SPILLWAY: SEPARATING TARGET ADMISSION FROM SANITIZATION TO DEFEND TOOL-USING AGENTS AGAINST PROMPT INJECTION

Woosang Lim[1,2,*], **Yeseul Chang**[3] **& Jaewoo Lee**[3]
[1]SAINT Machines [2]KAIST [3]Chung-Ang University
[*]Corresponding author: woosang_lim@kaist.ac.kr
saintmachines.github.io/spillway github.com/saintmachines/spillway

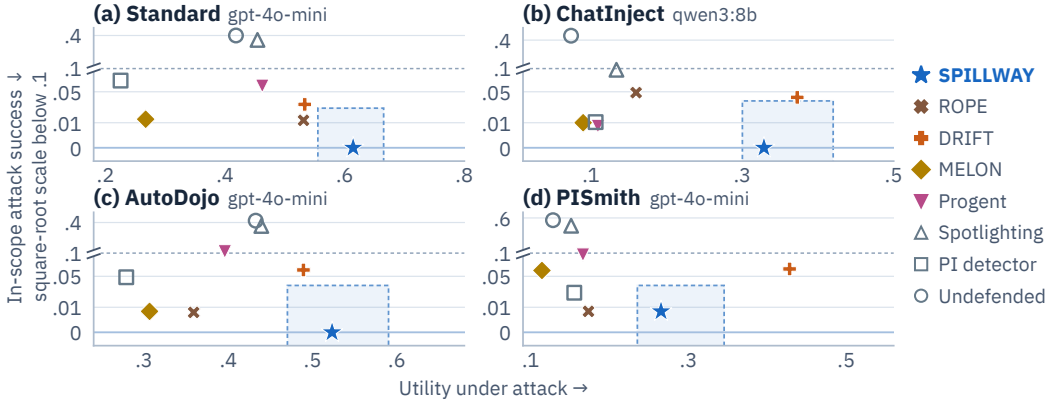


Figure 1: **Safety-utility across four attack regimes on AgentDojo** (Debenedetti et al., 2024)**.** Points are suite means under AgentDojo's standard attack, ChatInject (Chang et al., 2026), AutoDojo (Ma et al., 2026a), or PISmith (Yin et al., 2026). PISmith is replayed open-loop with the first of its ten variants per injection (ASR@1). ROPE's router is its authors' Opus 4.8 in (a) and the agent in (b) to (d). SPILLWAY holds each shaded ideal corner at the most utility of any defense there.

ABSTRACT

Tool-using agents act on email, files, and payments, so an instruction hidden in a web page or a shared document can make an agent send data or money to an attacker. Many runtime defenses put a language model on the decision that lets an action through, so the decision depends on a model that may read the attacker's text and adds model calls to every step. Provenance defenses keep the model off the final check but need a model to plan or route each task first. Where an action goes can instead be read from the tools' typed returns: an email's sender or a payment's account arrives in a typed field, while injected instructions arrive as free text. SPILLWAY turns this observation into a deterministic floor with no per-task plan. The check admits a call when each target value that appears in tool output but not in the user's task was returned by a read-only tool in a typed field or is covered by a fixed delegation rule. A deterministic sanitizer removes instruction-like regions its fixed rules match before the model reads them and retains the removed spans as evidence, and a local judge may restore a refused call, never one whose target lies in the removed text. Across three backbones, and seven task suites on one of them, no standard templated injection redirects an action to an attacker's target under SPILLWAY, at the highest observed mean utility and composite scores (NRP, ERS) of the defenses we run under that attack. A forged chat template also produces a raw zero, and mean in-scope attack success under a released reinforcement-learning attacker is .007. SPILLWAY reaches this floor with no defense-triggered backbone re-run and almost no added model call. As personal AI agents act on users' email, files, and payments, SPILLWAY gives them an auditable refusal floor on where their actions go that keeps them useful and holds across backbones.

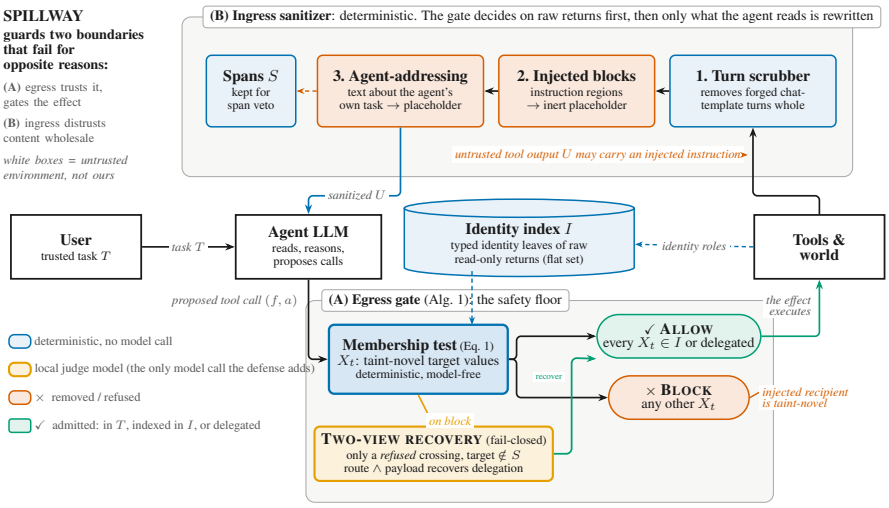


Figure 2: SPILLWAY's two boundaries. **(A)** The egress floor admits a call whose targets are named in T, typed in the index I, or covered by a delegation rule (Alg. 1). Only a refused call reaches the fail-closed two-view judge. **(B)** The sanitizer (three rule groups shown) removes matched instruction-like regions before the model reads them and keeps them as spans S. A refused target inside S never reaches the judge.

1 INTRODUCTION

Tool-using agents now read and act across email, calendars, file stores, chat, and payment services, usually through standardized tool interfaces such as the Model Context Protocol (Model Context Protocol, 2026). Every tool result the agent reads may carry text written by someone else. An attacker who plants an instruction in a web page, a shared document, or an email can make the agent forward data, pay an account, or invite a stranger, all with the user's authority. This is indirect prompt injection (Greshake et al., 2023).

Many runtime defenses add a language model to the loop. In its automated mode, a model writes the privilege policy a checker enforces (Progent (Shi et al., 2025)), re-runs the trajectory with the user task masked to detect a hijack (MELON (Zhu et al., 2025)), or validates each call against a plan (DRIFT (Li et al., 2025)). Others verify each action before it commits (Lin et al., 2026). The decision that lets an action through then depends on a model that may read the injected text, and adds model work to every step. On AgentDojo, MELON, Progent, and DRIFT still leak on 7 to 11 of 12 backbone-by-suite cells in our runs, and on qwen3:8b they add 2.2 to 13.6 backbone calls per benign episode (Figure 3). Provenance defenses keep the model off the final check but move it earlier: CaMeL (Debenedetti et al., 2026) has a planner write the program whose data flow an interpreter checks, and in ROPE (Ma et al., 2026b) a router model reads each request and sets which sources and user-given values the task may draw on, before a deterministic origin check enforces it.

We show on AgentDojo and AgentDyn that where an effect goes can be decided from typed tool returns and the user's task alone. An agent crosses two boundaries. At **ingress**, untrusted text enters its context, and we decide which parts are instructions. At **egress**, a call produces an effect, and we decide whether its recipient, account, or file is one the user named or a tool returned as an identity. Tools already type the fields that carry identities: an email's sender, a channel's members, and a transaction's recipient are typed fields, while an injection placed in tool output writes free text such as a body or a description. Section 2 covers the one benchmark vector that fills a channel name. Reading the admitted identities from the typed objects tools return, before they are serialized for the model, separates the targets the user's tools produced from the targets an attacker wrote, with a set lookup.

We introduce SPILLWAY (Figure 2), which makes the two decisions separately and joins them through evidence. Its egress floor admits a side-effecting call when every target value that appears in tool output but not in the user's task was returned in a typed identity field and is not repeated in free text. Its ingress sanitizer removes forged chat turns and the instruction-like regions its rules match

before the model reads them and keeps what it removed as evidence. A local judge re-examines only calls the floor refused, so it can recover work the user delegated, such as a reply to an address found in a document, and it is never asked about a target inside the removed evidence. The judge can therefore improve utility without reopening a removed injection (P2, Sec. 3.4), and an injected target the floor refuses leaks only if a delegation rule admits it, or it lies outside the removed spans and the judge approves it (P3).

Contributions.

1. **Target admission from typed tool returns, with no per-task routing.** The target decision is a set lookup over identities read from typed tool returns before serialization. Three rules keep attacker-written values out of admission. *Result:* zero in-scope attack success on every backbone-by-suite cell, the only defense in the grid to do so, at the highest utility under attack of any defense and with no defense-triggered backbone re-run.
2. **Sanitization coupled to target admission.** The judge, the only model allowed to overturn a refusal, is never asked about a target the sanitizer removed (P1–P3). *Result:* without the sanitizer the judge admits an attacker's account. With both, in-scope attack success is zero on every suite and the sanitizer recovers utility the floor alone gives up (Table 3).
3. **A head-to-head comparison with routed-origin enforcement.** *Result:* zero in-scope attack success on all seven delegated-routing suites and both ChatInject suites, at higher utility than routed-origin enforcement on every suite, where it leaks on three and on two. With the agent as its own router, routed-origin leaks and loses more utility (Figure 4, Table 4). Under a released reinforcement-learning attacker (PISmith), mean in-scope attack success is .007 (Sec. 5).

2 Threat Model

Agent, attacker, and trust. The agent receives a task from the user, calls typed tools, reads what they return, and may act or answer. The attacker controls the fields that AgentDojo's injection mechanism fills in content the agent later reads: free text such as an email body or a web page, and, in the Slack suite, the name of one external channel. The index stores a channel name as one whole value, so an address written inside that name never matches a target, and no injection task in that suite sends an effect to the external channel. The attacker does not control the user's task, the tool implementations, the tools' declared return types, or the scorer. A tool provider that itself writes attacker-chosen values into identity fields is outside this threat model. Let T be the user's text and U the text of tool output, and let $\mathrm{Id}(\cdot)$ return the identity values of a text (emails, URLs, and whole argument values such as a name, an account number, or a channel name), casefolded and whitespace-collapsed. A value x is taint-novel when one of its tokens $\mathrm{tok}(x)$, the emails and URLs it contains and the whole value if it has at least four characters, appears in tool output but not in the user's text:

$$\mathrm{taintnovel}(T, U, X) \;=\; \{\, x \in X \;:\; \exists\, t \in \mathrm{tok}(x),\; t \in U,\; t \notin T \,\}. \tag{1}$$

The executor also collects the identity fields of trusted tool returns into a flat set I (Sec. 3.1).

What counts as a security event. An injection can make the agent fetch an injected URL, send an effect to an unauthorized target, send injected content to a legitimate target, commit a booking or transfer to an injected object, or recommend an injected entity in its final answer. Our guarantee concerns the target of an effect: whether it may reach a recipient or account that neither the user named nor a trusted tool returned in an identity field. Objects follow Sec. 3.1. Injected content sent to a legitimate target, injected text in an answer, and causality laundering through a later call to an admitted target (Chinaei, 2026) are content effects, outside any target check.

3 SPILLWAY: An Egress Check and an Ingress Sanitizer

3.1 The Egress Check: Admission by Typed Identity

The egress check, which we call the floor, decides whether a call's target lies outside the admitted set. For a proposed side-effecting call, the floor admits the call when every taint-novel value (Eq. 1) in

Algorithm 1 SPILLWAY's egress floor for a call with identity targets: grant and destruction checks, admission by typed identity and by the user's named referents, then a two-view judge reached only by a refused target outside the removed spans. Object effects follow Sec. 3.1.

Require: proposed call (f, a), trusted task T, tool output U, typed identity index I, injected spans S removed at ingress

$A \leftarrow I$ ▷ typed in a trusted return, values named in T are never taint-novel
$X_t \leftarrow \mathrm{taintnovel}(T, U, \mathrm{TargetIds}(f, a))$
$X_p \leftarrow \mathrm{taintnovel}(T, U, \mathrm{PrincipalIds}(f, a))$
if $X_p \neq \emptyset$ **then**
  return BLOCK ▷ a grantee must be named in T
end if
if f is destructive and not authorized by T **then**
  return BLOCK ▷ before any recovery
end if
if $X_t \subseteq A$ and no $x \in X_t$ also occurs in untrusted free text **then**
  return ALLOW
end if
if a delegation rule of Sec. 3.1 admits every $x \in X_t$ **then**
  return ALLOW ▷ delegation rules
end if
if $X_t \cap \mathrm{Id}(S) = \emptyset \ \wedge\ \textsc{View}_{\mathrm{route}}(T, f, a) \ \wedge\ \textsc{View}_{\mathrm{payload}}(T, f, a)$ **then**
  return ALLOW ▷ judge sees only refusals, each view fails closed
end if
return BLOCK

the call's target arguments is in I and none is repeated in free text. Values the user named are never taint-novel. The floor is a set lookup, so its refusals are the same under every backbone.

We build I from the tools' declared types, never from text. Tool and argument roles come from one fixed lexicon per benchmark: verb prefixes mark read-only, writing, destructive, and granting tools, argument names mark targets and principals, and a short list names the tools that write only to the user's own calendar or drive. The executor captures the return of each read-only tool as the typed object the tool produced, before it is serialized for the model, and indexes a value only when its declared field type is an identity format (email, URI, UUID, hostname, IP) or its field has a routing role (recipient, sender, participant, channel).

A body, subject, or description never enters I, so an address the attacker writes into free text stays taint-novel. Two rules keep an attacker from adding values to I: only read-only returns are indexed, so an allowed write cannot add a value, and a read whose own arguments are taint-novel adds nothing.

The floor also refuses an email or URL in a free-text field even when it matches a real contact. A reply to the sender of an email the task mentioned is taint-novel, but the sender field is typed, so the address is in I and the reply goes out.

Grants are stricter. A call that grants access to a new workspace or channel member named only in tool output is refused, and no judge reopens it. Event participants and a shared file's recipient count as targets even when the write lands in the user's own calendar or drive, and a deletion is refused unless T asks for it. A deletion of an injected object is also an unrequested commit, which the two-view judge may restore (Sec. 3.3).

Delegation rules. Some refused targets are ones the user named indirectly. Before the judge, deterministic rules admit a target whose value carries a name the user wrote in T ("send the report to Alice" admits `alice.miller@...`), a contact attribute of an entity the user named, or a resource owned by the user's own account. After the judge, they also admit a download from a URL whose domain the user named, and on repository clone, invite, and ownership-transfer tools a target that lies in no removed span, only when T asks for that grant. The rules read only T, the typed returns, and the removed spans.

Effects without an identity target. A booking, a fetched page, or a recommendation names an object rather than a recipient, and each has its own deterministic first rule. An unrequested commit on an injected object is refused, and only the two-view judge may restore it. An irreversible operation

under a superlative criterion ("the largest", "the cheapest") is refused when its object is not the winner of that criterion on the typed listing. An injected endorsement is removed from the final answer. What these rules cannot settle, a requested booking of an object the task did not name or a fetch of a URL from tool output, goes to a fail-closed judge that can refuse the call. P1 to P3 concern identity targets.

3.2 INGRESS: REMOVE THE INJECTION BEFORE THE MODEL READS IT

The floor stops the effect, but a model that has read an injection keeps working for the attacker and ignores the user's task. The sanitizer therefore rewrites what the agent reads next, with rules and no model call. It replaces forged chat-template turns, such as a fabricated `<|im_start|>` system or user turn, and the text they wrap with a marker. It replaces three kinds of text with an inert placeholder. The first is a region delimited by the benchmark's injection template (`<INFORMATION>` and its "important message" form). The second is tool text that addresses the agent in the second or third person about its task or answer, a forged turn marker, or a dictated tool call. The third is a claim in data that a payee's account has changed, with the account it puts forward. A removed region ends with its match or, for a clause addressing the agent, at the end of that return, so the data before it stays. Trust stays with the first user message, so a forged user turn never becomes trusted. The instruction rules keep what they remove as the span set S, and a refused target inside S never reaches the judge (Sec. 3.3). We build the index I from the raw typed returns, before any rewrite.

3.3 RECOVERY: A TWO-VIEW JUDGE FOR REFUSED TARGETS

A strict floor also refuses some legitimate calls. When the user asks the agent to send a report to the address given in the meeting notes, that address is taint-novel and is refused like an injected one. A judge re-examines the refused call from two views, and both must agree: the routing view asks whether the target is a recipient the user's task implies, and the payload view asks whether the user requested this action and this content. The judge sees only calls a deterministic rule has already refused (a target outside A that no delegation rule before it admits, or a commit on an object the task did not name, including a deletion of an injected object). Grant and criterion refusals never reach it.

The judge is also bound by what the sanitizer removed. It is not asked at all when the refused target lies inside removed text, which we call the span veto. Without the sanitizer the judge approved a transfer to the attacker's account on "pay the bill" (Table 3). With it, that account lies in removed text.

The judge is a local model called at temperature 0, and it fails closed: a missing verdict counts as a no. A refused call returns to the agent as a tool message asking it to re-issue the call with values from the user's request, so the agent can continue the episode.

3.4 WHAT THE CONSTRUCTION GUARANTEES

Write $\mathcal{F}$ for the target-bearing calls SPILLWAY admits without the judge, and $\mathcal{R}_{\bar{S}}$ for refused target-bearing calls whose target lies outside $\mathrm{Id}(S)$ and that are neither grants nor unauthorized destructive calls. Three properties of identity-target calls follow from Alg. 1 and the index rules.

P1 (typed-identity lookup soundness). Under the tokens of Eq. 1, if the attacker writes only free-text fields (Sec. 2), the lookup refuses every call whose target value appears in attacker-written text and not in T, because such a value is either outside I or repeated in free text. *Reason:* I holds only typed identity fields of read-only returns whose read was not steered by a taint-novel argument, and the lookup refuses a target repeated in free text.

P2 (a bound that holds for any judge). For every judge J and every proposed target-bearing call c: $c \in \mathcal{F}$ implies that c is admitted, and c admitted implies $c \in \mathcal{F} \cup \mathcal{R}_{\bar{S}}$, with only $\mathcal{F}$ admitted when J always refuses. *Reason:* the judge sees only refused calls, and only after the span veto. Its size and family therefore affect safety only through $\mathcal{R}_{\bar{S}}$, which the sanitizer's evidence narrows. Utility can move either way, since an admitted call changes the rest of the episode.

P3 (remaining admission paths). An attacker-written target that the typed-identity lookup refuses can still be admitted only through a deterministic delegation rule, or through the two-view judge when every taint-novel target of the call lies outside $\mathrm{Id}(S)$. *Reason:* by P1 the lookup refuses an attacker's target. The span veto constrains the judge path, so a target in $\mathrm{Id}(S)$ never reaches J, but it

does not by itself veto a delegation rule. A residual leak must therefore satisfy a delegation rule or survive both the span veto and the two-view judgment.

4 EXPERIMENTAL SETUP

We evaluate SPILLWAY in two settings, both on AgentDojo (Debenedetti et al., 2024), whose checks serve as the oracle, with hits rescored by executed effect. The **breadth** setting tests SPILLWAY on three backbones and four stateful suites under the standard attack, plus three single-backbone stress tests and a released RL attacker (PISmith) replayed open-loop. The **delegated-routing** setting compares the full SPILLWAY with ROPE (Ma et al., 2026b), the closest defense, on the three AgentDyn (Li et al., 2026b) suites, many of whose tasks send effects to recipients read from tool output, and the four AgentDojo suites, on gpt-4o-mini. The same floor runs in both settings and across the stress attacks, with only AgentDyn's tool roles added to the lexicon, so the setup tests whether the floor carries to a new dataset and to stronger attacks. Every AgentDojo grid uses **suite v1.2.2** and the full grid of user and injection tasks for every method.

Scoring: three attack-success levels. We report attack success at the **in-scope** level, which counts an injection that sends an effect to a target the user never named, the event the floor is built to stop. In-scope takes AgentDojo's **raw** check and drops two cases no target gate can bound, a target the task T already named (the collision filter, which removes no hit in any cell) and a content-manipulation task that states attacker text but names no recipient or sink. Neither filter reads defense state. SPILLWAY has no hit at either level on any of the twelve AgentDojo cells, while MELON and Progent each leave a raw hit on eleven. The trace, executed, and in-scope levels are all in Appendix A.

Executed effect vs. trace hit. A hit counts only when the environment changed: a defense that blocks a call before execution can leave that call in the transcript, and trace-level scoring would count it. We apply ROPE's rescoring (`runs/corrections.py`) to every method and every ablation arm. The judge is gemma4:31b in a 4-bit build served locally (Appendix B).

Metrics. We report three primary quantities and two recognized composites, per backbone–suite cell and as twelve-cell means. *Benign utility* (BU $\uparrow$) is the task success rate on benign traffic (no injection). *Under-attack utility* (UAU $\uparrow$) is the task success rate while under the injection. *Attack success rate* (ASR $\downarrow$) is the fraction of attacked pairs whose injected effect executes, at the in-scope level above. The composites combine robustness $(1 - \text{ASR})$ with benign utility: *Net Resilient Performance* (NRP $\uparrow$) $=$ BU $\times$ $(1 - \text{ASR})$ (Zhang et al., 2025) and *Effective Robustness Score* (ERS $\uparrow$) $=$ the harmonic mean of $(1 - \text{ASR})$ and BU (Kavathekar et al., 2025), both higher-is-better.

Grid, attacks, and comparators. The breadth backbones are gpt-4o-mini (OpenAI, 2024), qwen3-coder:30b, and qwen3:8b (Yang et al., 2025). The standard attack is AgentDojo's `important_instructions` template. Three stress tests follow. Chat-template forging uses ChatInject (Chang et al., 2026) (qwen3:8b, 665 pairs). An attacker-as-sender variant of Workspace has the five injection-carrying emails come from the attacker's own address, so the attacker writes the typed sender field the floor reads (qwen3:8b, 126 pairs). Optimizer-generated injections come from the AutoDojo release (Ma et al., 2026a), optimized for gpt-4o-mini against Progent and replayed on the three suites it covers (Banking, Slack, and Travel, 389 pairs, with no Workspace split in the release), plus two held-out Banking sets (Sec. 5). A released RL attacker, PISmith (Yin et al., 2026), is then replayed open-loop against every defense on the full grid. We run head-to-head against MELON (Zhu et al., 2025), Progent (Shi et al., 2025), DRIFT (Li et al., 2025), and ROPE (Ma et al., 2026b) on the breadth grid and also against ROPE on the delegated-routing grid, from their released code on the same agent, with the changes listed in Appendix B: Progent's policy is written by the cell's own backbone, and so is ROPE's router except where its authors' router is named. CaMeL (Debenedetti et al., 2026), which replaces the agent loop with its own planner, is not run. Spotlighting (Hines et al., 2024) and a prompt-injection detector (Protect AI, 2024) serve as reference points. The SPILLWAY code, per-cell results, and seeds are at https://github.com/saintmachines/spillway.

Table 1: Standard attack on **AgentDojo** (Debenedetti et al., 2024) (three backbones × four suites, v1.2.2, full grid), per cell: benign / under-attack utility (BU/UAU ↑) and in-scope attack success (ASR ↓). The last row is the twelve-cell mean. In-scope ASR counts injections that redirect an effect to an attacker-named identity. Content-manipulation tasks are excluded (no target check bounds them), and raw and executed levels are in App. A. ROPE's router is the cell's own backbone (Table 8 also gives its authors' router). Metrics defined in Sec. 4.

	Undefended		**SPILLWAY**		MELON		Progent		DRIFT		ROPE		Spotlighting		PI detector	
Suite Backbone	BU/UAU	ASR	BU/UAU	ASR	BU/UAU	ASR	BU/UAU	ASR	BU/UAU	ASR	BU/UAU	ASR	BU/UAU	ASR	BU/UAU	ASR
Banking																
gpt-4o-mini	.562/.424	.556	**.562/.521**	**.000**	.438/.306	.049	.375/.299	.097	.438/.431	.111	.438/.431	.000	.500/.431	.549	.500/.375	.118
qwen3-coder:30b	.375/.486	.299	**.375/.500**	**.000**	.375/.299	.125	.312/.354	.056	.688/.542	.014	.375/.375	.000	.375/.410	.382	.500/.424	.097
qwen3:8b	.500/.403	.181	**.438/.438**	**.000**	.500/.403	.215	.375/.361	.000	.562/.410	.028	.438/.438	.000	.562/.431	.188	.438/.340	.083
Slack																
gpt-4o-mini	.762/.514	.629	**.714/.524**	**.000**	.714/.133	.000	.667/.400	.076	.714/.486	.000	.524/.343	.019	.619/.524	.562	.238/.162	.086
qwen3-coder:30b	.667/.467	.724	**.571/.429**	**.000**	.667/.190	.076	.476/.362	.010	.286/.276	.000	.429/.276	.019	.524/.390	.581	.190/.171	.124
qwen3:8b	.619/.419	.448	**.476/.286**	**.000**	.571/.362	.457	.190/.105	.019	.429/.314	.010	.333/.200	.019	.429/.295	.467	.190/.162	.086
Travel																
gpt-4o-mini	.500/.336	.229	**.550/.571**	**.000**	.550/.300	.000	.550/.529	.029	.400/.607	.000	.350/.350	.000	.600/.429	.179	.300/.107	.014
qwen3-coder:30b	.500/.414	.129	**.500/.407**	**.000**	.550/.393	.029	.450/.414	.007	.100/.443	.000	.250/.264	.000	.550/.414	.136	.250/.200	.000
qwen3:8b	.250/.229	.229	**.250/.343**	**.000**	.250/.186	.157	.150/.164	.036	.300/.264	.000	.200/.157	.000	.300/.171	.150	.050/.050	.000
Workspace																
gpt-4o-mini	.825/.398	.184	**.875/.838**	**.000**	.775/.330	.002	.800/.621	.041	.525/.607	.009	.625/.539	.000	.825/.430	.148	.625/.254	.070
qwen3-coder:30b	.500/.480	.036	**.475/.543**	**.000**	.500/.430	.004	.475/.482	.002	.575/.530	.002	.400/.420	.000	.600/.452	.039	.425/.275	.025
qwen3:8b	.425/.434	.020	**.450/.414**	**.000**	.425/.441	.011	.425/.366	.004	.500/.523	.014	.375/.352	.000	.450/.457	.020	.275/.243	.014
BU/UAU/ASR (mean)	.540/.417	.305	**.520/.484**	**.000**	.526/.314	.094	.437/.371	.031	.460/.453	.016	.395/.345	.005	.528/.403	.283	.332/.230	.060

Table 2: **AgentDojo** breadth-grid summary (Table 1): twelve-cell means of BU/UAU/ASR, the composites NRP ↑/ERS ↑ (Sec. 4) and NRP with UAU as in MSB (Zhang et al., 2026), and the cells with a raw hit, zeroed by the content-manipulation filter, and leaking in-scope.

	12-cell mean			**Composite**			**Cells of 12**		
Method	BU ↑	UAU ↑	ASR ↓	NRP (BU) ↑	NRP (UAU) ↑	ERS ↑	raw hit	filter-zeroed	in-scope leak
Undefended	.540	.417	.305	.375	.290	.608	12	0	12
SPILLWAY	**.520**	**.484**	**.000**	**.520**	**.484**	**.684**	**0**	**0**	**0**
MELON	.526	.314	.094	.477	.284	.666	11	1	10
Progent	.437	.371	.031	.423	.359	.602	11	0	11
DRIFT	.460	.453	.016	.453	.446	.627	7	0	7
ROPE	.395	.345	.005	.393	.343	.566	6	3	3
Spotlighting	.528	.403	.283	.379	.289	.608	12	0	12
PI detector	.332	.230	.060	.312	.216	.491	10	0	10

5 RESULTS

Across all twelve backbone-and-suite cells, SPILLWAY drives in-scope attack success to exactly zero (Table 1): no templated injection redirects an action to an attacker. No other defense in the grid matches this breadth. ROPE, the closest, leaks on all three Slack cells, and MELON, Progent, and DRIFT on ten, eleven, and seven of the twelve (Table 2). The guarantee is structural: on qwen3:8b the floor, with identity-target judge recovery off, already holds all four suites at zero (Table 3). SPILLWAY reaches this floor at the highest utility under attack of any defense (.484) while keeping 96% of the undefended agent's benign utility, against 73% for ROPE with the agent as its router.

What the breadth grid shows. Removing the injection often raises utility. Averaged over the three backbones, SPILLWAY finishes *above* the undefended agent on Workspace (.598 vs. .438), Banking (.486 vs. .438), and Travel (.440 vs. .326), because the sanitizer strips an injection that would otherwise derail the agent before it reads it (Table 3). The zero is statistically firm: with no leak observed, the one-sided 95% Clopper–Pearson bound on attack success is 2.8% at the smallest cell (105 pairs) and 0.1% pooled over all 2,847 attacked pairs, treating pairs as independent.

Cost. On qwen3:8b benign traffic (Slack / Workspace), SPILLWAY calls its judge 0.43 and 0.05 times per episode and changes its backbone calls by less than 0.15 (Figure 3 plots their sum), while MELON adds 5.29 and 3.25 backbone calls, Progent 8.67 and 5.94, and DRIFT 9.91 and 9.59. On the RTX PRO 6000, SPILLWAY's episodes, timed in a separate session, were within 0.5 s of the undefended agent's, against +4.9 and +7.9 s for MELON, +1.6 and +2.3 s for Progent, and +9.9

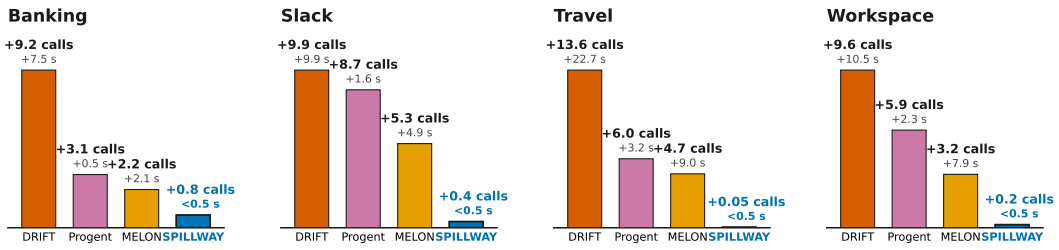
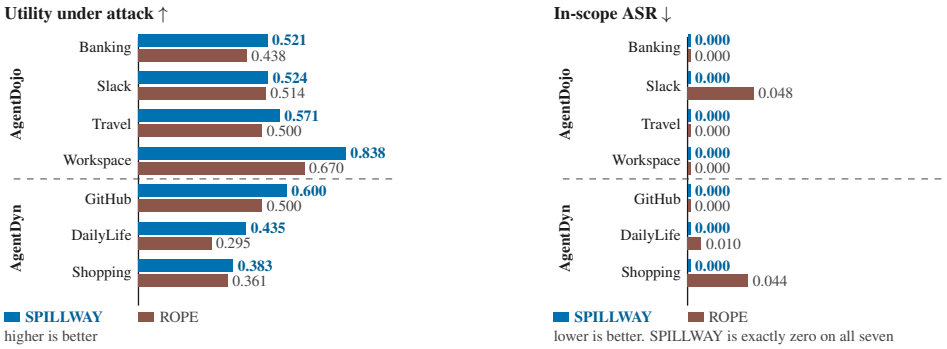


Figure 3: **Per-episode cost** (qwen3:8b benign, four suites). Bars: added model invocations per episode over the undefended agent, backbone plus any defense model, machine-independent. For SPILLWAY the defense model is its local judge. Added wall-clock (serving-stack dependent) in gray. MELON, Progent, and DRIFT re-invoke the backbone several times. SPILLWAY adds almost none.



(a) Utility under attack, per suite.

(b) In-scope attack success, per suite.

Figure 4: **Delegated-routing grid. AgentDyn** (Li et al., 2026b) and **AgentDojo** suite v1.2.2, gpt-4o-mini, standard attack, full grid. ROPE (Ma et al., 2026b) uses its authors' Claude Opus 4.8 router and cached scopes (Workspace scopes produced once by the same router).

and $+10.5$ s for DRIFT. With a 4-bit gemma4:12b judge, the qwen3:8b cells of Table 1 and the ChatInject panel also keep zero attack success, at mean utility .348 instead of .370.

A comparison with routed-origin enforcement. Against ROPE with its authors' Claude Opus 4.8 router (Figure 4), SPILLWAY holds in-scope attack success at zero on all seven suites, while ROPE leaks on three (DailyLife .010, Shopping .044, Slack .048). SPILLWAY's utility under attack is higher on five suites by .07 to .17 (Workspace .838 vs. .670), by .022 on Shopping, and by .010 on Slack. With the agent's own model as its router (gpt-4o-mini), ROPE's mean utility falls from .468 to .361 and it still leaks on Shopping (.072) and Slack (.019). Each suite runs once (seed 42). In a three-seed run of an earlier build, SPILLWAY had no raw hit and its utility varied by at most .011 on Slack and Shopping, where the margins are smallest. On benign qwen3:8b traffic SPILLWAY is also the lighter: it adds 0.19 and 0.39 model calls per Workspace and Slack episode, judge included, no more than ROPE's 0.62 and 0.39 and far below the model-call defenses (Figure 3).

What each component adds. Table 3 separates the components on one backbone. The floor lets no attacker target through on any suite, as P1 predicts. On Slack the agent still proposes the attacker's membership calls, and AgentDojo's trace-based check counts them in eight pairs although the floor refused every one. Adding the judge without the sanitizer recovers Banking utility ($+.042$) but opens the only executed leaks, because it admits the attacker's account on "pay the bill" and an attacker address as a meeting participant. Adding the sanitizer instead lifts utility on Slack ($+.105$) and Travel ($+.100$), since an agent that never reads the imperative stops proposing the attacker's calls, and costs .033 on Workspace. With both, SPILLWAY holds zero on all four suites at mean utility .370, against .300 for the floor. Removing only the span veto changes no attack outcome and moves utility by two Workspace pairs (last row), since with the sanitizer on the agent never proposed a removed target.

Table 3: Component ablation on qwen3:8b (**AgentDojo** v1.2.2, 949 pairs per arm): in-scope under-attack utility (UAU ↑) / attack success (ASR ↓, executed effect). Delegation rules are on in every arm, and object cases (Sec. 3.1) reach their judge in every row. The judge row recovers identity-target refusals. SPILLWAY in bold. The last row removes only the span veto.

Arm	Banking	Slack	Travel	Workspace
Floor	.333 / .000	.181 / .000	.243 / .000	.445 / .000
Floor + judge	.375 / .028	.190 / .000	.243 / .000	.439 / .002
Floor + sanitizer	.375 / .000	.286 / .000	.343 / .000	.412 / .000
SPILLWAY (all)	**.438 / .000**	**.286 / .000**	**.343 / .000**	**.414 / .000**
SPILLWAY, no span veto	.438 / .000	.286 / .000	.343 / .000	.418 / .000

Table 4: **Stress tests** (suite v1.2.2): in-scope under-attack utility (UAU ↑) / attack success (ASR ↓). Left: **ChatInject** (Chang et al., 2026), qwen3:8b, 560 Workspace and 105 Slack pairs. Right: **AutoDojo** (Ma et al., 2026a) optimized against Progent, gpt-4o-mini, 144 Banking, 105 Slack, 140 Travel pairs. Trace, executed, and in-scope levels in Table 6. NRP ↑/ERS ↑ (Sec. 4) use each panel's benign utility (Table 1). ROPE's ChatInject row uses a benign run in that row's setting (Appendix A). SPILLWAY in bold. Its AutoDojo row uses rules written on four failing pairs (held-out sets in Appendix A), and its ChatInject row predates them. ‡Released code adapted to our agent and suite (App. B). †authors' planner and validator.

	ChatInject (qwen3:8b)						**AutoDojo** (gpt-4o-mini)							
	Workspace		Slack		Composite		Banking		Slack		Travel		Composite	
Method	UAU	ASR	UAU	ASR	NRP	ERS	UAU	ASR	UAU	ASR	UAU	ASR	NRP	ERS
Undefended	.052	.245	.095	.648	.269	.496	.444	.354	.524	.571	.329	.329	.342	.574
SPILLWAY	**.371**	**.000**	**.286**	**.000**	**.463**	**.633**	**.486**	**.000**	**.533**	**.000**	**.550**	**.000**	**.609**	**.754**
MELON	.084	.020	.095	.000	.494	.660	.340	.021	.286	.000	.293	.000	.564	.716
Progent‡	.161	.005	.057	.010	.305	.457	.306	.104	.467	.124	.414	.121	.468	.654
DRIFT†	.393	.062	.352	.019	.445	.625	.438	.090	.543	.010	.486	.086	.490	.659
ROPE‡	.157	.002	.162	.095	.292	.462	.424	.000	.352	.019	.300	.000	.434	.604
Spotlighting	.191	.013	.076	.181	.398	.591	.431	.306	.543	.514	.343	.279	.360	.594
PI detector	.050	.021	.162	.000	.230	.374	.403	.049	.219	.067	.214	.029	.330	.498

Holding the floor under the strongest recent attacks. Beyond the standard template, we stress SPILLWAY with three of the strongest recent injection attacks: a forged chat template (ChatInject, ICLR 2026, Chang et al., 2026), optimizer-generated injections (AutoDojo, Ma et al., 2026a), and a released RL attacker (PISmith, Yin et al., 2026). Each breaks the undefended agent and most baselines. SPILLWAY holds them to its floor, a raw zero under ChatInject and on AutoDojo's 389 pairs and a mean in-scope ASR of .007 under PISmith, at the most utility among the defenses that reach it. ChatInject embeds a fabricated <|im_start|> turn inside untrusted output. With the forged turns scrubbed, SPILLWAY has no raw hit in any of the 665 pairs and recovers part of the utility the hijack destroys (Slack .095 → .286, Workspace .052 → .371). ROPE lets the hijacked agent propose the attacker's calls and blocks most of them at execution, so its utility stays low (.157, .162) and it leaks .002 on Workspace and .095 on Slack, and DRIFT keeps more utility (.393 and .352) but leaks .062 and .019 (Table 4). In the attacker-as-sender variant (Appendix C), no attack succeeds under SPILLWAY.

AutoDojo's injections were optimized against another defended agent: Progent, the defense they were optimized against, executes them in 10 to 12% of pairs (Table 4). The build before the address and payee-change rules let 4 of the 389 pairs through at in-scope level, each on the path P3 describes: no rule removed the text, and the judge took the attacker's target as delegated. We used those four pairs as development examples for the rules of Sec. 3.2, so the result on this set is descriptive, not held out: the rules leave no injected effect in the 389 pairs, neither a call to an attacker's target nor injected content in a final answer. On the two held-out Banking sets (144 pairs each, Appendix A), attack success is .007 by AgentDojo's check on the set optimized against DRIFT and .000 on the set optimized against the undefended agent, in which 18 of the 36 injection texts are optimized and the rest are AgentDojo's template. A second payee transfer the check misses makes the DRIFT set .014 and the mean of the two sets .007. The rules change no earlier result (Appendix B).

A frontier RL attacker. We replay PISmith (Yin et al., 2026), a released RL-trained attacker, open-loop against every defense: the first of its ten pair-conditioned variants per injection (ASR@1, gpt-4o-mini, Table 7). It breaks the undefended agent (.568 in-scope ASR) and five of the six

baselines (.025 to .489), yet SPILLWAY holds it to the floor (.007) with more utility than every other defense except DRIFT, which leaks .064 (Figure 1d). Only ROPE also reaches the floor, at far lower utility (.173 against SPILLWAY's .265). On qwen3:8b both hold the floor as well.

6 Related Work

Information-flow and capability systems (Costa et al., 2025; Zhong et al., 2025; An et al., 2025; Wu et al., 2024; Li et al., 2026a) also constrain untrusted data, and Task Shield (Jia et al., 2025) asks whether a call serves the user goal.

The closest sibling is ROPE (Ma et al., 2026b), whose deterministic origin check guards an audited set of sensitive parameters after a router reads each request. SPILLWAY takes a lighter point: a flat typed-identity floor fixed by the tool schemas and a role lexicon, with no per-task router and lighter than the per-step dependency state of APPA (Kravchenko et al., 2026) or AgentArmor (Wang et al., 2025) and the per-argument provenance of PACT (Fan et al., 2026), and a judge bounded by ingress evidence. CommandSans (Das et al., 2025) learns a token-level sanitizer for tool outputs. Tool-input and tool-output firewalls (Bhagwatkar et al., 2025) run an LLM on every tool input and output and filter content, with no check on where an effect goes. They report zero on the standard attack, but their own adaptive cascade raises their sanitizer's attack success to 30% (Slack). Agent data injection (Choi et al., 2026) spoofs trusted metadata inside untrusted text, such as a forged author field in a comment body, and succeeds against most defenses it tests except CaMeL's strict policy. A spoofed value in a typed field is the case of Sec. 7. SPILLWAY's floor and sanitizer use no model, and its only model call is a bounded judge, under one call per benign episode on qwen3:8b (Figure 3). With its deterministic floor, SPILLWAY holds in-scope attack success at .000 under a forged chat template and at .007 on average under a frontier reinforcement-learning attacker (PISmith), where content filtering alone does not (Tables 4, 7).

7 Limitations

SPILLWAY's guarantee is scoped to identity targets (Sec. 2): an injection that steers an effect to a legitimate identity, such as an existing contact or a payee in a bill the user asked to pay, is left to the sanitizer and the judge, and a target that never appears verbatim in tool output falls outside P1. An attacker who writes a typed field, for example by sending the email, puts its own address in I. Target roles come from a lexicon written once per benchmark, so a new tool set needs its roles declared. Because the floor is a set lookup rather than a classifier, rewording the injection around a target does not change whether the floor admits it. The judge, the delegation rules, and the lexicon are the surface an adaptive attacker (Zhan et al., 2025; Hofer et al., 2026) would target, which the four AutoDojo pairs on the P3 path already exercise.

8 Conclusion

We presented SPILLWAY, a prompt-injection defense for tool-using agents that separates target admission from sanitization. SPILLWAY reads where an action may go from the tools' typed returns and the user's task. The sanitizer removes instruction-like regions before the model reads them, and a local judge restores only refused calls that lie outside the removed text. Targets are decided by structure, and the model is used only where structure cannot decide. Across three backbones and four AgentDojo suites, no standard templated injection reached an attacker's target, at the highest mean utility and composite scores under that attack of the defenses we run. Under a forged chat template, in-scope attack success was a raw zero. Under a released reinforcement-learning attacker, it averaged .007. Personal agents now act on their own across a user's mail, calendar, and payments, and prompt injection can turn that autonomy against the user. SPILLWAY sits in front of such an agent as an efficient, auditable egress layer. Because the floor is a set lookup over the tool schemas, not a model call, it runs inline with almost no latency and carries to a new tool set once its roles are declared, so we expect this light layer to slot into many services and agent apps.

DISCLOSURE OF GENERATIVE AI USE

The authors conceived the research question, the defense, the threat model, and the experimental design, directed the implementation and every experiment, and verified the results. Under their direction, we used generative AI tools as assistants in four ways. They drafted sections and polished prose, which the authors revised. They surfaced candidate related work, which the authors read before citing. They wrote code to the authors' specifications and ran the experiment harness. And they served as a sounding board for design alternatives that the authors proposed and chose among. Every number, table cell, and figure value was recomputed from the run artifacts and checked by the authors, and no generative AI produced datasets, benchmark content, proofs, or results. The authors take full responsibility for everything the paper states.

REFERENCES

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. IPIGuard: A Novel Tool Dependency Graph-Based Defense Against Indirect Prompt Injection in LLM Agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1023–1039, 2025. doi: 10.18653/v1/2025.emnlp-main.53.

Rishika Bhagwatkar, Kevin Kasa, Abhay Puri, Gabriel Huang, Irina Rish, Graham W. Taylor, Krishnamurthy Dj Dvijotham, and Alexandre Lacoste. Indirect Prompt Injections: Are Firewalls All You Need, or Stronger Benchmarks?, 2025. arXiv:2510.05244.

Hwan Chang, Yonghyun Jun, and Hwanhee Lee. ChatInject: Abusing Chat Templates for Prompt Injection in LLM Agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.22830.

Mohammad Hossein Chinaei. Causality Laundering: Denial-Feedback Leakage in Tool-Calling LLM Agents, 2026. arXiv:2604.04035.

Woohyuk Choi, Juhee Kim, Taehyun Kang, Jihyeon Jeong, Luyi Xing, and Byoungyoung Lee. Agent Data Injection Attacks are Realistic Threats to AI Agents, 2026. arXiv:2607.05120.

Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing AI Agents with Information-Flow Control, 2025. arXiv:2505.23643.

Debeshee Das, Luca Beurer-Kellner, Marc Fischer, and Maximilian Baader. CommandSans: Securing AI Agents with Surgical Precision Prompt Sanitization, 2025. arXiv:2510.08829.

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 (NeurIPS 2024) Datasets and Benchmarks Track*, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating Prompt Injections by Design. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026. arXiv:2503.18813.

Linfeng Fan, Ziwei Li, Yuan Tian, Yichen Wang, Rongsheng Li, and Xiong Wang. The Granularity Mismatch in Agent Security: Argument-Level Provenance Solves Enforcement and Isolates the LLM Reasoning Bottleneck, 2026. arXiv:2605.11039.

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 (AISec '23)*, pp. 79–90, 2023. doi: 10.1145/3605764.3623985.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending Against Indirect Prompt Injection Attacks With Spotlighting, 2024. arXiv:2403.14720.

David Hofer, Edoardo Debenedetti, and Florian Tramèr. Assessing Automated Prompt Injection Attacks in Agentic Environments, 2026. arXiv:2606.10525.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The Task Shield: Enforcing Task Alignment to Defend Against Indirect Prompt Injection in LLM Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29680–29697, 2025. doi: 10.18653/v1/2025.acl-long.1435.

Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. TAMAS: Benchmarking Adversarial Risks in Multi-Agent LLM Systems, 2025. arXiv:2511.05269.

Arseny Kravchenko, Vadim Liventsev, Innokentii Konstantinov, Ildar Iskhakov, and Matvey Kukuy. APPA: Recoverable Information-Flow Control for Real-World LLM Agents, 2026. arXiv:2607.24625.

Evan Li, Tushin Mallick, Evan Rose, William Robertson, Alina Oprea, and Cristina Nita-Rotaru. ACE: A Security Architecture for LLM-Integrated App Systems. In *Network and Distributed System Security Symposium (NDSS)*, 2026a. arXiv:2504.20984.

Hao Li, Xiaogeng Liu, Hung-Chun Chiu, Dianqi Li, Ning Zhang, and Chaowei Xiao. DRIFT: Dynamic Rule-Based Defense with Injection Isolation for Securing LLM Agents. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. AgentDyn: Are Your Agent Security Defenses Deployable in Real-World Dynamic Environments?, 2026b. arXiv:2602.03117.

Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. VIGIL: Defending LLM Agents Against Tool-Stream Injection via Verify-Before-Commit. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9764–9785, 2026. doi: 10.18653/v1/2026.acl-long.443.

Xinhang Ma, Taoran Li, Chaowei Xiao, Zhiyuan Yu, Ning Zhang, and Yevgeniy Vorobeychik. AutoDojo: Adaptive Black-Box Attacks Reveal the Limits of IPI Defenses and Task-Specification Effects in LLM Agents, 2026a. arXiv:2606.15057.

Xinhang Ma, Chaowei Xiao, William Yeoh, Ning Zhang, and Yevgeniy Vorobeychik. ROPE: Routed Origin Policy Enforcement against Indirect Prompt Injection. *arXiv preprint arXiv:2608.27496*, 2026b.

Model Context Protocol. Model Context Protocol Specification. https://modelcontextprotocol.io/specification, 2026. Version 2026-07-28.

OpenAI. GPT-4o mini: Advancing Cost-Efficient Intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024. Model snapshot gpt-4o-mini-2024-07-18.

Protect AI. deberta-v3-base-prompt-injection-v2. Hugging Face model card, https://huggingface.co/protectai/deberta-v3-base-prompt-injection-v2, 2024.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI Agents with Privilege Control, 2025. arXiv:2504.11703.

Peiran Wang, Yang Liu, Yunfei Lu, Yifeng Cai, Hongbo Chen, Qingyou Yang, Jie Zhang, Jue Hong, and Ye Wu. AgentArmor: Enforcing Program Analysis on Agent Runtime Trace to Defend Against Prompt Injection. *arXiv preprint arXiv:2508.01249*, 2025.

Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao. System-Level Defense against Indirect Prompt Injection Attacks: An Information Flow Control Perspective. *arXiv preprint arXiv:2409.19091*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenlong Yin, Runpeng Geng, Yanting Wang, and Jinyuan Jia. PISmith: Reinforcement Learning-based Red Teaming for Prompt Injection Defenses, 2026. arXiv:2603.13026.

Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. Adaptive Attacks Break Defenses Against Indirect Prompt Injection Attacks on LLM Agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 7116–7132, 2025. doi: 10.18653/v1/2025.findings-naacl.395.

Dongsen Zhang, Zekun Li, Xu Luo, Xuannan Liu, Peipei Li, and Wenjun Xu. MCP Security Bench (MSB): Benchmarking Attacks Against Model Context Protocol in LLM Agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.15994.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.02644.

Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L. Titzer, Heather Miller, and Phillip B. Gibbons. RTBAS: Defending LLM Agents Against Prompt Injection and Privacy Leakage, 2025. arXiv:2502.08966.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. MELON: Provable Defense Against Indirect Prompt Injection Attacks in AI Agents. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80310–80329, 2025.

A ATTACK SUCCESS AT EVERY SCORING LEVEL

Table 5 gives, for every breadth cell, the raw and in-scope attack success of Sec. 4. Raw applies the executed-effect rule. Collision filtering removes no hit in any cell, so the collision-filtered level equals raw. A single value means raw and in-scope agree. SPILLWAY is .000 at both levels in all twelve cells. Table 6 gives the **AutoDojo** panel of Table 4 at the trace, executed, and in-scope levels. Before the address and payee-change rules, SPILLWAY's in-scope ASR on this panel was .007 (Travel), .014 (Banking), and .010 (Slack). After them, on two held-out Banking sets from the same release (144 pairs each, not used when the rules were written), it is .007 by AgentDojo's check (.014 counting a second transfer to an attacker-written payee) on the set optimized against DRIFT and .000 on the set optimized against the undefended agent, in which 18 of the 36 injection texts are optimized and the rest are the template. Utility is .444 (64/144) on each. ROPE's AutoDojo row has the setting of its gpt-4o-mini cells in Table 1 and uses their BU. Its ChatInject row runs without ROPE's clamp, which drops a router override looser than the audited defaults and is on in Table 1. Its ChatInject composites therefore use a benign run in that setting, Workspace .325 and Slack .286.

Table 5: **AgentDojo** breadth grid (suite v1.2.2, 949 attacked pairs per backbone: 144 Banking, 105 Slack, 140 Travel, 560 Workspace): raw / in-scope ASR per backbone, suite, and method. One value when the two agree.

Cell	Undefended	SPILLWAY	MELON	Progent	DRIFT	ROPE	Spotlighting	PI detector
Banking								
gpt-4o-mini	.556	.000	.049	.097	.111	.000	.549	.118
qwen3-coder:30b	.299	.000	.125	.056	.014	.000	.382	.097
qwen3:8b	.181	.000	.215	.000	.028	.000	.188	.083
Slack								
gpt-4o-mini	.629	.000	.000	.076	.000	.019	.562	.086
qwen3-coder:30b	.724	.000	.076	.010	.000	.019	.581	.124
qwen3:8b	.448	.000	.457	.019	.010	.019	.467	.086
Travel								
gpt-4o-mini	.300/.229	.000	.086/.000	.093/.029	.000	.079/.000	.236/.179	.014
qwen3-coder:30b	.186/.129	.000	.100/.029	.064/.007	.000	.043/.000	.193/.136	.000
qwen3:8b	.264/.229	.000	.186/.157	.064/.036	.000	.043/.000	.200/.150	.000
Workspace								
gpt-4o-mini	.184	.000	.002	.041	.009	.000	.148	.070
qwen3-coder:30b	.036	.000	.004	.002	.002	.000	.039	.025
qwen3:8b	.020	.000	.011	.004	.014	.000	.020	.014

Table 6: **AutoDojo** on gpt-4o-mini (144 Banking, 105 Slack, 140 Travel pairs): trace-level, executed (exec.), and in-scope ASR. DRIFT's three levels coincide. ROPE's scorer reports executed and in-scope ASR, not a separate trace (−).

	Banking			Slack			Travel		
Method	trace	exec.	in-scope	trace	exec.	in-scope	trace	exec.	in-scope
Undefended	.354	.354	.354	.571	.571	.571	.364	.364	.329
SPILLWAY	.000	.000	.000	.000	.000	.000	.000	.000	.000
MELON	.021	.021	.021	.000	.000	.000	.043	.043	.000
Progent	.104	.104	.104	.210	.124	.124	.150	.150	.121
DRIFT	.090	.090	.090	.010	.010	.010	.086	.086	.086
ROPE	–	.000	.000	–	.076	.019	–	.014	.000
Spotlighting	.306	.306	.306	.514	.514	.514	.307	.307	.279
PI detector	.049	.049	.049	.067	.067	.067	.029	.029	.029

Table 8 gives the delegated-routing grid of Figure 4 per suite, with ROPE under its authors' router and under the agent's own model. Per-cell benign utility (BU) is in Table 1.

B REPRODUCIBILITY

Agent. Every method runs on the same agent request: each benchmark's own default system prompt in the `system` role, temperature 0, seed 42, at most 8192 output tokens, and the backbone's reasoning switched off where it has a switch. AgentDojo's prompt has 469 bytes, and AgentDyn's (543 bytes) adds one line that tells the agent to complete all tasks without requesting user confirmation. Spotlighting appends its delimiter instruction. DRIFT keeps its authors' prompt, which embeds the default's instructions in an environment block and adds a line that tells its agent to complete all tasks without requesting confirmation. gpt-4o-mini-2024-07-18 is served by the OpenAI API, while

Table 7: **Frontier RL attacker.** PISmith (Yin et al., 2026), released by its authors, replayed open-loop with the first of its ten pair-conditioned variants per injection (ASR@1, gpt-4o-mini, v1.2.2, full grid): in-scope UAU ↑/ASR ↓, macro over the four suites. SPILLWAY (bold) holds the attack to the floor at the most utility there. Only ROPE also reaches it. ‡Released code adapted to our agent and suite (App. B). †authors' planner and validator.

Method	UAU ↑	in-scope ASR ↓
Undefended	.128	.568
SPILLWAY	**.265**	**.007**
MELON	.114	.061
Progent‡	.166	.096
DRIFT†	.428	.064
ROPE‡	.173	.007
Spotlighting	.151	.489
PI detector	.155	.025

Table 8: Delegated-routing grid (**AgentDojo** v1.2.2 above the rule, **AgentDyn** below), gpt-4o-mini: in-scope under-attack utility (UAU ↑) / attack success (ASR ↓), one run per suite (144, 105, 140, 560, 180, 200, 180 pairs from top). The mean is unweighted over suites. The last two rows give the composites NRP ↑/ERS ↑ (Sec. 4), macro-averaged over the seven suites. The AgentDyn suites have no benign run, so NRP and ERS here use UAU for BU, as MSB (Zhang et al., 2026) does for NRP (NRP $=$ UAU $\times(1-$ASR$)$).

Suite	SPILLWAY	ROPE (Opus 4.8 router)	ROPE (gpt-4o-mini router)
Banking	.521 / .000	.438 / .000	.431 / .000
Slack	.524 / .000	.514 / .048	.343 / .019
Travel	.571 / .000	.500 / .000	.350 / .000
Workspace	.838 / .000	.670 / .000	.539 / .000
GitHub	.600 / .000	.500 / .000	.383 / .000
DailyLife	.435 / .000	.295 / .010	.150 / .000
Shopping	.383 / .000	.361 / .044	.333 / .072
Mean	.553 / .000	.468 / .015	.361 / .013
NRP (UAU)	**.553**	.462	.357
ERS (UAU)	**.703**	.627	.519

qwen3-coder:30b and qwen3:8b run on Ollama. Every stack uses suite v1.2.2, whose task and data files equal AgentDojo 0.1.35's. The harness around them is each method's own, and the variant of Appendix C re-addresses five emails. Each cell runs once. To bound run-to-run variation, we reran SPILLWAY on all three backbones and four suites across three agent-decoding seeds ({42, 777, 123}) with three repeats each, nine runs per backbone and 27 in all. In-scope attack success was .000 in every run, and per-suite utility had a standard deviation of at most .041 across the three repeats of a fixed seed and .021 across the three per-seed means.

SPILLWAY. The floor and the sanitizer are rules with no model call. The judge is gemma4:31b (Q4_K_M) on a local Ollama server with a 4096-token context at temperature 0. The address and payee-change rules change no sanitizer output on any of the 27,392 tool messages behind Tables 1 and 3, the ChatInject panel, and Figures 3 and 4, and match no text in AgentDojo's environment data. Rerunning the 949 qwen3:8b pairs of Table 1 with the address rules reproduces each suite's utility and attack success to the third decimal. The role lexicon has 11, 22, 10, and 11 verb prefixes for read-only, writing, destructive, and granting tools, and 19 target-argument names for AgentDojo with 6 more for AgentDyn.

Comparators. MELON, Progent, DRIFT and ROPE run from their released code. MELON embeds with nomic-embed-text, the prompt-injection detector is protectai/deberta-v3-base-prompt-injection-v2, and Progent's policy is written by the cell's own backbone, with Progent's own seed and retry ladder for its policy calls. ROPE's router is the agent in Tables 1, 2 and 5 and on the ChatInject, AutoDojo and PISmith panels, and its authors' Claude Opus 4.8 with cached scopes in Figure 4 and Figure 1a (with gpt-4o-mini as router in Table 8).

Attacks and scoring. The standard attack is AgentDojo's `important_instructions`. AutoDojo replays variant 0 of its released gpt-4o-mini injections (commit 89413642). PISmith replays variant 0 of the ten its released attacker wrote for each pair, the same texts for every defense.

The AgentDyn suites have 180 (GitHub), 200 (DailyLife), and 180 (Shopping) attacked pairs. Local runs use one NVIDIA RTX PRO 6000 Blackwell GPU.

C ATTACKER-AS-SENDER VARIANT

The variant breaks P1's premise: the five injection-carrying Workspace emails come from the attacker's own address, so the address enters I through the typed sender field. It covers the nine user tasks that read these emails and all 14 injection tasks (126 pairs, qwen3:8b). SPILLWAY, the floor-only arm, Progent, and the prompt-injection detector let no attack through. The undefended agent, MELON, Spotlighting, and DRIFT each execute the attack in one of the 126 pairs. In that pair the floor-only agent tried to email the attacker's address and was refused, because the injected text repeats the address and the floor refuses a target repeated in free text. With the sanitizer on, the agent never proposed the call. An instruction that names the attacker only by role is not tested here. Utility under attack is .548 for SPILLWAY, .611 for the floor-only arm, and .603 undefended.