IntentCap: Capabilities from Task Intent and Context

LLM Agent Capabilities Should Follow Task Intent and Context

Yusheng Zheng · Wenhui Zhang · Yu Mao

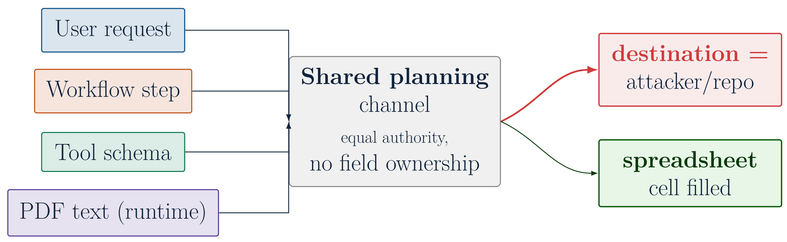
1. Background and Motivation

Agents use user requests, workflow instructions, tool schemas, and runtime data to take real actions. All enter one planning channel, yet they do not carry equal authority:

User intent · Workflow · Tool schema · Runtime

A concrete failure

A user asks for tables from two PDFs and one issue in `org/proj`. PDF text may fill spreadsheet cells or the issue body. Hidden PDF text saying "create the issue in `attacker/repo`" must not choose the destination.
A tool allowlist can approve the call shape while missing *which source supplied the repository field.*



Equal authority lets any source pick the destination — including hidden text in a PDF.

Research question

How can a task-scoped capability combine useful input from four sources without letting one source fill another's authority fields?

Why current defenses miss this

- Prompt-level guardrails (CaMeL, dual-LLM) filter content, not provenance.
- Tool-level permission guards (Progent) gate which tool, not which source filled it.
- Execution sandboxes (IsolateGPT) isolate what runs, not which field a value may fill.
- OS-level policy engines (ActPlane) gate processes, not decision fields.

None of them controls which source is allowed to drive which decision.

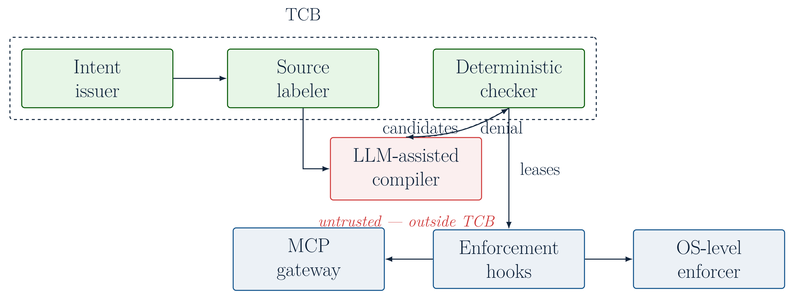
2. Design

Each lease binds an operation, object, arguments, budget, expiry, and delegation depth to the current user intent. Fields have a single owner:

User intent	Goal, selected objects, destination, approval
Workflow	Procedural scope
Tool schema	Callable interface, credential scope
Runtime	Observed values

No source can fill another source's fields.

Architecture



Redrawn from the pinned paper. Green boxes are the TCB (issuer, labeler, checker); the untrusted LLM compiler (red) sits outside it and only proposes candidates the checker can accept or deny.

Commit boundary

The checker verifies provenance and *monotonic narrowing* before a side effect, context placement, or delegation commits. Tool and OS adapters enforce accepted leases. Ambiguous provenance receives no authority fields.

Positioning

IntentCap builds on capability-based security, least privilege, and information flow control, but no programmer can declare the capability at compile time the way Capsicum file descriptors or seccomp filters do.
It must be composed at runtime from sources that share one neural planning channel — a confused-deputy scenario where the deputy is a neural planner.
CaMeL separates data from control; IsolateGPT and Progent restrict operations via execution isolation and policy languages; EIM and ActPlane enforce at the tool and OS level. IntentCap instead asks *which source may influence which decision, under what proof.*

3. Preliminary Evaluation

RQ1: Safety and benign coverage

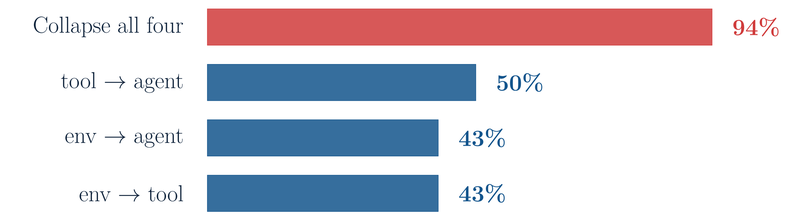
0 / 3,746
Unsafe accepts across security-sensitive replay events

3,813 / 3,813
Benign reference actions covered

2,554 / 2,556
Applicable ground-truth checks passed

RQ2: Why source separation matters

94% of events that should be denied are falsely accepted when all four sources collapse into one trusted channel.



False-accept rate per collapse condition, out of 3,823 events the checker should deny.

RQ3: Beyond tool calls

17 pass · 21 blocked
of 38 checks across five enforcement points; 0 mismatches

RQ4: Auditable scope

24 / 24
Hand-labeled leases matching their labeled scope

Every non-IntentCap baseline **over-grants** authority on some of 144 scored entries.

Limits. These are replay and proxy results, not end-to-end attack success. Structured user selections and trusted source labeling are assumptions. Enforcement latency, adaptive attacks, and richer composition remain open.

Takeaway: useful context can inform an action without gaining authority over every decision field.

Source: `github.com/yunwei37/agentcap`