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

Anonymous authors
Paper under double-blind review

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 check with no per-task plan. The check admits a call when each target was named by the user, directly or through a fixed delegation rule, or returned by a read-only tool in a typed field; 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, SPILLWAY is the only defense we evaluate under which no templated injection redirects an action to an attacker's target, it also holds under chat-template forging, and it requires no defense-triggered backbone re-run while keeping the highest observed mean utility under the basic attack. As personal AI agents take over users' email, files, and payments, this separation gives them a deterministic, auditable guard on where their actions go, one that keeps them useful and does not change with the backbone.

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 (Anthropic, 2025). 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. A model writes the privilege policy a checker enforces (Progent (Shi et al., 2025)), re-runs a masked copy of the trajectory to detect a hijack (MELON (Zhu et al., 2025)), or validates each call against a plan (DRIFT (Li et al., 2025b)); others gate the final commit (Lin et al., 2026; Li et al., 2025a). 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 3.3 to 9.9 backbone calls per benign episode (Sec. 5). 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 tools and sources 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

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

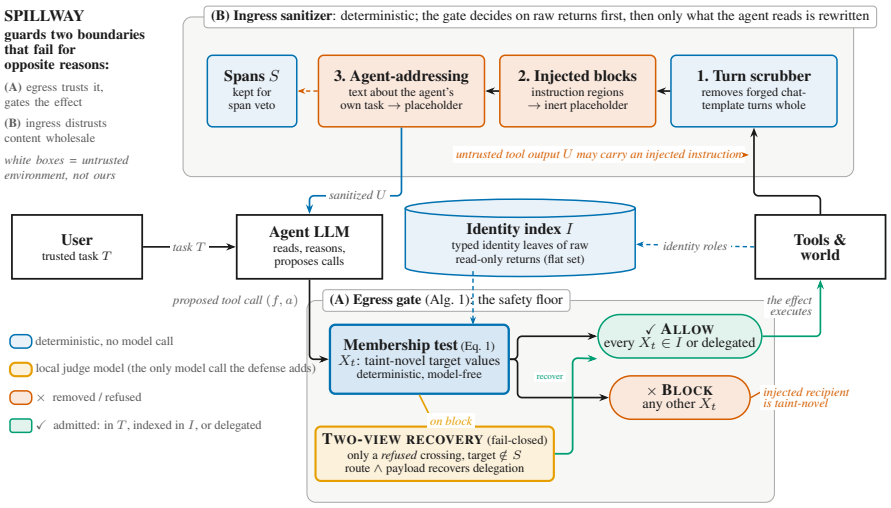


Figure 1: SPILLWAY's two boundaries. **(A)** The egress floor admits a call when its targets are named in T, typed in the index I, or covered by a delegation rule (Alg. 1); a refused call may get one fail-closed two-view judge. **(B)** The deterministic sanitizer (three of its rule groups shown) removes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

its context, and the question is which parts are instructions, a judgment about language; at **egress**, a call produces an effect, and the question is 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 in our threat model an injection placed in tool output writes only free text such as a body or a description. 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 1), which makes the two decisions separately and joins them through evidence. Its egress floor admits a side-effecting call when every target value that carries tool-output text was named by the user or 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 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:* no raw attack success on any of the twelve backbone-by-suite cells, where no other defense is zero on more than five, at the highest twelve-cell mean utility under the basic attack (.484) and with no defense-triggered backbone re-run; zero also under chat-template forging on qwen3:8b (Tables 1 and 4).
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, attack success is zero on every suite and mean utility rises from the floor's .300 to .370 (Table 3).
3. **A head-to-head comparison with routed-origin enforcement.** *Result:* zero in-scope attack success on seven delegated-routing suites and both ChatInject suites, where the routed-origin defense leaks on three and on two; utility is higher on all seven suites, by .07 to .17 on five and by .010 and .022 on Slack and Shopping, and with the agent's own model as router

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 that only a refused target outside the removed spans reaches. 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 routed-origin defense loses utility (mean $.468 \rightarrow .361$) and still leaks on two (Figure 3, Table 4).

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 free-text fields that AgentDojo's injection mechanism fills in content the agent later reads, such as an email body or a web page. 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; the sanitizer reduces them but does not bound them.

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

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 (11, 22, 10, and 11 prefixes), argument names mark targets (19 for AgentDojo, 6 more for AgentDyn) 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 that appears 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 follow a stricter rule. A principal that a call gives access, such as a new workspace or channel member, may not come from tool output unless the user named that person, and no judge reopens this refusal. The participants of a new calendar event and the recipient of a shared file are targets, even when the write lands in the user's own calendar or drive, and a deletion is refused unless T asks for one.

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 ("invite Alice" admits `alice.miller@...`), a contact attribute of an entity the user named, a resource owned by the user's own account, or a URL whose domain the user named; on repository clone, invite, and transfer tools they also admit a target that lies in no removed span, and 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 injected instruction keeps working toward the attacker's goal and neglects 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 with an inert placeholder three kinds of text: regions delimited by the benchmark's injection template (`<INFORMATION>` and its "important message" form); 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; and 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 is anchored to the first user message, so a forged user turn never becomes trusted. The instruction rules keep the text they remove as the span set S: a refused target inside S never reaches the judge (Sec. 3.3). The index I is built 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

Table 1: Basic attack on **AgentDojo** (Debenedetti et al., 2024) (three backbones × four suites, suite v1.2.2, full grid): utility under attack ↑ / in-scope ASR ↓; SPILLWAY in bold. Table 2 gives the means.

		Undefended		**SPILLWAY**		MELON		Progent		DRIFT		Spotlighting		PI detector	
Suite	Backbone	Util	ASR	Util	ASR	Util	ASR	Util	ASR	Util	ASR	Util	ASR	Util	ASR
Banking															
	gpt-4o-mini	.424	.556	**.521**	**.000**	.306	.049	.299	.097	.431	.111	.431	.549	.375	.118
	qwen3-coder:30b	.486	.299	**.500**	**.000**	.299	.125	.354	.056	.542	.014	.410	.382	.424	.097
	qwen3:8b	.403	.181	**.438**	**.000**	.403	.215	.361	.000	.410	.028	.431	.188	.340	.083
Slack															
	gpt-4o-mini	.514	.629	**.524**	**.000**	.133	.000	.400	.076	.486	.000	.524	.562	.162	.086
	qwen3-coder:30b	.467	.724	**.429**	**.000**	.190	.076	.362	.010	.276	.000	.390	.581	.171	.124
	qwen3:8b	.419	.448	**.286**	**.000**	.362	.457	.105	.019	.314	.010	.295	.467	.162	.086
Travel															
	gpt-4o-mini	.336	.229	**.571**	**.000**	.300	.000	.529	.029	.607	.000	.429	.179	.107	.014
	qwen3-coder:30b	.414	.129	**.407**	**.000**	.393	.029	.414	.007	.443	.000	.414	.136	.200	.000
	qwen3:8b	.229	.229	**.343**	**.000**	.186	.157	.164	.036	.264	.000	.171	.150	.050	.000
Workspace															
	gpt-4o-mini	.398	.184	**.838**	**.000**	.330	.002	.621	.041	.607	.009	.430	.148	.254	.070
	qwen3-coder:30b	.480	.036	**.543**	**.000**	.430	.004	.482	.002	.530	.002	.452	.039	.275	.025
	qwen3:8b	.434	.020	**.414**	**.000**	.441	.011	.366	.004	.523	.014	.457	.020	.243	.014

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 admits, or a commit on an object the task did not name). Grant, deletion, 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 is a refusal. 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.

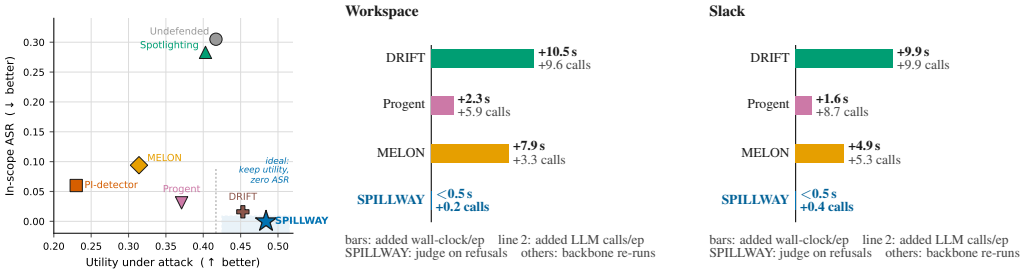
P1 (floor soundness). Under the tokens of Eq. 1, if the attacker writes only free-text fields (Sec. 2), the lookup refuses every call whose target value appears in attacker-written text and not in T, because such a value is either outside I or repeated in free text. *Reason:* I holds only typed identity 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 (what a leak requires). An injected target leaks only if it lies outside the removed spans *and* the judge or a delegation rule admits the refused call. *Reason:* by P1 the lookup refuses an attacker's target, and by the span veto a removed one never reaches J. P3 is a structural conjunction, not an assumption that the two failures are independent; one adaptive input may target both.

4 Experimental Setup

We evaluate SPILLWAY in two settings, both on AgentDojo (Debenedetti et al., 2024), whose checks serve as the utility and attack-success oracle, with hits rescored by executed effect as defined below. The **breadth** setting tests SPILLWAY on three backbones and four stateful suites under the basic



(a) Utility vs. ASR. (b) Workspace: added time, calls. (c) Slack: added time, calls.

Figure 2: Safety–utility position (a, 12-cell means) and added runtime per suite (b,c, qwen3:8b benign episodes on an RTX PRO 6000; SPILLWAY timed in a separate session; calls include its judge).

Table 2: **AgentDojo** breadth grid summary (Table 1): twelve-cell means, and the number of cells with a raw hit, brought to zero by the content-manipulation filter, and still leaking at in-scope. The collision filter zeroes no cell.

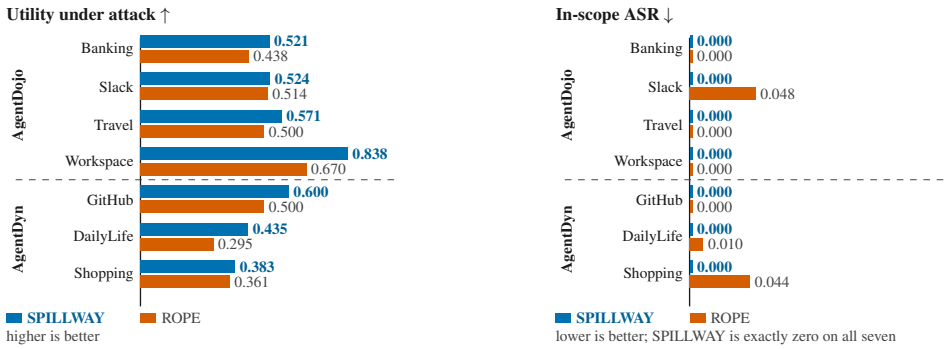
	12-cell mean			Cells of 12		
Method	Benign util	Util under attack	In-scope ASR	raw hit	zeroed by content filter	in-scope leak
Undefended	.540	.417	.305	12	0	12
SPILLWAY	**.520**	**.484**	**.000**	**0**	**0**	**0**
MELON	.526	.314	.094	11	1	10
Progent	.437	.371	.031	11	0	11
DRIFT	.460	.453	.016	7	0	7
Spotlighting	.528	.403	.283	12	0	12
PI detector	.332	.230	.060	10	0	10

attack, plus three single-backbone stress tests. The **delegated-routing** setting compares the full SPILLWAY with ROPE (Ma et al., 2026b), the closest defense: the three AgentDyn (Li et al., 2026) suites, many of whose tasks send effects to recipients read from tool output, and the four AgentDojo suites, on gpt-4o-mini. Every AgentDojo grid uses **suite v1.2.2** for every method, on the full grid of user and injection tasks. SPILLWAY runs all its components in every setting, and Table 3 separates their contributions.

Scoring: three attack-success levels. **Raw** is AgentDojo's success check with the executed-effect rule below. **In-scope** drops hits whose attacker target the trusted task T already named (the collision filter, which removes no hit in any cell) and content-manipulation tasks, which make the agent state attacker text but name no recipient or sink for an effect gate to intercept (Appendix A). Neither filter reads defense state or touches SPILLWAY, which has no raw hit on any of the twelve cells while MELON and Progent each have raw hits on eleven. We report at in-scope, the level at which the guarantee is defined.

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); on qwen3:8b benign traffic it is consulted 0.05 (Workspace) and 0.43 (Slack) times per episode.

Grid, attacks, and comparators. The breadth backbones are gpt-4o-mini (OpenAI, 2024), qwen3-coder:30b, and qwen3:8b (Yang et al., 2025). The basic attack is AgentDojo's `important_instructions` template. Three stress tests follow: chat-template forging (Chat-Inject (Chang et al., 2026); qwen3:8b, 665 pairs); an attacker-as-sender variant of Workspace in which 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); and optimizer-generated injections, the AutoDojo release (Ma et al., 2026a) optimized for gpt-4o-mini against Progent, replayed on the three suites it covers (Banking, Slack, Travel; 389 pairs), plus two held-out Banking sets (Sec. 5). We run head-to-head against MELON (Zhu et al., 2025), Progent (Shi et al., 2025), and DRIFT (Li et al.,



(a) Utility under attack, per suite.

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

Figure 3: **Delegated-routing grid. AgentDyn** (Li et al., 2026) and **AgentDojo** suite v1.2.2, gpt-4o-mini, on the full grid of each suite. ROPE (Ma et al., 2026b) uses its authors' Claude Opus 4.8 router and cached scopes, with the Workspace scopes produced once by the same router.

2025b) on the breadth grid and against ROPE (Ma et al., 2026b) 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 on the ChatInject panel so is ROPE's router. 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 in the supplement.

5 Results

SPILLWAY holds attack success at zero on all twelve breadth cells, while MELON, Progent, and DRIFT still leak on ten, eleven, and seven (Table 2); on qwen3:8b the floor alone, with no judge, also holds zero (Table 3). Its twelve-cell mean utility under attack is the highest of any method (.484; next DRIFT, .453; one run per cell), and on benign traffic it keeps 96% of the undefended agent's utility. A strict check also refuses some legitimate calls: SPILLWAY is below the undefended agent in four of the twelve cells, most on Slack with qwen3:8b, where it keeps 68% of the undefended agent's utility under attack.

What the breadth grid shows. 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), where injections derail the undefended agent's tasks and the sanitizer removes them before the agent reads them (Table 3). With zero observed leaks, the one-sided 95% Clopper–Pearson bound is 2.8% at the smallest cell (105 pairs) and 0.1% pooled over the 2,847 attacked pairs, for AgentDojo's templated injections and treating pairs as independent.

A comparison with routed-origin enforcement. Against ROPE with its authors' Claude Opus 4.8 router (Figure 3), 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) and within one seed's spread on Slack (+.010) and Shopping (+.022). 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, with seed 42; in a three-seed run of an earlier SPILLWAY build (seeds 42, 123, 777), SPILLWAY had no raw hit under any seed and its seed-42 utility was within .031 of the three-seed mean on every suite.

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

Table 3: Component ablation on qwen3:8b (**AgentDojo** suite v1.2.2, 949 pairs per arm): utility under attack / in-scope ASR (executed effect). Each row adds the named components to the floor; the judge row adds recovery of identity-target refusals (object cases of Sec. 3.1 reach their judge in every row), and the delegation rules are on in every arm. 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: forged chat turns and optimized injections** (suite v1.2.2; ASR is executed effect). Left: **ChatInject** (Chang et al., 2026) on qwen3:8b, 560 Workspace and 105 Slack pairs. Right: **AutoDojo** (Ma et al., 2026a), injections optimized against Progent, on gpt-4o-mini, 144 Banking, 105 Slack and 140 Travel pairs. SPILLWAY in bold; its AutoDojo row uses the rules written on four failing pairs (held-out sets: Appendix A); its ChatInject row predates them and is unchanged by them. DRIFT and ROPE were not run on AutoDojo. ‡Authors' code with a compatibility overlay; †authors' planner and validator; ^{c}an injected recommendation (content, Sec. 2).

	ChatInject (qwen3:8b)			
	Workspace		Slack	
Method	Util	ASR	Util	ASR
Undefended	.052	.245	.095	.648
SPILLWAY	**.371**	**.000**	**.286**	**.000**
MELON	.084	.020	.095	.000
Progent‡	.161	.005	.057	.010
DRIFT†	.393	.062	.352	.019
ROPE‡	.157	.002	.162	.095

	AutoDojo (gpt-4o-mini)					
	Banking		Slack		Travel	
Method	Util	ASR	Util	ASR	Util	ASR
Undefended	.444	.354	.524	.571	.329	.364
SPILLWAY	**.465**	**.000**	**.505**	**.000**	**.543**	**.007**^{c}
MELON	.340	.021	.286	.000	.293	.043
Progent‡	.306	.104	.467	.124	.414	.150
Spotlighting	.431	.306	.543	.514	.343	.307
PI detector	.403	.049	.219	.067	.214	.029

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 the highest mean utility (.370 against .300 for the floor); the sanitizer keeps injected targets away from the judge, and the judge recovers over-blocked calls, mainly on Banking, as P2 predicts. Removing only the span veto changes no attack outcome and moves utility by two Workspace pairs (last row): with the sanitizer on, the agent never proposed a removed target, so the veto was never reached. On this grid the sanitizer protects by keeping injected targets out of the agent's view; the veto covers an agent that still proposes a removed target, which the grid does not produce.

Stress tests and transferred optimizer attacks. Chat-template forging (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 $.095 \rightarrow .286$, Workspace $.052 \rightarrow .371$). 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 is a transfer test, not an adaptive attack on SPILLWAY: its injections were optimized against another defended agent. Progent, the defense these injections were optimized against, executes the attack in 10 to 15% 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 one injected hotel recommendation in a final answer (a content effect, Sec. 2) and no call to an attacker's target. 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. In the DRIFT set, a second transfer to an attacker-written payee that the check does not count makes two of 144 pairs, or .014 (Sec. 7). The rules change no earlier result (Appendix B).

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 2 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 (a cross-session comparison), against $+4.9$ and $+7.9$ s for MELON, $+1.6$ and $+2.3$ s for Progent, and $+9.9$ and $+10.5$ s for DRIFT. A smaller judge also suffices: with a 4-bit gemma4:12b judge, the qwen3:8b cells of Table 1 and the ChatInject panel keep zero attack success, at mean utility .348 instead of .370.

6 Related Work

Beyond the defenses of Sec. 1, information-flow and capability systems (Costa et al., 2026; Zhong et al., 2025; An et al., 2025; Wu et al., 2024) also constrain untrusted data. Task Shield (Jia et al., 2025) and DRIFT (Li et al., 2025b) also watch both boundaries, but ask the backbone 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; our admission is fixed by the tool schemas and a role lexicon per benchmark, with no per-task router, and is paired with 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) put an LLM at both boundaries and report zero AgentDojo attack success. 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. PACT (Fan et al., 2026) checks the origin of every tool argument and reports full security on AgentDojo for the three strongest of its five models. Recoverable information-flow control (APPA (Kravchenko et al., 2026)) and typed dependency graphs (AgentArmor (Wang et al., 2025)) keep richer per-step state than our flat index.

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. The AutoDojo injections were optimized against Progent; an optimizer run against SPILLWAY itself (Zhan et al., 2025; Hofer et al., 2026) is future work.

8 Conclusion

We presented SPILLWAY, a prompt-injection defense for tool-using agents that separates target admission from sanitization: where an action may go is decided from the tools' typed returns and the user's task, the sanitizer removes the instruction-like regions its rules match before the model reads them, and a local judge may restore only refused calls whose targets the sanitizer did not remove. Identity is decided with structure, and model judgment is spent only where structure runs out. Across three backbones and four AgentDojo suites, no templated injection redirected an action to an attacker's target, while SPILLWAY kept the highest twelve-cell mean utility under attack ($.484$, undefended $.417$) and 96% of benign utility with no defense-triggered backbone re-run; the same held under chat-template forging on qwen3:8b and against routed-origin enforcement on seven delegated-routing suites. Needing no per-task plan and no backbone re-run, SPILLWAY can sit in front of the personal agents now reaching users' mail, files, and payments as an efficient, auditable egress layer that keeps them useful; tool ecosystems with typed outputs, such as MCP servers, are where we expect it to extend next.

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: to draft sections and polish prose, which the authors revised; to surface candidate related work, which the authors read before citing; to assist research execution by writing evaluation, aggregation, and plotting code to the authors' specifications and running the experiment harness; and 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

An et al. 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)*, 2025.

Anthropic. Model Context Protocol Specification. https://modelcontextprotocol.io/specification, 2025.

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.

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. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026. 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, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating Prompt Injections by Design (CaMeL). In *4th IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026. arXiv:2503.18813.

Edoardo Debenedetti et al. 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.

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)*, 2023.

Keegan Hines et al. Defending Against Indirect Prompt Injection Attacks With Spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

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

Feiran Jia et al. 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)*, 2025.

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, 2026.

Evan Li, Tanwi Mallick, Evan Rose, William Robertson, Alina Oprea, and Cristina Nita-Rotaru. ACE: A security architecture for LLM-integrated app systems. *arXiv preprint arXiv:2504.20984*, 2025a.

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)*, 2025b.

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?, 2026. 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)*, 2026. arXiv:2601.05755.

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, 2026. Concurrent 2026 preprint; no peer-reviewed venue.

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.

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, 2024. AgentDojo's default transformers-based prompt-injection detector.

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

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.

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. 7101–7117, 2025. doi: 10.18653/v1/2025.findings-naacl.395.

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 et al. MELON: Provable Defense Against Indirect Prompt Injection Attacks in AI Agents. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 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.

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

Table 6: **AutoDojo** on gpt-4o-mini (144 Banking, 105 Slack, 140 Travel pairs): trace-level, executed (exec.), and in-scope ASR. SPILLWAY's one Travel hit is an injected recommendation (content), removed at in-scope. DRIFT and ROPE were not run on AutoDojo.

	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	.007	.007	.000
MELON	.021	.021	.021	.000	.000	.000	.043	.043	.000
Progent	.104	.104	.104	.210	.124	.124	.150	.150	.121
Spotlighting	.306	.306	.306	.514	.514	.514	.307	.307	.279
PI detector	.049	.049	.049	.067	.067	.067	.029	.029	.029

Table 7 gives the benign utility of every breadth cell, whose twelve-cell means are the benign column of Table 2, and Table 8 gives the delegated-routing grid of Figure 3 per suite, with ROPE under its authors' router and under the agent's own model.

B REPRODUCIBILITY

Agent. Every method runs on the same agent request: AgentDojo's default system prompt (469 bytes) 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; Spotlighting appends its delimiter instruction, and DRIFT appends its own environment block, which tells its agent to complete tasks without asking for confirmation. gpt-4o-mini-2024-07-18 is served by the OpenAI API; qwen3-coder:30b and qwen3:8b by 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.

Table 7: **AgentDojo** breadth grid, benign utility per backbone, suite, and method (16, 21, 20, and 40 user tasks in Banking, Slack, Travel, and Workspace).

Cell	Undef.	SPILLWAY	MELON	Progent	DRIFT	Spotl.	PI det.
Banking							
gpt-4o-mini	.562	.562	.438	.375	.438	.500	.500
qwen3-coder:30b	.375	.375	.375	.312	.688	.375	.500
qwen3:8b	.500	.438	.500	.375	.562	.562	.438
Slack							
gpt-4o-mini	.762	.714	.714	.667	.714	.619	.238
qwen3-coder:30b	.667	.571	.667	.476	.286	.524	.190
qwen3:8b	.619	.476	.571	.190	.429	.429	.190
Travel							
gpt-4o-mini	.500	.550	.550	.550	.400	.600	.300
qwen3-coder:30b	.500	.500	.550	.450	.100	.550	.250
qwen3:8b	.250	.250	.250	.150	.300	.300	.050
Workspace							
gpt-4o-mini	.825	.875	.775	.800	.525	.825	.625
qwen3-coder:30b	.500	.475	.500	.475	.575	.600	.425
qwen3:8b	.425	.450	.425	.425	.500	.450	.275

Table 8: Delegated-routing grid (**AgentDojo** suite v1.2.2 above the rule, **AgentDyn** below), gpt-4o-mini: utility under attack / in-scope ASR, one run per suite (144, 105, 140, 560, 180, 200, 180 pairs from top); the mean is unweighted over suites.

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

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 2 and 3, 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.

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; on the ChatInject panel ROPE's router is the agent, and in Figure 3 it is its authors' Claude Opus 4.8 with cached scopes.

Attacks and scoring. The basic attack is AgentDojo's `important_instructions`; AutoDojo replays variant 0 of its released gpt-4o-mini injections (commit 89413642). 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 (Sec. 7) is not tested here. Utility under attack is .548 for SPILLWAY, .611 for the floor-only arm, and .603 undefended.