using graph-based learning. In: ICSOC. pp. 143–152. Springer (2022)
10. Le, T.H.C., Tran, P.H.V., Mai Duc, T., Le, X.B.: LTM-RAG: Longitudinal trust mining for poison-resilient retrieval-augmented generation. In: Proc. IEEE ICDM (2026), to appear
11. Leucker, M., Schallhart, C.: A brief account of runtime verification. J. Log. Algebr. Program. **78**(5), 293–303 (2009)
12. Meflah, W., Brabra, H., Acheli, M., Ben Douma, H., Sellami, M., Gaaloul, W., Zeghlache, D.: From natural language to TOSCA: Leveraging LLMs for automated service composition. In: ICSOC. pp. 3–18. Springer (2026). https://doi.org/10.1007/978-981-95-5012-8_1
13. Model Context Protocol Project: Model context protocol specification, revision 2025-03-26 (2025), tool-annotation revision. Accessed 24 September 2026. https://modelcontextprotocol.io/specification/2025-03-26/changelog
14. OASIS: eXtensible access control markup language (XACML) version 3.0 (2013), OASIS Standard, 22 January 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
15. Open Policy Agent Project: Open policy agent documentation (2026), accessed 24 September 2026. https://www.openpolicyagent.org/docs/
16. OpenAPI Initiative: OpenAPI specification v3.1.0 (2021), 15 February 2021. https://spec.openapis.org/oas/v3.1.0.html
17. OWASP Foundation: OWASP top 10 for large language model applications (2023), https://owasp.org/www-project-top-10-for-large-language-model-applications/
18. Park, J., Sandhu, R.: The UCON$_{ABC}$ usage control model. ACM Trans. Inf. Syst. Secur. **7**(1), 128–174 (2004). https://doi.org/10.1145/984334.984339
19. Peng, B., Liu, M., Liu, Y., Tian, C., Yu, S., Wang, Z.: MCPybarra: A multi-agent framework for low-cost, high-quality MCP service generation. In: ICSOC. pp. 51–65. Springer (2026). https://doi.org/10.1007/978-981-95-5012-8_4
20. Pesl, R.D., Mombrey, C., Klein, K., Zyberaj, D., Georgievski, I., Becker, S., Herzwurm, G., Aiello, M.: Compositio prompto: An architecture to employ large language models in automated service computing. In: ICSOC. pp. 276–286. Springer (2024). https://doi.org/10.1007/978-981-96-0808-9_20
21. Pretschner, A., Hilty, M., Basin, D.: Distributed usage control. Commun. ACM **49**(9), 39–44 (2006). https://doi.org/10.1145/1151030.1151053
22. Rasori, M., Mori, P., Saracino, A., Aldini, A.: Exploiting usage control for implementation and enforcement of security by contract. Internet of Things **33**, 101697 (2025). https://doi.org/10.1016/j.iot.2025.101697
23. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Comput. **29**(2), 38–47 (1996)
24. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. In: NeurIPS (2023)
25. Tran-Truong, P.T., Le, X.B.: TraceToChain: Auditing LLM-agent reliability with absorbing Markov chains. In: Proc. IEEE ISSRE (2026), to appear
26. Vovk, V.: Conditional validity of inductive conformal predictors. Mach. Learn. **92**(2–3), 349–376 (2013)
27. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer (2005)
28. von Wright, G.H.: Deontic logic. Mind **60**(237), 1–15 (1951)