Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Towards Trustworthy and Governable Multi-Agent Systems: Detection and Mitigation of Prompt Injection and Tool-Call Hijacking in Enterprise DevOps Pipelines

Krishna Murthy Pradhan

krishnamurthy.pradhan@gmail.com

Independent Researcher

Durga Prasad Mahapatra

National Institute of Technology Rourkela

Research Article

Keywords: agentic AI security, prompt injection, tool-call hijacking, runtime enforcement, DevSecOps, audit trace

Posted Date: November 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11723623/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Towards Trustworthy and Governable Multi-Agent Systems: Detection and Mitigation of Prompt Injection and Tool-Call Hijacking in Enterprise DevOps Pipelines

Krishna Murthy Pradhan*
Independent Researcher
Hyderabad, Telangana, India
*Corresponding author: krishnamurthy.pradhan@gmail.com

Durga Prasad Mahapatra
Dept. of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, India

***Abstract*—Agentic AI systems in DevOps pipelines can issue tool calls that are individually permitted yet jointly unsafe, as in indirect prompt injection and tool-call hijacking; audit logs record the damage only after the fact. We ask what a deterministic mediator that judges each proposed call against the run's accumulated path buys over per-call authorization, and what it costs legitimate work. On a local simulated testbed with 19 DevOps scenario families (228 runs) and 3162 cases derived from InjecAgent, audit-only monitoring let all 132 attack instances execute; the mediator, with every escalation approved, let 12 through (pipeline tampering, which carries neither a sensitive nor a boundary signal), blocked 10 of 11 attack families against 8 for per-step authorization with validated approvals ($p = 0.50$), and wrongly blocked 2 of 8 benign families against 3 under session-latch memory. Path memory mattered only for transformed and cross-agent exfiltration. A hash-chained trace is replayable, but a coherent forgery passes both chain and replay unless the log head is anchored externally. Agents are scripted and scenarios author-built: the results show a mechanism, not field effectiveness.**

***Index Terms*—agentic AI security, prompt injection, tool-call hijacking, runtime enforcement, DevSecOps, audit trace**

I. Introduction

An agent that reads an issue, retrieves a CI/CD secret and posts to a webhook has made three individually permitted calls (Fig. 1). Whether the sequence is a breach depends on the run so far, not on any single call. Indirect prompt injection supplies the trigger: instructions hidden in issues, pull requests or build output can redirect an agent that treats retrieved content as commands [2], [11]. An allow-list says which tools exist, a per-call authorizer sees only the current proposal, and an audit log records an external send but cannot retract it.

Path-aware governance is not new [3], [5], [10], and we do not claim it. The gap is controlled evidence, with workload and agent behaviour held fixed. We ask four questions: **RQ1** does mediation before dispatch stop what audit-only monitoring permits? **RQ2** can every decision be reconstructed deterministically from one typed trace, and where does that guarantee end? **RQ3** what does path memory add over per-step authorization with equally strict approval checks, and what does it cost in wrongly blocked work? **RQ4** which claims remain unsupported, where does the mechanism fail under adaptive probes and a richer action taxonomy, and what must be measured next?

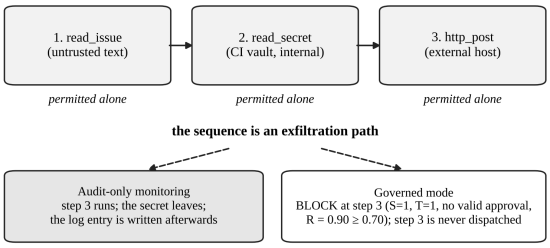

Fig. 1. The problem. Each call is permitted alone; the sequence is an exfiltration path. Audit-only monitoring logs it after the send; governed mode blocks it before.

Contributions. (N1) A deterministic pre-dispatch mediator with validated, scoped, single-use approvals and lineage-scoped sensitivity, and a hash-chained action trace with a replay checker. (N2) A controlled comparison of eight policies, including a per-step baseline with the same approval checks, on 19 DevOps scenario families and on a mapping of the public InjecAgent benchmark [11]. (N3) The cost side reported with the same prominence as the benefit: wrongly blocked benign work, review load and blocked authorized transfers. (N4) Stated failures: adaptive probes, and an integrity experiment in which coherent forgeries pass both chain and replay.

II. Related Work

Execution monitors enforce safety properties [9], edit automata add suppression and insertion [4], and complete mediation requires every access to be checked [8]; our policy is a safety property over finite action prefixes. InjecAgent [11], AgentDojo [1] and ToolEmu [7] measure model susceptibility, not a pre-execution layer with replayable decisions. Kaptein et al. [3] propose policies over paths, Paduraru et al. [5] trace-based contracts, and Uchibeke [10] a synchronous per-call authorizer. None is evaluated on our workloads, so we compare against our own per-step baseline, not their systems.

Table 1
CLAIMS, RESEARCH QUESTIONS AND EVIDENCE.

Claim	RQ	Evidence and result
C1 Pre-dispatch mediation stops what audit-only permits	RQ1	S2, 132 attacks: payload executed 132/132 vs 12/132
C2 Path state beats per-step authorization with validated approvals	RQ3	S2, 11 families: 10/11 vs 8/11; not significant ($p = 0.50$)
C3 Path memory is needed only for a specific class	RQ3	Encode-then-publish and handoff; not on S3
C4 Lineage cuts over-blocking vs the session latch	RQ3	2/8 vs 3/8 benign families blocked; recall unchanged
C5 Decisions replay; limits are stated	RQ2, RQ4	730/1000 re-hashed edits caught; 120/120 forgeries pass

Relationship to the pilot. To keep the two submissions distinct, this paper reuses the pilot's scorer, constants and signal definitions unchanged, does not re-run its twelve families and does not repeat its results. New here: the DevOps suite, validated approvals, lineage-scoped sensitivity, the InjecAgent mapping, and the integrity, latency and adaptive-probe experiments. The overlap is disclosed in the declarations.

Position in the programme. This is the second step of a staged doctoral programme on run-time governance of tool-calling agents. The pilot [6], a shorter paper under review at a workshop, established the mechanism on twelve synthetic families and found no clear gain from session-level path memory. Here we retire three of its stated limits: a per-step baseline with the same approval checks, lineage-scoped memory in place of the session latch, and a mapping to InjecAgent. Blind labelling, LLM-driven agents and a production authorizer remain open.

III. THREAT MODEL AND DESIGN

The agent proposes calls to a bounded set of DevOps tools and ingests attacker-influenced content (issue text, pull-request descriptions, build output). A mediator between agent and tool gateway sees structured action metadata and the session history, never prompts or model reasoning. The attacker can influence what the agent proposes, but cannot alter the mediator, forge trace entries, or obtain a valid approval for a destination the approver did not authorize.

Signals and scorer. Three binary signals are derived from a proposed action a_t and the prior trace: sensitive-data involvement S, valid approval A (absence is missing approval) and trust-boundary crossing T. A deterministic scorer maps them to a rule output and verdict:

$$R_t = \sigma\big(b + w_S S + w_A(1 - A) + w_T T\big), \qquad (1)$$

with the verdict *block* if $R_t \geq 0.70$, *escalate* if $R_t \geq 0.30$, else *allow*; $b = -1.0$, $w_S = 1.20$, $w_A = 1.00$, $w_T = 1.0315$, hand-set and carried unchanged from the pilot [6]; none was tuned to any result here. R_t is a rule output, not an attack probability. With binary signals the surface has eight points (Table 2), all enumerated by the test suite. Approval lowers the score but never clears a sensitive boundary crossing, and a missing approval alone scores 0.50 and escalates.

Table 2
SCORE SURFACE OF THE SCORER (ALL EIGHT SIGNAL COMBINATIONS).

S	A	T	R_t	Verdict
0	1	0	0.27	allow
0	0	0	0.50	escalate
0	1	1	0.51	escalate
1	1	0	0.55	escalate
0	0	1	0.74	block
1	0	0	0.77	block
1	1	1	0.77	block
1	0	1	0.90	block

Signal definitions. $S = 1$ if the action reads a sensitive source (a secret, a private repository or a signing key) or consumes an artifact tainted by one, under the policy's memory model. $A = 1$ if the action carries an approval that, under validated checking, names the destination host actually used, has not expired and has an unconsumed nonce; under asserted checking the mere presence of an approval object gives $A = 1$. $T = 1$ if the parsed destination host does not end with an internal suffix, so look-alike hosts such as `vault.corp.internal.attacker.net` count as external.

Two extensions to the pilot. *Validated approvals:* an approval carries a destination scope, lifetime and nonce, and counts only if unexpired, unconsumed and scoped to the host actually used. *Lineage-scoped sensitivity:* instead of latching S for the whole session, taint follows the artifacts an action consumes and produces, so a derived artifact sent outward is flagged while an unrelated call after an unrelated secret read is not (no declassification).

Sensitivity memory. Three models are compared. *Path-blind* scoring flags an action only if it reads a sensitive source itself or consumes an artifact produced directly by such a read, so derived artifacts lose their taint. The *session latch*, used in the pilot, sets $S = 1$ for every action once any sensitive source has been read in the session. *Lineage* lets taint follow the artifacts an action consumes and produces.

Verdicts and evidence. BLOCK halts the run; ESCALATE holds the action for review. With no human in the study we bound the reviewer: *halt* rejects every escalation, *approve* accepts every one, the worst case for prevention. The Message-Action Trace (MAT) is an append-only, SHA-256 hash-chained log of every action, signal, score, verdict and outcome; a replay checker recomputes each verdict from it. Replay is a consistency check, not proof of security.

IV. TESTBED AND EXPERIMENTAL DESIGN

The testbed (`pathgov`, Python 3.12, about 740 lines, no third-party runtime dependencies, no network egress) replays scripted action sequences through the mediator to simulated tools, so intervention authority is the only manipulated variable (Fig. 2). Nine automated tests cover the score surface,

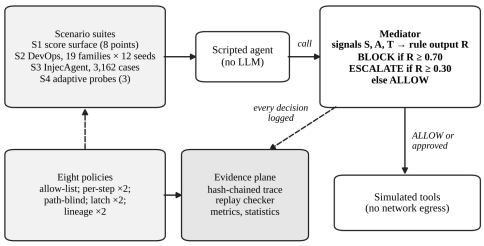

Fig. 2. Testbed. Four suites drive a scripted agent through the mediator to simulated tools; the MAT, replay checker and metrics form the evidence plane.

Table 3
POLICIES COMPARED. PER-STEP POLICIES BLOCK IFF $T = 1$ AND NO (VALID) APPROVAL, AND NEVER ESCALATE; THE OTHER POLICIES SCORE WITH EQ. (1).

Policy	Approval check	Sensitivity memory
Allow-list	none	none
Per-step, asserted	asserted	none
Per-step, validated	validated	none
Path-blind, validated	validated	direct reads only
Latch, asserted (pilot)	asserted	session latch
Lineage, asserted	asserted	lineage
Latch, validated	validated	session latch
Lineage, validated (proposed)	validated	lineage

host classification, approval validation, lineage versus latch, cross-agent handoff, determinism and replay, chain tampering, scenario counts and the InjecAgent mapping.

Suites. S1 checks the eight-point surface. S2 (19 families $\times$ 12 seeds $=$ 228 runs) has eleven attack families (injected read then exfiltration; hijack after legitimate access; encode-then-publish; private repository to paste site; public comment leak; wrong-destination, stale and replayed approvals; cross-agent handoff; destination aliasing; pipeline tampering) and eight benign (routine build; approved internal and partner uses; unapproved low-stakes automation; long internal chain; unrelated approved call after a secret read; approved auditor export; sign-then-release). S3 maps the 544 data-stealing and 510 direct-harm base cases of InjecAgent [11] onto the signals as 2108 attack (plain and forged-approval) and 1054 authorized-twin variants, under an illustrative tool-level taxonomy, a reconstruction rather than an independent annotation, in a narrow and a broad reading. S4 has three adaptive probes.

S2 runs have 1 to 29 steps (mean 4.7). Each family has a fixed skeleton; its 12 seeds vary the attacker and partner host, the number (zero to three) of benign internal steps, and approval lifetimes and nonces. S3 attack variants are plain or carry a forged approval (an approval object present but not scoped to the action); each authorized twin carries a valid in-scope approval. In the broad reading every data-stealing case has $S = T = 1$; in the narrow reading $S = 1$ only for financial, health, identity and access-control tools.

Policies and metrics. Eight policies are compared (Table 3); lineage with validated approvals is proposed. Blocked counts BLOCK only; flagged counts BLOCK or ESCALATE. We report recall, false-block rate (FPR), MCC and payload executed. Instances within a family are correlated, so intervals use a cluster bootstrap over families and tests use exact McNemar on families. Deterministic reruns demonstrate repeatability, not effectiveness. An attack run counts as blocked if its payload step receives BLOCK; a benign run is a false block if any step is blocked; a family counts as blocked only if all 12 instances are. Intervals are 95% percentile intervals from a cluster bootstrap over families (10 000 resamples). **Reproducibility:** every table and figure is generated from one results file written by one command; apart from latency, all results are deterministic; the two InjecAgent files used are identified by SHA-256 checksums, and the nine tests re-derive the scenario counts (132 attack and 96 benign instances), the S3 counts (544 and 510 base cases) and the headline S2 result.

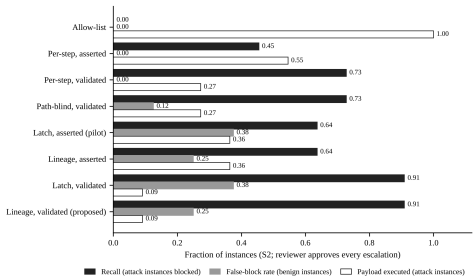

Fig. 3. S2 outcome by policy (reviewer approves every escalation): recall on attacks, false-block rate on benign runs, and share of attack payloads executed.

V. RESULTS

Audit-only versus governed (RQ1). On the same 132 S2 attack instances, audit-only monitoring executed the payload 132 times. With every escalation approved, governed operation executed it 12 times, all pipeline tampering. With every escalation rejected it executed it 0 times, but then only 25% of benign runs completed, because every secret read escalates. The prevention is real; so is the review load.

Detection and cost (RQ3). Table 4 and Fig. 3 use the approve reviewer. Validating approvals lifts per-step authorization from 5 to 8 of 11 attack families (wrong-destination, stale and replayed approvals). Path state adds encode-then-publish and handoff, where the final step carries a genuine, correctly scoped approval and no sensitive flag of its own, giving 10 of 11; Table 5 shows why. Pipeline tampering carries no sensitive or boundary signal and is only escalated. Recall is 0.91 (95% cluster-bootstrap CI 0.73–1.00) and FPR 0.25 (0.00–0.63). Versus the allow-list the gap is significant (10 vs 0 families, $p = 0.002$) but trivial by construction; versus validated per-step it is 2 vs 0 ($p = 0.50$), so the advantage of path state is not statistically established. Against the session latch, lineage keeps recall at 10 of 11 and cuts benign blocks from 3 of 8 to 2 of 8 (the unrelated approved call after a secret read). The remaining two are the approved auditor export (blocked by design) and sign-then-release (conservative taint). Under the flagged reading, 6 of 8 benign families need review (Table 6).

Path memory on InjecAgent (RQ3). Every non-trivial policy blocks all 544 plain data-stealing cases, so path memory adds nothing there. Forged approvals defeat per-step authorization with asserted approvals completely (all 1054 pass);

Table 4
S2, BLOCKED READING, REVIEWER APPROVES EVERY ESCALATION (132 ATTACK, 96 BENIGN INSTANCES). FAM. = FAMILIES BLOCKED; OTHER COLUMNS ARE INSTANCE-LEVEL.

Policy	Atk fam.	Ben fam.	Recall	FPR	MCC	Payload exec.
Allow-list	0/11	0/8	0.00	0.00	–	1.00
Per-step, asserted	5/11	0/8	0.45	0.00	0.51	0.55
Per-step, validated	8/11	0/8	0.73	0.00	0.73	0.27
Path-blind, validated	8/11	1/8	0.73	0.13	0.60	0.27
Latch, asserted (pilot)	7/11	3/8	0.64	0.38	0.26	0.36
Lineage, asserted	7/11	2/8	0.64	0.25	0.38	0.36
Latch, validated	10/11	3/8	0.91	0.38	0.57	0.09
Lineage, validated (proposed)	10/11	2/8	0.91	0.25	0.67	0.09

Table 5
WORKED TRACE, FAMILY A3 (ENCODE-THEN-PUBLISH), SEED 0, FROM THE MAT (STEP 2, A BENIGN BUILD STEP, IS OMITTED).

Step	Per-step, validated	Lineage, validated
1 read secret (vault)	ALLOW	$S{=}1\ A{=}1\ T{=}0$, 0.55, ESC.
3 encode (ci, internal)	ALLOW	$S{=}1\ A{=}1\ T{=}0$, 0.55, ESC.
4 publish (paste host, valid approval)	ALLOW	$S{=}1\ A{=}1\ T{=}1$, 0.77, BLOCK

Table 6
WHICH S2 FAMILIES EACH POLICY BLOCKS (✓ = BLOCKED AT THE PAYLOAD STEP FOR ATTACKS, AT ANY STEP FOR BENIGN). AL ALLOW-LIST; PS PER-STEP; PB PATH-BLIND; LA LATCH; LI LINEAGE; A = ASSERTED, V = VALIDATED APPROVALS. BENIGN FAMILIES B1–B5 ARE BLOCKED BY NO POLICY AND ARE OMITTED.

Family	AL	PS-a	PS-v	PB-v	LA-a	LI-a	LA-v	LI-v
A1 Injected read, exfiltration	–	✓	✓	✓	✓	✓	✓	✓
A2 Hijack after legit. access	–	✓	✓	✓	✓	✓	✓	✓
A3 Encode-then-publish	–	–	–	–	✓	✓	✓	✓
A4 Private repo to paste site	–	✓	✓	✓	✓	✓	✓	✓
A5 Public comment leak	–	✓	✓	✓	✓	✓	✓	✓
A6 Wrong-destination approval	–	–	✓	✓	–	–	✓	✓
A7 Stale approval	–	–	✓	✓	–	–	✓	✓
A8 Replayed approval	–	–	✓	✓	–	–	✓	✓
A9 Cross-agent handoff	–	–	–	–	✓	✓	✓	✓
A10 Destination aliasing	–	✓	✓	✓	✓	✓	✓	✓
A11 Pipeline tampering	–	–	–	–	–	–	–	–
B6 Unrelated call (benign)	–	–	–	–	✓	–	✓	–
B7 Approved auditor export (benign)	–	–	–	✓	✓	✓	✓	✓
B8 Sign-then-release (benign)	–	–	–	–	✓	✓	✓	✓

Table 7
S3 INJECAGENT, BLOCKED COUNTS (ATTACKS $n = 2108$; AUTHORIZED TWINS $n = 1054$). *PATH-BLIND, LATCH AND LINEAGE COINCIDE ON S3. TAXONOMY IS AN ILLUSTRATIVE RECONSTRUCTION, NOT AN INDEPENDENT ANNOTATION.

Policy	Atk-B	Atk-N	Auth-B	Auth-N
Allow-list	0	0	0	0
Per-step, asserted	782 (37%)	680 (32%)	0	0
Per-step, validated	1564 (74%)	1360 (65%)	0	0
Scorer, asserted*	1700 (81%)	1207 (57%)	748 (71%)	442 (42%)
Scorer, validated*	1904 (90%)	1530 (73%)	748 (71%)	442 (42%)

Atk/Auth = attack/authorized cases; B/N = broad/narrow taxonomy.

Table 8
S2 ABLATIONS OF THE PROPOSED POLICY (BLOCKED READING, REVIEWER APPROVES).

Variant (lineage, validated)	Atk fam.	Recall	FPR
Full	10/11	0.91	0.25
No boundary signal	5/11	0.45	0.00
No approval signal	7/11	0.64	0.25
No sensitivity signal	8/11	0.73	0.00
Path-blind	8/11	0.73	0.13
Own-agent history only	9/11	0.82	0.25
Substring host matcher	10/11	0.91	0.25

validation closes this (Table 7). Under the broad taxonomy 204 attack cases (578 narrow) are only escalated: effectors such as unlocking a door or placing an order carry neither signal. The cost is stark: 748 of 1054 authorized twins are blocked under the broad reading (442 narrow), so deployment needs a stronger approval class for legitimate transfers of sensitive data.

Ablations. Every signal matters on S2 (Table 8). Block thresholds 0.55–0.70 behave identically; 0.75 drops recall to 0.64 and 0.80 or more to 0.45, because the 0.74 point of the surface (external, no approval) falls below the threshold. The substring host matcher changed nothing on S2 because the sensitive signal alone blocks the aliasing family.

Replay, integrity and latency (RQ2). All 1000 unhashed edits were caught by the chain. Of 1000 re-hashed edits the chain passed all and replay caught 730 (Table 9); the rest changed no verdict-relevant field (for example an approval expiry shifted by one tick). In 120 coherent forgeries, where a payload destination was rewritten to an internal host and the trace regenerated, chain and replay both passed 120 of 120; the head hash differed in every case. The log is therefore tamper-evident only if its head is anchored outside the mediator's trust domain. In-process decisions take a median of about 7 µs (p99 about 20 µs) and about 26 ms per 1000 logged steps on one virtual CPU; five repeated runs gave medians of 6.7 to 11.4 µs, so latency is the one result that varies between runs. It excludes network, model and sandbox cost and is not production latency.

Adaptive probes (RQ4). An internal relay that forwards outward, reached with a valid approval, evades every policy as a block (it is only escalated). An untracked channel (a secret stored in the environment, re-read as a fresh artifact, published with a valid approval) evades lineage and per-step authorization but is caught by the session latch: the price lineage pays for fewer false blocks. A secret split into chunks is blocked by all three (Table 10).

Table 9
RE-HASHED EDITS ($n = 1000$): EDITS CAUGHT BY REPLAY, BY FIELD.

Field edited	Edits	Caught by replay
verdict	143	143
score	143	143
signals	143	143
destination	143	131
approval dropped	143	130
tool name	143	40
approval expiry (+1 tick)	142	0
Total	1000	730

Table 10
S4 ADAPTIVE PROBES: IS THE PAYLOAD STEP BLOCKED?

Probe	Per-step, val.	Latch, val.	Lineage, val.
P1 internal relay, valid approval	no (allow)	no (esc.)	no (esc.)
P2 untracked channel, valid approval	no	yes	no (esc.)
P3 chunked exfiltration, no approval	yes	yes	yes

VI. DISCUSSION AND LIMITATIONS

What the evidence shows. The boundary that matters is the move from proposal to execution: a post-execution detector can identify the send but not retract it. Most of the gain over per-step authorization came from validating approvals; path memory closed a smaller class (transformed and cross-agent exfiltration), and lineage reduced, but did not remove, the over-blocking of session memory. A practical reading is to implement approval validation first and add path memory where agents transform or pass on retrieved secrets. The cost concentrates where approval cannot clear a sensitive crossing: production use needs a stronger approval class, for example bound to a specific artifact and destination, which we do not evaluate.

Lineage versus the latch. Lineage removes one class of false blocks (unrelated calls after a secret read) at the price of exposure to untracked channels (probe P2). If a deployment can instrument every channel through which an artifact flows, lineage is preferable; if it cannot, the latch is the conservative choice.

Trace integrity. A hash chain and replay checker detect accidental or naive edits but not a coherent forgery: after a payload destination is rewritten and the trace regenerated, both pass, yet the head hash differs. Tamper-evidence therefore holds if the head is anchored outside the mediator's trust domain, for example in a separate store or a signed checkpoint. This is a requirement implied by the experiment; no anchored store was built or tested.

What this does not prove. Labels, scenarios and constants are ours and no blind second annotator exists, so results may reflect agreement between labelling and policy. The InjecAgent tool taxonomy is a reconstruction chosen to be defensible by domain, not an independent annotation, so S3 counts demonstrate the mechanism and are not independent evidence; an independent, ideally two-annotator, labelling is future work. The planner is scripted: false-block and attack-success rates under an LLM agent, paraphrase and content-level injection are unmeasured, and AgentDojo was not run. Signal extraction is untested. Eleven attack and eight benign families give wide intervals, and instances are not independent. Reviewer models are bounds, not observed people; no production authorizer was compared; the multi-agent scope is one handoff pattern. The mechanism does not cover integrity attacks and needs a fourth signal for privileged or irreversible changes.

VII. CONCLUSION

On author-built DevOps workloads, a deterministic pre-dispatch mediator prevented all but one class of payload that audit-only monitoring allowed, blocking 10 of 11 attack families at the price of 2 of 8 benign blocks and 6 of 8 escalations. It does not cover integrity attacks, blocks legitimate approved transfers of sensitive data, and is tamper-evident only with an externally anchored log head. Next: blind relabelling, LLM-driven agents with recorded model, prompt and seed, an AgentDojo mapping, held-out threshold calibration, an anchored trace store, a production-authorizer comparison and third-party replication.

ACKNOWLEDGMENTS

Funding: the research was self-funded; no external funding was received.

DECLARATIONS

Ethics and Consent to Participate declarations: not applicable. The data are synthetic or from the public InjecAgent benchmark; no human participants; no network egress. **Competing interests:** none declared. **Related submission:** the pilot [6] was submitted to RELIABLE-SE 2027 and is under review; this paper extends it as described in Section II. **Use of AI tools:** code, scripts and a draft of this text were prepared with LLM assistance and reviewed by the authors, who take responsibility for the content. **Data and code availability:** `pathgov`, the scenario generators, tests and results are available from the authors on request and will be deposited in a public archive with a permanent identifier on acceptance. **Author contributions:** K. M. Pradhan conceived the study, designed the experiments and prepared the manuscript; D. P. Mahapatra supervised the research direction and reviewed the manuscript.

REFERENCES

[1] Debenedetti E, Zhang J, Balunović M, Beurer-Kellner L, Fischer M, Tramèr F. 2024. AgentDojo: a dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *NeurIPS 37 (Datasets and Benchmarks)*.

[2] Greshake K, Abdelnabi S, Mishra S, Endres C, Holz T, Fritz M. 2023. Not what you've signed up for: compromising real-world LLM-integrated applications with indirect prompt injection. In *AISec '23*. ACM.

[3] Kaptein M, Khan V-J, Podstavnychy A. 2026. Runtime governance for AI agents: policies on paths. arXiv:2603.16586.

[4] Ligatti J, Bauer L, Walker D. 2005. Edit automata: enforcement mechanisms for run-time security policies. *Int J Inf Secur* 4(1–2):2–16.

[5] Paduraru C, Bouruc P-L, Stefanescu A. 2026. A trace-based assurance framework for agentic AI orchestration. In *ENASE 2026*. doi:10.5220/0014840300004015.

[6] Pradhan KM. 2026. Path-aware runtime governance for agentic tool calls: a reproducible pilot of pre-execution enforcement and replayable action traces. Submitted to RELIABLE-SE 2027 (co-located with ISEC 2027), under review.

[7] Ruan Y, Dong H, Wang A, Pitis S, Zhou Y, Ba J, Dubois Y, Maddison CJ, Hashimoto T. 2024. Identifying the risks of LM agents with an LM-emulated sandbox. In *ICLR 2024*.

[8] Saltzer JH, Schroeder MD. 1975. The protection of information in computer systems. *Proc IEEE* 63(9):1278–1308.

[9] Schneider FB. 2000. Enforceable security policies. *ACM TISSEC* 3(1):30–50.

[10] Uchibeke U. 2026. Before the tool call: deterministic pre-action authorization for autonomous AI agents. arXiv:2603.20953.

[11] Zhan Q, Liang Z, Ying Z, Kang D. 2024. InjecAgent: benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of ACL 2024*, pp. 10471–10506. Test cases: github.com/uiuc-kang-lab/InjecAgent (accessed 14 Nov 2026).