THOX.ai Research · Technical report, comprehensive edition
v1.1.0 · 2026-11-04

Preprint, not peer reviewed

ThoxWork for the Enterprise: Governed Digital Employees, MeshGroups, Rolodex and Evidence-First Teamwork

An evidence-bounded account of THOX's third platform: what the code does today, where the records disagree, and what a browser lab and timed experiments show about the rules.

THOX.ai Research · Authors: Craig Ross, Chief Executive Officer, and Tommy Xaypanya, CTO and Co-Founder (corresponding author), THOX.ai LLC, Cedar Park, Texas · ORCID 0009-0005-9872-3673 (Ross), 0009-0000-4425-9565 (Xaypanya)

ABSTRACT

Companies are adopting AI agents faster than they are governing them, and the open question for a buyer is no longer which model to use but which rules the work runs under. ThoxWork™ is THOX's third platform, the workspace where people and named Digital Employees share assignments, MeshGroups bound who may use which resources (planning is a preview today and never executes), Rolodex carries the assign, check-in and review loop, and the platform is designed so that every decision leaves a record. This comprehensive edition, v1.1.0, supersedes the published v1.0.1 edition of the ThoxWork paper [1] and extends it in five ways. First, it audits the code behind the model: 10 THOX code bases at stated commits, 3,747 tests passed when the suites were rerun for this paper, with 3 failures, and real screenshots and recordings of ThoxWork, ThoxEmployee, the Rolodex directory and validated dispatch. Second, it states the lineage honestly: the ThoxWork desktop and server shell is built on the open-source OpenWork project and the OpenCode engine, and THOX's contributions are the governance layers built on top. Third, it reconciles the record: three MeshGroup admission definitions and five assignment state machines are shown side by side with an owner, not merged. Fourth, it measures. A browser Evidence Lab re-implements the ThoxWork routing planner and agrees with the product's own function on 12,000 of 12,000 seeded previews; it agrees with the reference admission kit on 20,000 of 20,000 decisions. Comparing the two admission definitions on the same 20,000 requests, 56.3% of the reference rule's placements land on a node smaller than the demand under our demand model, while the pooled rule admits 4,333 requests the single-node rule cannot serve. For evidence, hash links alone missed tail truncation and a rebuilt suffix in all 500 trials each; a session HMAC key catches the rebuild and an external head anchor catches both, at 27.6 microseconds per keyed record. Fifth, it replays the protocol experiments of the paper of record against the live ThoxWork core service, the platform’s own server implementation of the model: 0 of 12,000 policy violations leaked, handoff chains were complete on all 5,000 reassigned tasks, the admission boundary held at exactly 120 of 121 resource units, no review-gate bypass succeeded in 6,000 attempts, and 50 parallel assignments admitted exactly to pooled capacity with zero over-admission, with three independent judges agreeing on all 102 workspaces (Section

> 7.7). Nothing here is measured on a customer deployment, and ThoxWork remains Coming Soon.

CONTENTS

Executive summary

1. Introduction

2. Background and related work

3. System and threat model

4. Design

5. What is built

6. The Evidence Lab

7. Evaluation

8. Governance mapping

9. Discussion and limitations

10. Conclusion

11. Roadmap and founder decisions

Appendix A. Reproducibility

Appendix B. Notation

Appendix D. Document control, authorship and publication boundary

References

Your AI. Your Data. Your Rules.™ · THOX®, ThoxOS™, MeshStack™ and ThoxWork™ are marks of THOX.ai LLC. OpenWork and OpenCode are the projects of their respective owners.

This edition: version DOI 10.5281/zenodo.23965052; concept DOI 10.5281/zenodo.23963935.

Executive summary

THOX builds local-first AI in three platforms: ThoxOS™ for ownership on each device, MeshStack™ for private distributed capacity [2], and ThoxWork for the daily work of a company. ThoxWork's thesis is that an enterprise needs an operating model for AI colleagues, not another agent framework: named Digital Employees with roles and sponsors, team-scoped access through MeshGroups, an assign, check-in and review loop through Rolodex, and a record of who did what and who approved it. The THOX ThoxWork paper v1.0.0 stated that model as five invariants and checked them in a seeded simulator [1]. This edition asks what the code does, how the records agree, and what the rules cost.

3,747

tests passed when 18 suites in 9 THOX code bases were rerun for this paper (3 failed, causes in Section 7.1)

MEASURED

12,000 / 12,000

seeded routing previews on which the Evidence Lab planner and ThoxWork's own planner agree on status, reason, selection and every per-node denial

MEASURED

56.3%

of reference-rule placements land on a node smaller than the request, because that rule never compares demand with capacity (illustrative demand model, 4,570 placements)

MEASURED

0% / 100%

tail truncation caught by hash links alone versus with an external head anchor, 500 seeded trials each on 1,000-record chains

MEASURED

0 / 12,000

policy violations leaked when the protocol experiments were replayed against the live ThoxWork core service, the admission boundary held at exactly 120 of 121 resource units (Section 7.7)

MEASURED

> PRODUCT IMPLICATIONS
>
> First, the Digital Employee stack's governance layers, in ThoxEmployee and the Digital Employee contract, are real and tested: a fail-closed action gateway, a person-only approval gate, a seven-state assignment contract, hashed handoffs, contradiction tags that never merge, and a deterministic MeshGroup planner that is deliberately preview-only; their integration into the ThoxWork shell is in development. Second, the record needed one admission rule, and the ruling has been made. The paper of record states a pooled-capacity rule, the reference kit places whole requests on one node without a capacity check, and the product planner orders by locality without either; Section 4.4 shows the three side by side, and the pooled rule now governs: it is implemented in the ThoxWork core service and measured against the live API in Section 7.7, with the ruling recorded in Section 11. Third, evidence must be keyed and anchored. The built hashes detect accidental change, not an insider who rewrites both a record and its hash; a session key plus a head anchor held by a second party closes that gap for a few microseconds per record. Fourth, the platform credits its foundation. The ThoxWork shell is built on OpenWork and OpenCode, and their upstream license notices travel with every distribution.

1 Introduction

A company that lets AI agents do real work needs answers to four questions before the agents are useful: who is doing this work, what are they allowed to touch, who approved the consequential step, and how do we know afterwards. Workplace benchmarks explain why the answers cannot be left to the agents. The best agent in TheAgentCompany completes 30 percent of tasks in a simulated software company on its own [3]; on OSWorld the best model reaches 12.24 percent against 72.36 percent for people [4]; tau-bench finds that a strong model succeeds on under half of its tasks, with pass^8 below 25 percent in retail [5]. Multi-agent systems add their own failures: MAST annotated more than 1,600 traces across seven frameworks and placed inter-agent misalignment and task verification among its three failure categories [6]. A platform for agent work therefore has to ship review, scoping and records as mechanisms, not as advice.

ThoxWork is THOX's answer, positioned in the paper of record in one line: ThoxWork organizes the work, Rolodex connects you to your Digital Employee team, MeshGroups define team access, and MeshStack connects approved resources [1]. The terms name distinct responsibilities. A **Digital Employee** (a DigitalHuman staffed into a role, offered as ThoxEmployees) is a named, role-holding synthetic colleague with a human sponsor. **ThoxEmployee** is the local employee operating layer that runs, governs and exchanges those colleagues. **Rolodex** is the team interface: assign a task, check in, review the result. A **MeshGroup** is membership with explicit actions plus the exact workspaces, pools and nodes the members may use. **ThoxWork** holds these together as shared workspaces, assignments, policy and evidence.

The THOX ThoxWork paper v1.0.0 stated the model and evaluated it in a deterministic simulator, with every result labelled simulated [1]. It deliberately left the code out of scope. A buyer, an auditor or a peer reviewer will ask the next question: does the shipped code implement the stated rules, and where it does not, which record wins? This paper answers that question with evidence gathered for it on 2026-11-04.

Contributions.

1. An evidence-bounded account of the system across 10 repositories at stated commits: architecture, lineage, licenses and 3,747 tests passed on rerun for this paper, with 3 failures, with real product screenshots and recordings (Section 5).
2. A reconciled model: one vocabulary that maps five assignment state machines, three MeshGroup admission definitions and three handoff and evidence mechanisms, with every disagreement shown side by side and assigned an owner (Section 4).
3. The ThoxWork Evidence Lab, a single-file browser instrument that runs the gateway, approval, lifecycle, handoff, admission and evidence rules, captures screenshots and recordings into a hash-chained ledger, and is checked for agreement against the product's own planner code (Section 6).
4. Measurements: planner agreement, admission-rule divergence, evidence-chain detection limits and cost, and decision latency, each from scripts released with the paper (Section 7).
5. A governance mapping of the mechanisms to NIST AI RMF, the NIST generative AI profile, ISO/IEC 42001, the EU AI Act, OWASP and NIST zero trust guidance, stated as alignment, not conformance (Section 8).
6. The errata for v1.0.0, reconciled into the text of this edition rather than carried as a register; a replay of the paper of record’s protocol experiments against the live core service (Section 7.7); and the decisions the record needed, with the admission ruling recorded (Section 11).

2 Background and related work

Every property in the ThoxWork model has close prior work. The contribution, if there is one, is where the properties meet, so this section names the nearest precedent for each and states the difference.

Enterprise context

Adoption is ahead of governance. McKinsey's 2025 survey of 1,993 participants in 105 nations found 62 percent of organizations at least experimenting with AI agents, 23 percent scaling an agentic system in at least one function, and no more than 10 percent scaling agents in any given function [7]. A 2025 survey of 353 IT professionals found 82 percent of organizations already using AI agents and 44 percent with policies in place to secure them [8]. Microsoft's 2026 Work Trend Index, with 20,000 AI-using knowledge

workers in ten markets, attributes about twice as much of reported AI impact to organizational factors as to individual ones, an associational result rather than a causal one [9]. MIT CISR describes digital colleagues as agents managed like employees and reports that 75 percent of 132 surveyed organizations predicted a 25 percent average rise in revenue per employee over three years [10]. Standards bodies are moving: NIST launched an AI Agent Standards Initiative [11] and an NCCoE concept paper on agent identity and authorization [12], and the Cloud Security Alliance proposed an Agentic Trust Framework with maturity levels from Intern to Principal [13, 14]. The Model Context Protocol and the Agent2Agent protocol, announced by Anthropic and Google [15, 16], are now stewarded under the Linux Foundation [17, 18]; they standardize agent-to-tool and agent-to-agent traffic, not the workspace where people and agents share accountable work.

2.1 Multi-agent systems and agent workplaces

Role-based multi-agent frameworks established that assigning LLM agents distinct roles and passing structured work between them improves results: CAMEL's role-playing agents [19], MetaGPT's standard operating procedures for an assembly line of roles [20], ChatDev's chat chain across a software company [21], AutoGen's conversable agents [22] and AgentVerse's dynamic expert groups [23]. Generative Agents showed persistent identities with memory and reflection [24]. Surveys organize the field [25, 26, 27, 28]. Magentic-One coordinates specialist agents under an orchestrator [29]. These systems run handoffs inside one autonomous session; none records each handoff as an approvable, verifiable artifact, and none scopes resources by team. In a controlled simulation of event-driven orchestration across registries of under 10, 20 to 80 and 200 agents, both tested systems scored lower at enterprise scale, the authors suggest a larger and more confusable discovery space may contribute, and a minimal task manager cut high-priority queue latency by 14 to 75 percent [30], and DynaFlow learns free-form orchestration programs that make the hand-off between stages explicit [31].

Benchmarks of agents at work include TheAgentCompany [3], WorkArena [32], CRMArena [33], OSWorld [4] and tau-bench [5]. CorpGen simulates whole corporations staffed by autonomous digital employees and is the closest use of that term in the literature, as a benchmark rather than a governed product [34].

2.2 People in the loop

Magentic-UI's action guards are the closest published analogue to an approval gate, built for one user and one task [35]. Feng, McDonald and Zhang define five levels of agent autonomy by the user's role, one of which is approver [36]; Cocoa lets plan steps be assigned to the person or the agent [37]. Collaborative Gym reports that collaborative agents beat autonomous ones with real users at 86, 74 and 66 percent win rates across three tasks [38]. Field evidence points the same way: in a randomized experiment that assigned 2,234 participants to human-human and human-AI teams, human-AI teams produced 50 percent

more ads per worker, and their collaboration was more task-oriented and showed more delegation [39]; a second preregistered experiment with 1,258 participants found that pairing agent and human personalities changed output quality, which bears on how named personas are presented [40]. The Collaborative Human-Agent Protocol (CHAP) makes the human override a structured event with a diff, a rationale and a content hash, turns a shift handoff into a portable envelope, and makes an approval a signed decision that can be replayed [41]. It is the closest specification to the ThoxWork team layer; its abstract does not describe admission of work against approved resources, which is where MeshGroups differ.

2.3 Governance, regulation and security

Standards converge on a named, competent person who can stop or reverse an AI system. NIST AI 100-1 asks organizations to define roles for human-AI configurations and oversight (GOVERN 3.2) and to document oversight processes (MAP 3.5) [42]; the generative AI profile adds governance, content provenance, pre-deployment testing and incident disclosure as considerations [43]. ISO/IEC 42001 is a management-system standard for organizations that provide or use AI [44]. The EU AI Act requires record-keeping (Article 12), effective human oversight including the ability to override, reverse or stop (Article 14), and deployer log retention of at least six months (Article 26(6)) for high-risk systems [45]; the Digital Omnibus on AI moved the high-risk application date for Chapter III Sections 1 to 3 to 2 December 2027 [46]. A general workspace is not high-risk by default, but Digital Employees that allocate tasks to people or evaluate their performance would bring a deployment into Annex III scope.

OWASP names excessive agency (LLM06) and prompt injection (LLM01) among the top risks for LLM applications and prescribes least privilege, human approval for high-impact actions and complete mediation outside the model [47]; its agentic guidance and 2026 top ten add tool misuse, identity and privilege abuse, and human-agent trust exploitation [48, 49]. Indirect prompt injection turns retrieved content into instructions [50]; AgentDojo and InjecAgent measure how often agents follow them [51, 52]. In a multi-agent workspace a handoff moves content from one agent's context into another's, so the access check has to happen again at each handoff and at each group boundary. NIST zero trust guidance grants access per session with least privilege and states that authorization to one resource does not grant another [53]; attribute-based access control supplies the policy model [54]. South et al. extend OAuth and OpenID Connect to delegated agent credentials [55], and Chan et al. argue for agent identifiers, activity logs and infrastructure for accountability [56, 57].

2.4 Provenance and tamper-evident records

Secure audit logs that an attacker can neither read nor undetectably modify date to Schneier and Kelsey [58]. Crosby and Wallach showed that tree-structured logs give compact membership and consistency proofs where a

plain hash chain needs a linear scan [59]; Certificate Transparency put those proofs into production for public logs [60, 61]. W3C PROV defines entities, activities, agents and delegation [62, 63]; in-toto verifies signed per-step attestations against a declared layout [64]; SLSA defines build provenance [65]. For agents, PROV-AGENT extends PROV to agent interactions [66], Auditable Agents defines five auditability dimensions and reports 8.3 ms median overhead for mediated, tamper-evident execution [67], and a September 2026 study shows that agents can delete their own traces, so logging must sit outside the agent's control [68]. The primitives are SHA-256 [69] and HMAC [70]. Local-first software keeps the primary copy of data with its owner [71], which is where THOX places workspace records.

2.5 Positioning

Generative Agents [24]	yes	not stated	not stated	not stated	not stated
MetaGPT [20]	partial	not stated	partial	not stated	not stated
ChatDev [21]	partial	not stated	partial	not stated	not stated
AutoGen [22]	partial	partial	not stated	not stated	not stated
Magentic-UI [35]	not stated	yes	not stated	not stated	not stated
CorpGen [34]	yes	not stated	not stated	partial	not stated
Authenticated Delegation and Authorized AI Agents [55]	not stated	partial	not stated	partial	partial
PROV-AGENT [66]	not stated	not stated	partial	not stated	partial
Auditable Agents [67]	not stated	not stated	not stated	not stated	yes
CHAP [41]	not stated	yes (signed decisions)	yes (handoff envelopes)	not stated	partial (content hashes)
ThoxWork, built today	yes (named employees with sponsors)	yes (person-only, role-named)	partial (per-handoff body hash)	yes, preview-only (MeshGroups)	partial (hashes detect accidental change)
ThoxWork with the lab's proposals	yes	yes	yes (chained, keyed)	yes	yes (keyed chain plus external anchor)

Table 1. Positioning against the closest prior work. Entries for other systems are judged from their abstracts only ("not stated" means the abstract does not say, not that the system lacks it). ThoxWork rows are judged from code reviewed for this paper (Section 5).

No single prior system combines the five properties, and each property alone has precedent. The defensible claims are therefore about the seams: an approval stored in the same tamper-evident record as the action it authorizes, an access check repeated at every handoff and every group boundary, and evidence written by the workspace rather than by the agent [68].

3 System and threat model

Actors. People (members, sponsors, reviewers and approvers), Digital Employees (synthetic colleagues with a role, an autonomy level and a human sponsor), the ThoxWork workspace and its server, the ThoxEmployee runtime, and the nodes and sites that serve models. A MeshGroup binds actors to workspaces, model pools and nodes.

Assets. Workspace data at a stated data class; the authority to cause an external effect (send, write, execute, call a tool); the record of who did what and who approved it; and node capacity owned by the organization or by individual device owners.

Adversaries and failure sources. (A1) a Digital Employee following injected instructions from retrieved content [50]; (A2) a member or employee reaching resources bound to another group; (A3) an insider or compromised process with write access to stored records, including one that recomputes unkeyed hashes; (A4) a stale or forged node observation that makes an unavailable or unverified node look eligible; (A5) a careless or rubber-stamp approval (automation bias, Article 14(4)(b) [45]); (A6) an agent that deletes or rewrites its own trace [68]. Out of scope: compromise of the host operating system, of the person's own credentials, and of the model weights.

Requirements. The paper evaluates the design against eight requirements.

R1	Fail closed	An action not explicitly known and permitted is denied; a tenant cannot widen a deployment ceiling.
R2	People decide consequential steps	An external effect waits for a person when any approval reason holds; an agent cannot approve; each request names and records the role required to decide it; a decision is final.
R3	Legal lifecycle	An assignment moves only along its contract's transitions; every refused move is recorded.
R4	Team-scoped access	A request is planned only within its MeshGroup's bindings and the member's explicit actions; cross-site and remote use need every policy level to allow it.
R5	Admission honesty	Work that cannot be served is refused with a reason, never silently routed elsewhere.

R6	Recorded handoffs	Each change of hands leaves a written handoff that a reviewer can verify.
R7	Tamper-evident record	Editing, reordering, deleting, truncating or rebuilding the record is detectable by a party that did not write it.
R8	Contradictions surface	Conflicting facts are kept and cross-tagged, never merged or dropped.

Table 2. Requirements used in Sections 4 to 7.

4 Design: the ThoxWork model, reconciled

This section states the model at the level of rules. It reconciles the paper of record [1] with the code reviewed for this edition and keeps implementation particulars (ports, hostnames, thresholds, per-role settings and persona instructions) out of scope.

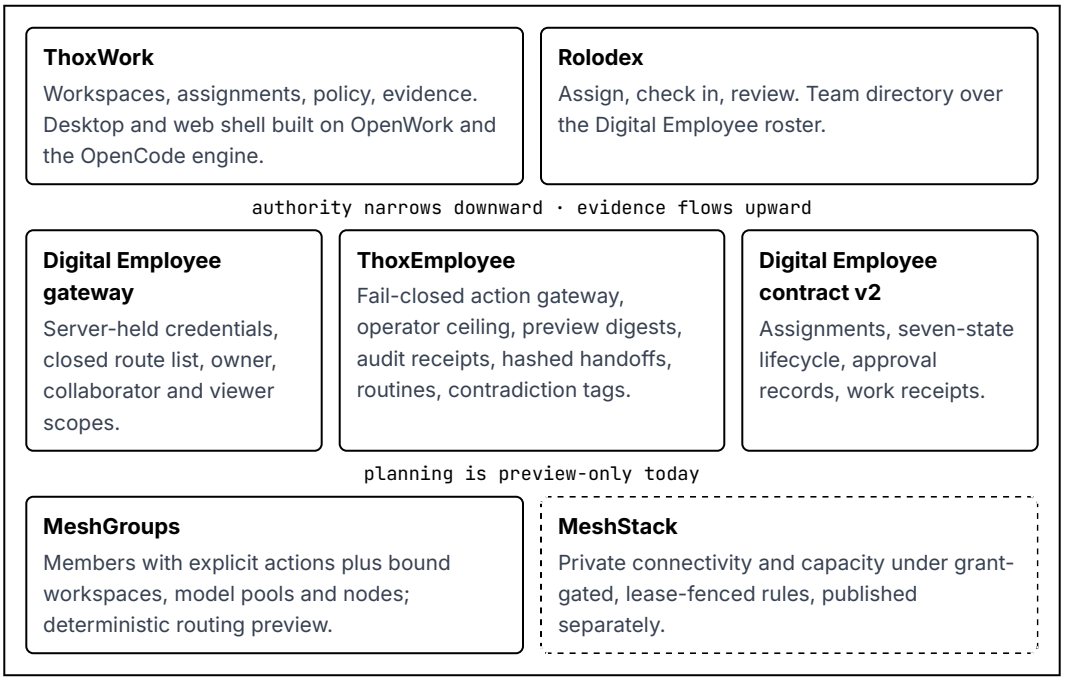


Figure 1. ThoxWork layers as built and as published. Dashed: a separate THOX platform cited from its own paper.

4.1 Digital Employees and the action gateway

A Digital Employee has an identity, a role, a sponsor and an autonomy level. In ThoxEmployee every gated action passes a policy gate before it can take effect, and the gate fails closed. Algorithm 1 gives the order of evaluation as built; the first rule that decides is recorded with the decision, and an allowed effect is wrapped in receipts: a running entry before it and its outcome after. A write or execute action can be previewed first; the preview yields a digest of the request and the policy in force, and an execution that presents a digest for different content is refused.

```
gate(actor, action, payload, channel):
  1. on the deployment deny list    → DENY deployment_deny
     (tenant policy cannot widen it)
  2. not a known action             → DENY unknown_action
  3. on the tenant deny list        → DENY deny_actions
  4. actor not on channel roster    → DENY channel_membership
  5. payload has a secret marker    → DENY secret_leak
  6. payload over the size cap      → DENY size_limit
  7. command execution disabled,    → DENY exec_disabled,
     empty or not allowlisted            exec_empty, exec_allowlist
  8. otherwise                      → ALLOW default_allow
     (running receipt before the effect, outcome after)
```

4.2 Approval gate and assignment lifecycle

Work assigned to a Digital Employee is an assignment under the Digital Employee contract. Before any external effect, the assignment requires approval when any of four reasons holds (Equation 1): the caller kept the external-effect gate on, which is the default; a responsibility of the role requires a person; the data sensitivity is confidential or restricted; or the effective autonomy, the lower of employee and role autonomy, is zero.

$$\begin{aligned}\mathrm{needs_approval}(a) \;=\; & g_a \;\vee\; \exists r \in \mathrm{resp}(a) : h_r \\ & \vee\; \sigma_a \in \{\mathrm{conf},\, \mathrm{restr}\} \;\vee\; \min(\alpha_{e(a)}, \alpha_{\rho(a)}) = 0\end{aligned} \tag{1}$$

Only a person can decide. Subjects that identify an employee, agent, assignment or service are refused; each request names and records the role required to decide it, which rises with sensitivity, and the Digital Humans gateway checks that role before it accepts a decision; a second decision on the same request is refused. Approval returns the assignment to running; rejection blocks it. The contract defines seven states and eighteen legal transitions (Table 3); every other move is refused before any event is emitted, and every change increments a version used for optimistic concurrency.

queued	running, blocked, failed, cancelled
running	awaitingApproval, blocked, completed, failed, cancelled
awaitingApproval	running (approve), blocked (reject), failed, cancelled
blocked	queued, running, failed, cancelled
completed	terminal
failed	queued (retry)
cancelled	terminal

Table 3. Assignment lifecycle of the Digital Employee contract: 7 states, 18 legal transitions.

Five state machines describe work across the THOX repositories: the contract above, a gateway review inbox, the ThoxEmployee review workspace, the ThoxEmployee kernel's task states and Migration Studio's plan-first states. Rolodex, as described in the paper of record, presents four user states and one

exception [1]. Table 4 maps them onto the contract. One disagreement matters: the contract maps rejection to blocked, while the gateway's mapping sends it to cancelled. The contract is the authority; Section 11 asks for the gateway to align.

Queued	queued	queued	queued
In Progress	running	running	busy
In Review	awaitingApproval	awaiting_approval	(no human queue; operator-replayed preview digest)
Done	completed	completed	done
Changes requested (back to In Progress)	blocked, then running	rejected, then queued	blocked
Denied (terminal, at admission)	refused at admission; no assignment created	not modelled	blocked (policy refusal)

Table 4. One vocabulary across the assignment state machines found in the code. The gateway inbox and Migration Studio follow the same pattern with their own names.

4.3 Rolodex, ordered handoffs and validated dispatch

In the paper of record, Rolodex is the loop through which a person assigns work, checks in and reviews it [1]. In the code, the name covers three different things, and this paper keeps them apart. The **Rolodex directory** is a web app over the nine customer-facing Digital Employees: an ID card for each, a recorded introduction, chat with one employee at a time after an explicit consent to hosted processing, and a team builder whose output is a selection marked for review that grants no tools and no permissions. It has no ordered task and no handoff. **RolodexGo** is a native client for iPhone, iPad and Mac that pairs with a private server and carries the same roster as a contacts client, with the Sadie Link relay for agent contact. Its Teams and Work tabs, the ordered-task surface, exist as a built, test-covered changeset that has not landed in the shipped client; the shipped client is the contacts client, and no ordered task runs in it today. The flow that the public site describes as building a team, running an ordered task and reviewing written handoffs is the **ThoxEmployee guest demo** on thox.ai, which reuses the ThoxEmployee workspace engine.

The guest demo's engine is the clearest built example of the assign, check-in and review loop. A person composes a task for a small team (the interface allows up to three employees), always with approval required. The engine runs one step per employee in order, and each step receives every earlier handoff as context. A step can only claim the next pending position, every write is fenced by a revision check, and each run and step carries an idempotency key, so a retried or duplicated step cannot double-run or skip ahead. Each step leaves a written handoff in the audit trail, the task then waits for a person, and approve or reject is a recorded decision that performs no external action. The hosted demo adds daily call limits per session, per network address and in total, and

expires a guest session after 24 hours. Table 4's first column is this loop's vocabulary.

Validated dispatch is the pattern THOX uses where one answer is assembled from several workers. A request passes input guardrails, is planned, executed by workers in parallel, cross-checked and synthesized, then judged by a validator in two layers: fixed rules first, which send obviously empty or refused output back for revision without a model call, and a model judge second. A pass needs both layers to agree. If the judge's verdict cannot be read, the dispatcher withholds the answer by default. If the revision budget runs out, it returns the last answer marked as not validated. The precise claim is therefore that every answer carries its verdict and an answer the validator cannot rule on is withheld, not that nothing unvalidated is ever returned.

4.4 MeshGroups: team-scoped access and three admission definitions

A MeshGroup belongs to one organization and binds members to workspaces, model pools and nodes. Each member holds explicit actions in that group (planning inference, reading the workspace, proposing tool use), and membership in another group is never added in. A Digital Employee's request also requires its human sponsor to be active and to hold the same actions in the same group, so an employee can never reach further than the person accountable for it. The ThoxWork planner evaluates a routing request in two stages. First, ordered early denials stop requests from the wrong organization, from unknown or revoked principals, without the required actions, without an accountable sponsor, against an unbound workspace or pool, for an unsupported task, or from an unknown source. Second, every node bound to the group and serving the pool is classified by locality (the requester's own device, the same site, another site, or remote) and collects deny reasons: unavailable, model not ready, stale or future-dated observation, data class not permitted, cross-site or remote use not allowed by all three policy levels (organization, group and workspace), and, for non-local nodes, no mesh opt-in, no sharing consent, unverified or unencrypted transport, no ready site gateway, or relay denied. The effective data class is the stricter of the workspace's and the payload's, so a public payload cannot downgrade a restricted workspace. Eligible nodes are ordered device first, then same site, then other sites, then remote, then home site first, then by identifier, and the first is selected. Every response states that it is a preview: no execution, no authorization, no cloud fallback.

Three admission definitions exist in the THOX record, and they answer different questions (Table 5). The product planner chooses by locality and policy and does not consider capacity. The THOX reference admission kit checks tenant, data classification against the group's ceiling, node health and authentication, allowed and authenticated sites, and cross-site use only when policy enables it and the request consents; it then places the whole request on the eligible node with the most free memory, ties to the earlier node, and never compares demand with that node's capacity. The paper of record states a third rule:

required resources must be a subset of the group's approved set, demand above the group's pooled capacity is denied as unplaceable, demand above free capacity waits, and there is no partial placement, splitting or borrowing from another group [1]. Pooled capacity can exceed every single node, so the third rule admits work that only a multi-node execution could serve, which the first two rules never do and which THOX does not claim: no single request has been run across two sites. Section 7.3 measures how often the rules disagree.

Ruling. On 2026-11-04 the corresponding author ruled that the pooled rule of the paper of record governs (decision D1, Section 11). It is the rule implemented in the ThoxWork core service, enforced server-side at task creation and at assignment, and Section 7.7 measures that enforcement against the live API. The product planner and the reference kit are to align to it. Admission under the pooled rule is a statement about the group’s pooled capacity, not about any single node: a task admitted against the pool is admitted honestly, and if dispatch later cannot place it on one node, that shortfall is a dispatch outcome, never a retroactive admission failure.

Unit of decision	one routing preview	one request	one task
Who may ask	member with explicit actions; employee needs an accountable sponsor	tenant and group match	task's group
Data limits	stricter of workspace and payload class; node data classes	classification at or below group ceiling	not specified
Sites	cross-site and remote only if organization, group and workspace all allow	allowed, authenticated sites; cross-site needs policy and consent	not specified
Capacity	not considered	not compared with demand	demand against pooled group capacity; wait if not free
Selection	locality, then home site, then identifier	most free memory, then inventory order	admit or deny; placement unspecified
Effect	preview only	returns a node	admits to run

Table 5. The three MeshGroup admission definitions on record, shown side by side rather than merged. Ruled by the CTO, 2026-11-04: the pooled rule governs (Sections 7.7 and 11).

4.5 Handoffs and evidence

Two evidence mechanisms are built. ThoxEmployee's audit receipts record actor, action, target, rule and outcome before and after each effect, and its security documentation states plainly that the audit table is not a tamper-proof compliance ledger. Each handoff stores the SHA-256 of its body, and accepting a handoff whose body no longer matches is refused. Because the hash lives in the same writable row as the body, this detects accidental change, not an insider who rewrites both (adversary A3). The lab adds two proposed

mechanisms and measures them. A record r_i carries a hash that commits to its content and to the previous record, and, with a session key k held outside the store, an HMAC tag:

$$h_i = \mathrm{SHA256}\big(\mathrm{canon}(i,\, t_i,\, \kappa_i,\, a_i,\, \pi_i,\, h_{i-1})\big), \quad h_{-1} = 0^{256}, \qquad \tau_i = \mathrm{HMAC}_k(h_i) \tag{2}$$

where $t_i, \kappa_i, a_i, \pi_i$ are the time, kind, actor and payload and canon is JSON with sorted keys. An anchor is the pair (n, h_{n-1}) of record count and head hash, held by a second party (another MeshGroup member, the sponsor, or a separate store). Handoffs follow the same pattern: each handoff's chained hash commits to the task, the step, the member, the brief, the body hash and the previous handoff.

Proposition 1 (keyed chain). Let a verifier hold k. Any edit of a record's fields, any reorder, any deletion or insertion inside the chain, and any rebuild of the suffix after an edit, made by a party without k, is detected at the first affected position, except with probability bounded by forging HMAC-SHA256. Sketch. An edit at i changes h_i under second-preimage resistance; a rebuild that restores the links must produce τ_i for the new h_i without k.

Proposition 2 (truncation). No check that reads only the stored chain, keyed or not, detects removal of a suffix, because every prefix of a valid chain is itself a valid chain. Detection needs state outside the chain, such as an anchor (n, h_{n-1}) held by a second party. Sketch. Verification of $r_0 \ldots r_{m-1}$ uses only those records and k, and they are unchanged by deleting $r_m \ldots r_{n-1}$.

Section 7.4 confirms both propositions empirically and states the cost. The September 2026 trace-tampering result adds the deployment rule that follows from them: the workspace, not the agent, must write the record, and the anchor must sit outside the agent's reach [68]. Tree-structured logs would add compact proofs for audits of single events [59, 61]; the lab keeps a linear chain because per-team logs are small and verification is cheap at that size.

4.6 Memory that keeps contradictions

ThoxEmployee stores memory in seven tiers. For semantic facts it splits a statement into subject and value at the first colon, equals sign, arrow or form of the verb to be, normalizes case and spacing, and tags a new fact that collides with an earlier fact on the same subject. Both facts are kept and cross-referenced; nothing is merged or dropped (R8). The rule is lexical: a probe during review found that "the build is green" and "build: red" are not tagged because the subjects differ, a false negative stated here as a limit.

4.7 Security argument

R1 and R2 answer excessive agency and tool misuse structurally: an injected instruction (A1) can at most request an action, the gateway decides outside the model, and consequential effects wait for a person [47, 49]. Once the planner authorizes, R4 would bound the blast radius of A1 and A2 to one group, re-checked at each request, consistent with zero trust's rule that access to one

resource does not grant another [53]. The sponsor rule prevents an employee from exceeding its accountable person. Stale observations (A4) are rejected by the planner. A5 is not solved by any mechanism here; the record of what the approver saw and when is the mitigation the evidence chain supports, and reviewer capacity must be planned as a resource [1]. A3 and A6 are what the keyed, anchored chain addresses, with the limit of Proposition 2.

5 What is built: an evidence-bounded account

Three review teams read the repositories at the commits in Table 8 on 2026-11-04, started the products with local or demonstration engines only, and captured the screens and recordings used in this paper; a test harness then reran each repository's own suites, one at a time, in isolated copies, and two further teams built the planner comparison and the experiments of Section 7. Status labels follow the paper of record [1]: **Built** means code exists and ran here; **Preview** means built and deliberately non-executing; **Designed** means specified with partial code; **Not built** means absent. Test counts are read from the suite output by the harness of Appendix A.

ThoxWork shell (desktop, web, server)	Built, on an upstream base	Built on the open-source OpenWork project, running the OpenCode engine [72, 73]. THOX restyled it and added the layers below.	975 passed, 2 failed
MeshGroups	Preview	Strict schema, host-only preview endpoints, deterministic planner, settings view and end-to-end browser script. Execution, grants, persistence, enrollment and multi-node execution are not built in the shell. A standalone core service built after this review snapshot persists MeshGroups with their approved resources and enforces the pooled admission rule server-side (Section 7.7); multi-node execution remains not built and is not claimed.	within ThoxWork
Digital Employee gateway	Built	Server-held credentials, closed route list, owner, collaborator and viewer scopes, pinned contract release; the app never holds backend credentials.	within ThoxWork
ThoxEmployee	Built (pre-release)	Kernel with fail-closed gate, operator ceiling, preview digests, audit receipts, hashed handoffs, routines and contradiction tags; local review workspace; Migration Studio for bringing existing bots in. Clean-room from public behavior of OpenBot, OpenMausBot and Rakazo.	752 passed, 0 failed, 6 skipped
Digital Employee contract v2	Built	Assignment API with the seven-state lifecycle, approval records, work receipts and event streams.	510 passed, 0 failed, 28 skipped

Digital Humans framework (Rust)	Built	Gateway with approval inbox and role-checked decisions, runtime and memory crates.	337 passed, 0 failed
Digital Humans runtime (Python)	Built	Fleet runtime for the THOX roster, built on the open-source Hermes Agent [74].	24,109 passed, 72 failed, 160 skipped (review run, not rerun)
Rolodex directory	Built	Nine-card directory with recorded introductions, consent-gated chat with one employee at a time, and a team selection that is marked for review and grants no tools or permissions. No ordered task or handoff.	481 passed, 1 failed
ThoxEmployee guest demo	Built (hosted adapter not run here)	Ordered team task with written handoffs, revision-fenced steps, human approve or reject, daily limits and 24-hour expiry. Captured on the ThoxEmployee workspace engine in its offline demonstration mode.	engine within ThoxEmployee
RolodexGo	Built (pre-release)	Native client for iPhone, iPad and Mac with a React Native companion; pairs with a private server. Not captured: no macOS host was reachable.	292 passed, 0 failed
Validated dispatch	Built	Guardrails, planning, parallel workers, synthesis and a two-layer validator; unreadable verdicts withheld, budget exhaustion marked not validated.	257 passed, 0 failed
MeshGroup reference admission kit	Built (reference)	Reference admission policy with its own tests; not wired to the ThoxWork planner (Table 5).	8 passed, 0 failed

Table 6. Components reviewed for this paper, with status labels from the paper of record. Clean-room means rebuilt from public behavior without copying source.

6 The ThoxWork Evidence Lab

The Evidence Lab is a single HTML file that runs in any modern browser, opens offline, and also runs as the hosted demo on thox.ai, where it can keep a team log. It exists so that a buyer, auditor or reviewer can exercise the rules before any deployment exists, and so that a person running a product trial can capture evidence in one place. It never calls a model and never takes an external action; every employee output is deterministic demonstration text, and the example organization is fictional.

MeshGroup admission	The reference kit rule with per-node reasons, and the paper of record's pooled rule beside it on the same request	Agreement with the kit, Section 7.3
Routing planner	A clean re-implementation of the ThoxWork planner rules	Agreement with the product function, Section 7.2

Approvals	Algorithm 1 gateway, the Equation 1 approval gate with role check, and the Table 3 lifecycle	Rules read from the ThoxEmployee kernel and the v2 contract; unit tests
Ordered handoffs	A team runs a brief in order; body hashes as built and chained hashes as proposed; review to Done or changes requested	Unit tests; Proposition 1
Tamper tests	Five of the six E1 attacks on a copy of the live ledger, checked by hash links, by the session key and by an anchor	Experiment E1, Section 7.4
Captures	Screenshots, screen recordings and PDFs hashed in the browser and written into the ledger; a self-rendered snapshot of the ledger head	SHA-256 known-answer tests on load

Table 7. Evidence Lab benches. All logic is in four dependency-free modules shared with the experiment scripts, so the page and the measurements run the same code.

The ledger is a keyed chain (Equation 2). The session key is generated in the browser and never leaves the tab; an exported bundle carries hashes and tags but not the key, and an imported bundle can be checked for hash links by anyone. When the page runs in its hosted form it can save sessions to a shared team log, write a separate anchor record for each saved session, store captured files alongside the log and offer the bundle as a download; outside that host every feature degrades to the browser alone. The page follows the THOX design tokens (#05A451 on dark; on white, the templates' text green #047C3D, because #05A451 on white falls below WCAG AA contrast for body text; Inter and JetBrains Mono), works at 390 pixels wide without horizontal scrolling in both themes, and runs SHA-256 and HMAC known-answer tests on load (FIPS 180-4 examples and RFC 4231 cases).

7 Evaluation

Setup. All experiments ran on one host: 2 vCPU Intel(R) Xeon(R) Processor @ 2.10GHz, 8 GB RAM, Linux, Node v22.23.4. Each timed configuration had one warm-up run and 30 measured repetitions; figures report medians, with interquartile ranges where shown. Every random input comes from a seeded generator (master seed 20261104). Experiment scripts write raw JSON; one analysis script reduces it to summary.json, from which every number in this section, the figures and the lab's measured panel are generated. The questions are: does the code pass its own tests (7.1); does the lab reproduce the product planner (7.2); how do the admission definitions differ (7.3); what does an evidence chain detect and at what cost (7.4); and what do decisions cost (7.5). Section 7.6 reports the product evidence captured, and Section 7.7 replays the protocol experiments of the paper of record [1] against the live ThoxWork core service, a separate implementation measured over its HTTP API; those results are reported apart from the lab results and are labelled measured.

7.1 Do the components pass their own tests?

MeshGroup reference admission kit	657c7c4	1 / 0 / 0	8	0	0	MeshGroup reference admission kit only
ThoxWork shell and servers	ab29b61	18 / 0 / 2	975	2	0	2 failures in one server suite, tracked internally; includes tests inherited from OpenWork; hosted enterprise and live-provider suites not run
ThoxEmployee runtime	efe63da	4 / 0 / 2	752	0	6	Swift kit and Docker kernel suites not run (no macOS host, no container runtime)
Digital Employee contract service	8997830	2 / 0 / 1	510	0	28	PostgreSQL integration suite not run (no disposable database); 28 skipped for the same reason
Digital Humans gateway (Rust)	73088ef	3 / 0 / 0	337	0	0	Rust workspace tests; clippy with warnings as errors and rustfmt also pass
Digital Humans agent runtime (Python)	c7b4c33	0 / 1 / 0	24,109	72	160 (+52 errors)	Parsed from the review team's run, not rerun; suite largely inherited from Hermes Agent; failures and errors concentrated in browser-automation, SSH, chat-platform and terminal tests that need components the review host lacked
Rolodex directory	eeef7de	3 / 0 / 1	481	1	0	1 failure, tracked internally; browser suite replaced by captures
Thoxy dispatch	2ee63c6	2 / 0 / 0	257	0	0	Backend and web suites
thox.ai website	8999c14	1 / 0 / 1	135	0	0	Claim-related subset only; hosted guest-demo check needs service credentials and was not run
RolodexGo	cdee70c	1 / 0 / 0	292	0	0	TypeScript and domain suites; Swift targets not compiled

Table 8. Product test suites on 2 vCPU Intel(R) Xeon(R) Processor @ 2.10GHz, 8 GB RAM, Linux, Node v22.23.4. 'Run' suites were executed for this table; 'log' parsed the review team's run of the same commit; subset and repeat runs are excluded from the counts. Counts are read from each tool's own summary output. Browser end-to-end assertion groups (17 passed) and 10 scripted lint, type, format and project checks (all passed) are reported separately; a contract check that does not yet gate on its

output is not counted. The run column counts suites and checks that ran for this table; the Passed column counts tests from rerun suites only, except the Python agent runtime, whose counts are parsed from the log.

The 18 suites rerun for this paper passed 3,747 tests and failed 3, with 34 skipped. Two failures are in one ThoxWork server suite and the third is in the Rolodex directory; all three are tracked internally. The Python runtime's suite, parsed from the review team's run rather than rerun, adds 24,109 passing tests, 72 failures and 52 errors; it is largely the Hermes Agent suite the runtime inherits, so it says more about the upstream than about THOX's additions, and it is reported, not counted, in the headline. Seven suites did not run here because they need a macOS host, a container runtime, a disposable database, hosted credentials or a live model provider (Table 8). The Evidence Lab's own 75 unit tests pass and are excluded from the product totals.

7.2 Does the lab reproduce the ThoxWork planner?

Experiment E3 generated 250 seeded MeshGroup directories (1 to 100 nodes and up to 6 groups each) and 12,000 routing requests, mixing allowed, denied and unservable cases. Every input passed the product's own schema. The product planner, imported from the ThoxWork source at the reviewed commit, and the lab's clean re-implementation then answered every request. They agreed on status, reason, selected node and every candidate's set of deny reasons for 12,000 of 12,000 requests, and on every compared field, including candidate order, locality, effective data class and the order of deny reasons, for 12,000; 55,832 of 55,832 per-node verdicts matched. The outcome mix was 3,038 eligible, 5,127 with no eligible node and 3,835 denied before candidates were listed, covering all 9 early denials and all 11 per-node deny reasons. On 4,000 deliberately altered inputs of 18 kinds, the product schema and the lab validator gave the same accept or reject verdict 4,000 times. Agreement shows that the lab runs the product's rules as implemented; as limitation 3 notes, it does not show the rules are right.

Planning is cheap; validation is not free. With 100 bound nodes, a full product preview took a median 783 microseconds and parsing the request schema alone 608, so schema validation is about 78 percent of the call (at these sizes the ratio moves by a few points between runs); the lab's plan step alone took 83 microseconds. Validating a full 100-group directory took 42.3 ms with the product schema and 6.7 ms with the lab's validator. These are medians of 30 interleaved repetitions on the host of Section 7 while it was otherwise idle (one-minute load average near 1).

7.3 How do the admission definitions differ?

Agreement with the reference kit. The lab's reference-rule module and THOX's reference admission kit decided 20,000 seeded requests (200 scenarios of 50 nodes and 100 requests, with ties in free memory made common on purpose). They agreed on 20,000 of 20,000, including every tie.

The outcome mix was 4,570 placed, 4,377 admitted with no eligible node, and 11,053 denied before placement (Figure 2).

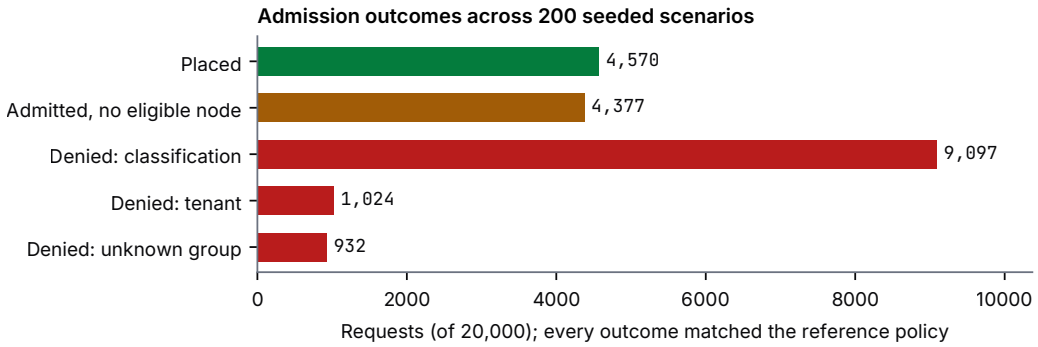


Figure 2. Outcomes of 20,000 seeded admission requests under the reference rule; the lab module and the THOX reference kit returned the same outcome for every request.

Divergence between the definitions. The same 20,000 requests were given a seeded demand, uniform between 1 GB and 1.5 times the largest free memory in each inventory, and judged by both the reference rule and the paper of record's pooled rule (request-level denials applied first, so the comparison isolates placement). Figure 3 shows the result. Of the reference rule's 4,570 placements, 2,572 (56.3%) land on a node with less free memory than the request, because the reference rule never compares the two. The pooled rule admits 1,655 requests only by counting memory across several nodes, which no single node could serve, and it admits 2,678 requests that the reference rule leaves with no eligible node; for all 2,678 of them the reference rule would place the request if cross-site use were enabled and consented, so the gap is the pooled rule's missing notion of sites: cross-site default-deny never applies. The numbers depend on the demand model and are illustrative; the direction does not: each definition admits work the others refuse.

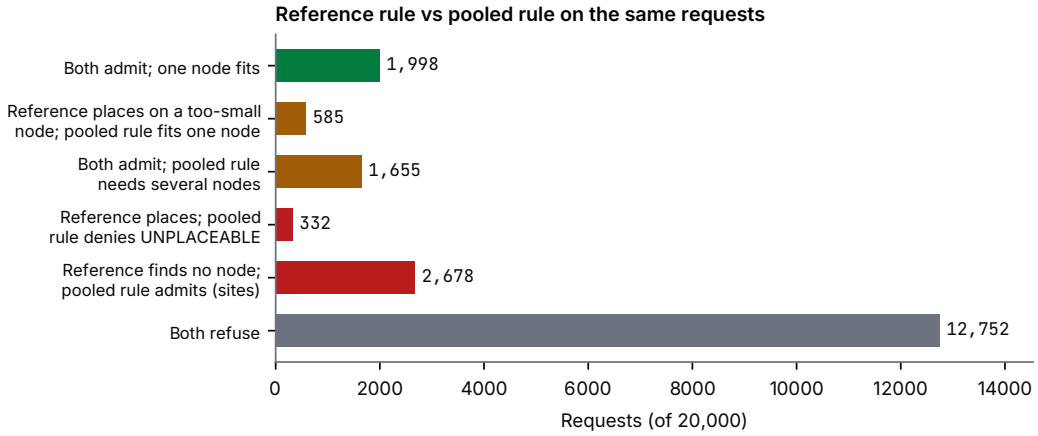


Figure 3. How the reference rule and the paper of record's pooled rule judge the same 20,000 requests with seeded demand. Illustrative demand model stated in the text.

7.4 What does an evidence chain detect, and at what cost?

Six attacks were applied to copies of 1,000-record chains in 500 seeded trials each and checked four ways: hash links alone, with the session HMAC key, with

an external anchor, and with both (Figure 4). Edits to a decision or an actor, swaps and deletions inside the chain were detected in every trial by every check, and in-place edits were located at the exact record 100% of the time. Cutting the tail was missed by every in-band check, keyed or not, in every trial, and caught by the anchor in every trial, as Proposition 2 predicts. Editing a record and rebuilding every later hash was missed by hash links alone in every trial and caught by the key or the anchor in every trial, as Proposition 1 predicts. The built mechanisms correspond to the first column: they detect accident, not an insider.

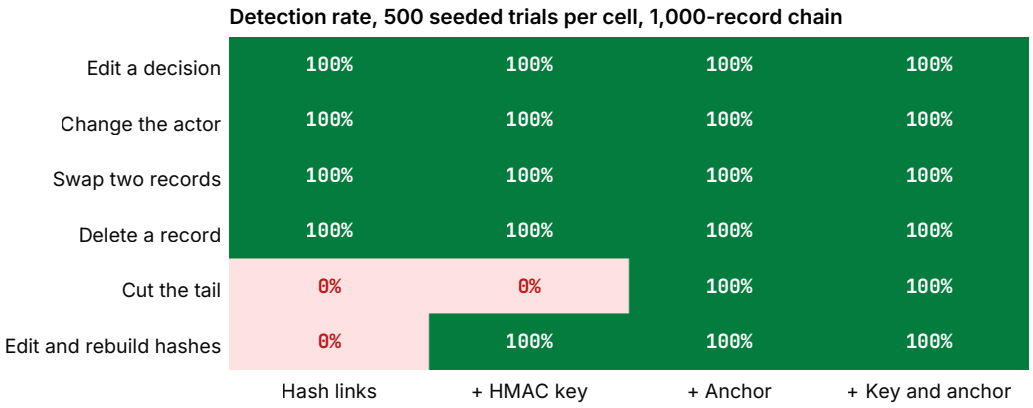


Figure 4. Detection rate by attack and check. Red cells are attacks a check never caught; green cells are attacks it always caught.

The cost is small at team scale (Figure 5). On this host a keyed append took a median 27.6 microseconds per record and keyed verification 31.6 microseconds per record on 1,000-record chains; an unkeyed append took 14.2. Verifying a 10,000-record keyed chain took a median 307 ms. An exported bundle costs about 380 bytes per record. These figures are for a pure JavaScript SHA-256 written for the lab so that it runs anywhere; a native or WebCrypto implementation would be faster. For comparison only, Auditable Agents reports 8.3 ms median overhead for its full pre-execution mediation on its own hardware [67]; the two numbers measure different work.

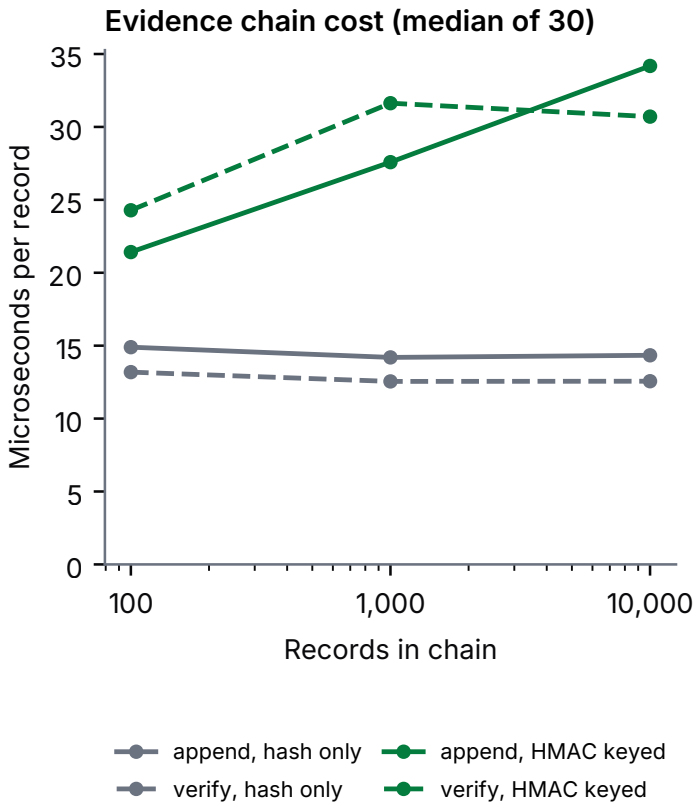


Figure 5. Evidence chain cost per record by chain length, hash-only and HMAC-keyed, append and verify (medians of 30 repetitions).

7.5 What do decisions cost?

Admission decisions are cheap at the scale of a team (Figure 6). The lab module took a median 0.11, 1.05 and 13.3 microseconds per decision for inventories of 10, 100 and 1,000 nodes, and the THOX reference kit in Python 1.57, 6.3 and 50.1 microseconds. Cost grows with inventory size because every node is checked. Batches include early denials, which cost almost nothing, in the proportions of Figure 2.

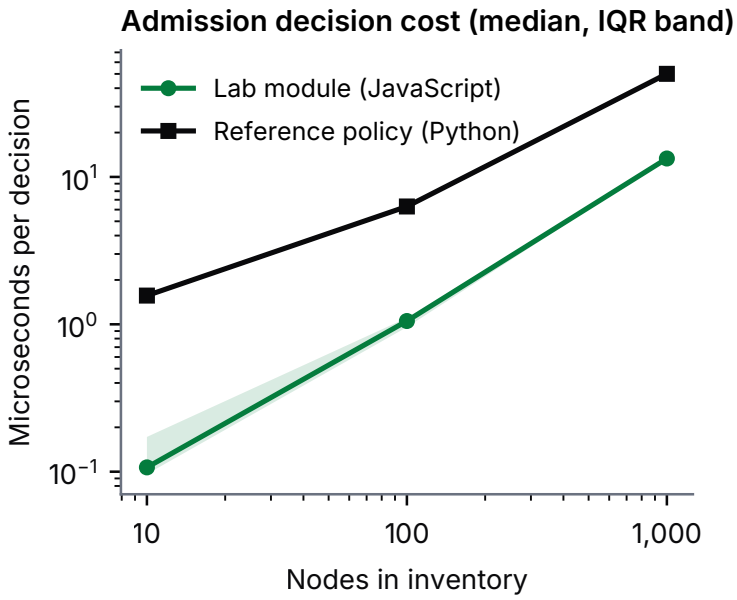
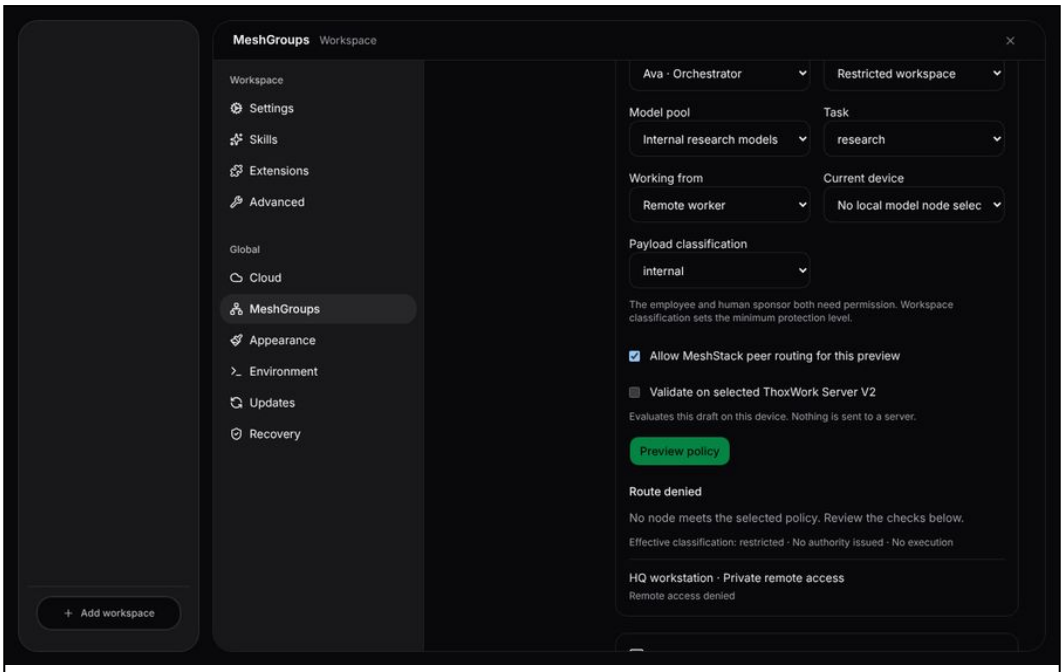


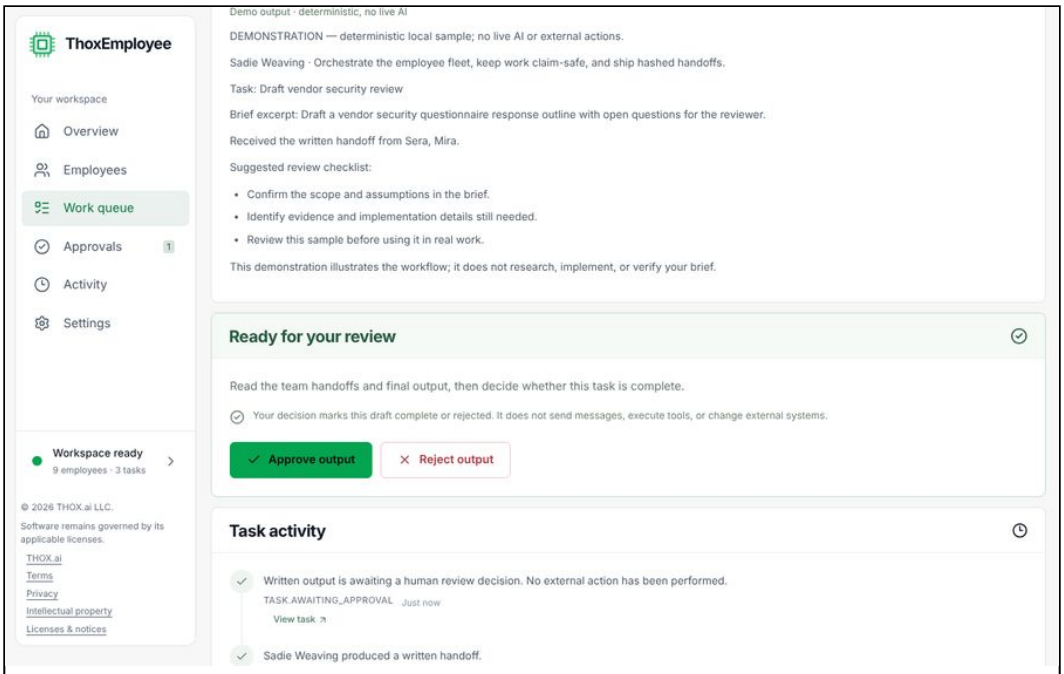
Figure 6. Admission decision cost against inventory size for the lab module and the THOX reference kit (median and interquartile range of 30 repetitions of 1,000-request batches).

7.6 Product evidence captured

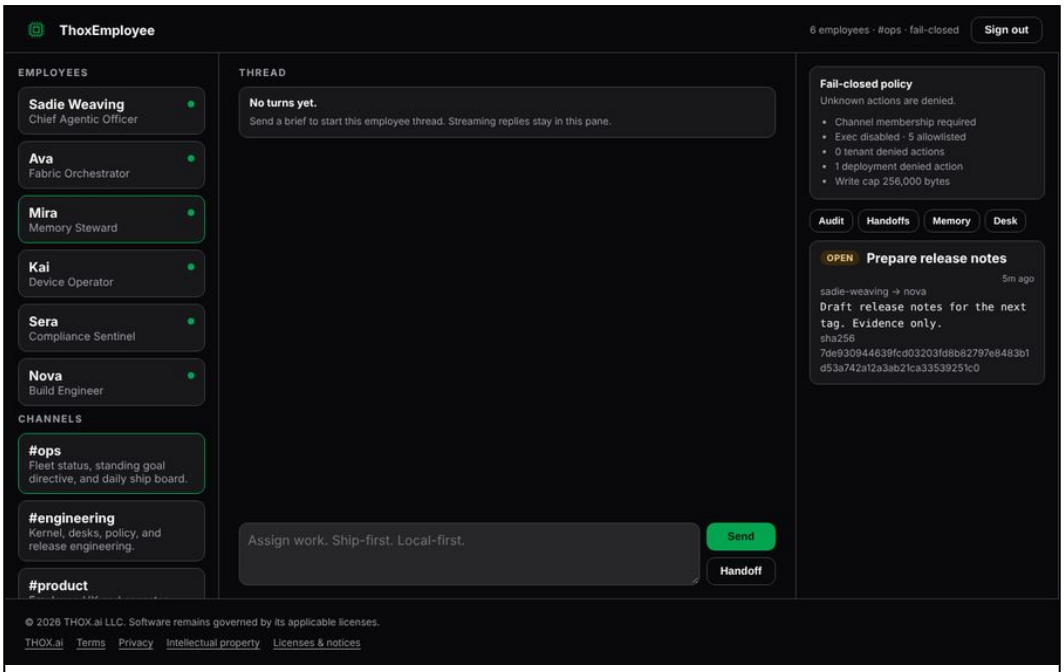
The review teams captured the real running products, never mockups, at 1280 by 800 and 390 by 844 pixels, and recorded screen videos of complete flows. Figure 7 shows four of them. A subset reviewed against the disclosure boundary of Appendix D ships in the reproducibility package (Appendix A) with SHA-256 hashes; captures that show local ports, device registries or generated identifiers are withheld.



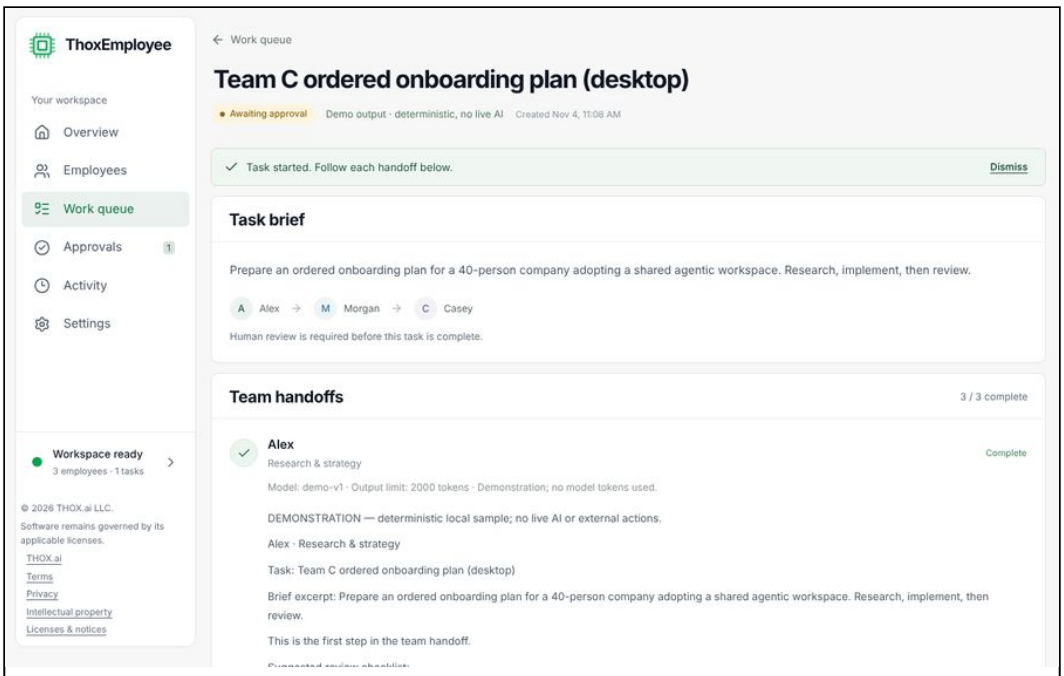
ThoxWork MeshGroups: routing preview denied for a restricted workspace



ThoxEmployee workspace: output waits for a person's review



ThoxEmployee kernel: fail-closed policy and a hashed handoff



ThoxEmployee guest-demo engine: ordered team handoffs before review

Figure 7. Screens from the real running products on 2026-11-04, local and demonstration engines only. The demonstration outputs are deterministic and labelled as such by the products themselves.

7.7 The live core service: the protocol experiments, measured

The paper of record [1] evaluated its five invariants in a deterministic, seeded simulator, with every result labelled simulated. The ThoxWork core service is the platform's own server implementation of the same model: a Rust service with a PostgreSQL system of record, MeshGroups persisted with their approved resources, pooled capacity computed inside the admission transaction and never stored, a server-enforced Rolodex state machine whose reassignment requires a written handoff, DONE reachable only through review approval, and

one append-only, hash-chained event log that the database roles may insert into but never update or delete. For this edition, the simulator's experiment generators were ported one-to-one (same seed family, master seed 20261104, 20 replicates per experiment) and driven against the live service over its HTTP API. Three judges then recomputed every invariant independently: the platform's separate verifier service, through its API and through its command line, and a harness-side checker written against the protocol, with both hash chains recomputed.

E1 Policy enforcement	12,000 violating attempts: 0 leaked in every replicate. 8,000 legitimate attempts: 0 blocked (false-block rate 0.0, bar ≤ 0.01). Audit coverage 1.0 in every replicate.	PASS
E2 Handoff integrity	5,000 reassigned tasks: chain completeness 1.0; 15,000 entries: note coverage 1.0. 4,000 negative probes (empty or whitespace notes, non-member assignee, reassignment from review): 0 leaks. Maximum chain depth 5.	PASS
E3 Admission honesty	4,000 oversized tasks: all denied UNPLACEABLE, synchronously in the create response; 0 silently placed, 0 cross-group spills, 0 stranded. 6,000 right-sized tasks: all placed. Boundary held in every replicate: a 120-resource-unit demand admitted, 121 denied.	PASS
E4 Review gate	6,000 bypass attempts across six classes (direct finish, self-review, wrong role, non-member reviewer, stale approval, skipped review): 0 successes. 4,000 of 4,000 legitimate completions; 0 done without review over 8,000 DONE tasks; rejection integrity 1.0 over 8,000 checks. Control arm with review disabled: 4,000 of 4,000 completed without review, so the measurement discriminates.	PASS
Parallel admission	50 assignments in parallel against a pooled capacity of 120 resource units at 12 units each: exactly 10 ACCEPTED, 40 WAIT, 0 denied, 0 rejected; admitted demand exactly 120 units and stable. Zero over-admission.	PASS

Table 11. The protocol experiments of the paper of record, replayed against the live ThoxWork core service. Measured. The live run supersedes the simulator as evidence for E1 to E4; E5 was not run live and remains simulated (text below).

Judges and drills. The three judges agreed on all 102 judged workspaces: every invariant true, both hash chains clean, zero disagreements. Fail-closed drills passed: with the event log's insert permission revoked on a scratch instance, assignment returned a 503 operational error and the task was unchanged; with the database severed, assignment returned 503 and nothing was admitted. The suspension and revocation drill found the one defect of the run: suspend and revoke first returned 404 for a principal present in more than one workspace, because the handler resolved the workspace from the bare identifier (no mutation occurred; the security property itself held). The fix scopes those routes by workspace and adds a token epoch, checked inside the transition transaction, that suspend, revoke and rotation increment so outstanding credentials die immediately; the drill then passed in full, and a post-fix verifier run over 121 workspaces was clean.

What this does and does not show. E3 measures admission under the pooled rule of Section 4.4, the rule the ruling in Section 11 adopts: capacity is the group's pool, computed at decision time. Live dispatch is not part of this milestone: the core's dispatch leg reports itself unconfigured rather than simulating success, and no multi-node execution of a single request is claimed anywhere in this paper. The fifth invariant's scale grid (E5 in [1]) was not run against the live service; it remains a simulated result whose falsifiable content is non-collapse (completion 1.0 in all 16 cells), not the efficiency figure, which exceeds 1.0 only as an artifact of the simulated-latency model. Throughput posture is stated plainly in the implementation record: writes serialize through one database connection behind a lock, which is what makes the sequential semantics exact under concurrency; it is a pilot posture, not a scale posture.

8 Governance mapping

Table 9 maps the mechanisms of ThoxWork and of the Digital Employee stack beneath it (ThoxEmployee, the Digital Employee contract and the Digital Humans gateway) to the oversight, record-keeping and access requirements of Section 2.3. It states alignment with the text of each document, not conformance; no audit has been performed, the NIST framework is voluntary, and whether the EU AI Act binds a deployment depends on its use.

Define roles for human-AI configurations and oversight	NIST AI 100-1 GOVERN 3.2 [42]	Named Digital Employees with roles and accountable human sponsors; each approval request names its required role	Built in the Digital Employee stack
Document oversight processes	NIST AI 100-1 MAP 3.5 [42]	Approval reasons, required role and decision recorded per request	Built in the Digital Employee stack
Content provenance	NIST AI 600-1 [43]	Hashed handoffs and receipts as built; keyed, anchored chain as proposed	Built in part (ThoxEmployee); proposed
Automatic recording of events	EU AI Act Art. 12 [45]	Audit receipts around each gated effect; lifecycle events per assignment	Built in ThoxEmployee (not tamper-evident)
Override, reverse or stop	EU AI Act Art. 14 [45]	Person-only approval; reject blocks; cancel is terminal; previews never execute	Built in the Digital Employee stack
Keep logs at least six months	EU AI Act Art. 26(6) [45]	Local-first records owned by the organization; retention policy is the deployer's	Deployer duty
Management system for AI	ISO/IEC 42001 [44]	Evidence the system can supply to an organization's management system	Supporting evidence only

Excessive agency, complete mediation	OWASP LLM06 [47]	Fail-closed gateway outside the model; operator ceiling; approval for effects	Built in ThoxEmployee
Identity and privilege abuse; task-scoped access	OWASP Agentic Top 10 [49]	MeshGroup actions per member; sponsor rule; group bindings	Preview
Least privilege per session; no transitive access	NIST SP 800-207 [53]	Group-scoped planning; cross-site and remote only if all three policy levels allow	Preview

Table 9. Mechanisms against requirements. 'Built' names where the mechanism runs today; its integration into the ThoxWork shell is in development. 'Preview' means the ThoxWork rule is implemented and tested but the planner does not yet execute or authorize.

9 Discussion and limitations

1. **One host, no fleet.** Every timing in Section 7 comes from one cloud host with the stated CPU and runtime. No THOX device, no customer deployment and no multi-node run was measured, and no figure is a device benchmark.
2. **Preview is not enforcement.** The MeshGroup planner decides but does not authorize or execute. Its agreement with the lab shows that the rules are implemented as documented, not that a deployment enforces them. The planner in the ThoxWork shell remains preview-only in exactly this sense; the standalone core service, however, enforces its rules server-side, and Section 7.7 measures that enforcement against the live API.
3. **Agreement is not correctness.** The lab planner was written from the same documented rules the product implements; a rule that is wrong in both would agree. The reference-kit comparison is against a second, independent implementation, which is stronger.
4. **Illustrative demand.** Section 7.3's divergence counts depend on a uniform demand model chosen for illustration; the qualitative result (each definition admits work the others refuse) does not.
5. **Enumerated attacks only.** Section 7.4 tests six attack classes. Propositions 1 and 2 bound what any in-band check can do; they do not cover a compromised verifier or a stolen session key.
6. **Partial test coverage.** Swift, Docker-based and PostgreSQL-backed suites did not run here (Table 8), and the Python runtime's suite was parsed from a review run rather than rerun. The native ThoxWork, RolodexGo and MeshStack apps for iPhone, iPad and Mac were not built or captured because no macOS host was reachable from the review environment; that capture is scheduled for the next edition.
7. **Abstract-based positioning.** Table 1 judges other systems from their abstracts; full texts could show more than the abstracts state.

8. **Interested party.** The authors are THOX principals and the code reviewed is THOX's own; the review teams worked from the code and reported defects, which are tracked internally and not detailed here.

10 Conclusion

ThoxWork's case does not rest on a new model or a new agent framework. It rests on rules that sit outside the model and on records that a person can check. The code reviewed for this paper implements most of those rules today: a gateway that fails closed, an approval gate that only a person can pass, a lifecycle that refuses illegal moves, a MeshGroup planner that scopes every request to a team and its sponsor, handoffs that carry their own hashes, and memory that keeps contradictions visible. The measurements add two corrections that the record needs: the admission rule is now one rule, the pooled rule, implemented in the core service and measured in Section 7.7; and the evidence must be keyed and anchored to resist an insider. The Evidence Lab lets anyone exercise those rules before trusting them.

11 Roadmap and founder decisions

One item has been ruled on and implemented since the review (D1). The rest are stated with the option this paper recommends; none of the rest is applied by this paper.

D1	Which MeshGroup admission definition governs (Table 5)	**Ruled, 2026-11-04.** The pooled rule of the paper of record governs: pooled capacity computed over the group’s approved resources, work above the pool denied UNPLACEABLE, work above free capacity waiting, no splitting and no borrowing. It is implemented in the ThoxWork core service and measured in Section 7.7; the planner’s scoping and locality ordering and the kit’s classification ceiling carry into placement under it. Admission is a pooled statement; a later dispatch shortfall is a dispatch outcome, not an admission failure.
D2	Upstream credit for the ThoxWork shell	Keep the upstream license notices beside THOX's and credit OpenWork and OpenCode in every distribution (applied in the ThoxWork repository on 2026-11-04); counsel reviews third-party license terms before any release
D3	Evidence chain	Adopt the keyed, anchored chain of Equation 2 for receipts and handoffs, written by the workspace, anchored with a second party
D4	Assignment vocabulary	Publish Table 4 as the mapping; align the gateway's rejection to blocked, as the contract states
D5	Customer-facing roster	Pick one roster and one name for the workspace assistant across products and the website
D6	Reviewer capacity	Plan reviewers per MeshGroup as a resource, as the paper of record recommends [1]

Table 10. Founder decisions.

The roadmap that follows from the evidence: align the shell planner and the reference kit to the pooled rule the core service already enforces, and put live dispatch behind the core's admission, with the keyed record written at each decision; carry the sponsor rule into the Digital Employee gateway; finish the native client work: land the RolodexGo ordered-task changeset described in Section 4.3 and complete the macOS-host client captures begun after this review; and measure approval-gate latency and approval-record completeness with people in the loop, which no verified study has done for multi-agent enterprise work. ThoxWork remains Coming Soon in THOX's product copy, with no price or date stated here.

A Reproducibility

Host: 2 vCPU Intel(R) Xeon(R) Processor @ 2.10GHz, 8 GB RAM, Linux, Node v22.23.4. Python 3.13.18; Node v22.23.4; matplotlib, Pillow and pypdf for figures and PDF; KaTeX, @fontsource/inter and @fontsource/jetbrains-mono for rendering; Playwright with Chrome for PDF and QA. The THOX reference admission kit and the ThoxWork planner are THOX source and are not redistributed; experiments that compare against them (E2, E3) record the THOX outcomes in their result files so the comparison can be re-checked without the source.

```
node --test lab/test/*.test.js               # lab unit tests
node exp/evidence_experiments.js             # E1
python3 exp/meshgroup_reference.py ...       # E2 reference side (THOX internal)
node exp/meshgroup_lab.js                    # E2 lab side, agreement
tsx exp/planner_agreement.ts                 # E3 (THOX internal)
tsx exp/planner_timing.ts                    # E4 (THOX internal)
node exp/admission_divergence.js             # E5
python3 exp/product_tests.py                 # product tests (THOX internal)
python3 exp/analyze.py                       # writes summary.json
python3 wp/figures.py && python3 wp/build.py
node wp/katex_render.cjs wp/paper.src.html wp/paper.html
node wp/pdf.cjs wp/paper.html wp/cover.pdf wp/body.pdf
python3 wp/merge_pdf.py && python3 wp/build_lab.py
```

Evidence Lab known-answer tests: 7 of 7 pass (FIPS 180-4 SHA-256 examples, RFC 4231 HMAC cases 1, 2 and 6). Seeds: master seed 20261104 for E1 to E4 (E3 and E4 derive one child seed per directory or size from it) and 20261109 for the E5 demand draw. The package also contains the reviewed subset of screenshots and recordings with their SHA-256 hashes.

B Notation

g_a, h_r, σ_a	external-effect gate of assignment a, human-approval flag of responsibility r, sensitivity of a
$\alpha_{e(a)}, \alpha_{\rho(a)}$	autonomy level of the employee and of the role on assignment a

h_i, τ_i, k	hash and HMAC tag of record i, session key
$t_i, \kappa_i, a_i, \pi_i$	time, kind, actor and payload of record i
(n, h_{n-1})	anchor: record count and head hash held by a second party

D Document control, authorship and publication boundary

This preprint is issued by THOX.ai Research. The named authors are Craig Ross (Chief Executive Officer) and Tommy Xaypanya (CTO and Co-Founder, corresponding and accountable author), THOX.ai LLC, matching the roster of THOX's prior research papers and THOX's shared publication author profile. Conflicts: the authors are THOX principals; this is an interested-party technical account.

Document control. Comprehensive edition v1.1.0, 2026-11-04. Version DOI 10.5281/zenodo.23965052; concept DOI 10.5281/zenodo.23963935. This edition supersedes v1.0.1 (version DOI 10.5281/zenodo.23963936) as the current edition of the ThoxWork paper; v1.0.0 was the internal edition of the paper of record [1]. Changes in this edition: the code audit, the reconciled model, the Evidence Lab and the measurements of Sections 5 to 7 are incorporated; the errata recorded against v1.0.0 (reference titles and author lists, the Ju and Aral figures, the SailPoint and Work Trend Index readings, the orchestration finding, the stewardship of MCP and A2A, and the OpenWork and OpenCode lineage) are reconciled into the text and references of this edition rather than carried as a register; the RolodexGo account is corrected in Section 4.3 (the shipped client is the contacts client with Sadie Link; its Teams and Work changeset is built and test-covered but has not landed); decision D1 is recorded as ruled, with the pooled admission rule implemented in the core service; and Section 7.7 adds the live-service replay, measured, which no earlier edition contained. The authors' shared publication profile and consent cover this edition. Every number in this edition is generated from the experiment outputs in the accompanying reproducibility package, reduced to one summary file.

Disclosure boundary. The paper states rules and results. It omits internal hostnames, ports, project and database identifiers, credentials, per-role autonomy settings and approver mappings, persona instructions, model-to-base mappings beyond what THOX has published, prices, dates, customer and partner identities, and security defects that are not yet fixed, which were reported to THOX internally. Lineage and license facts are stated because the licenses require attribution.

References

[1] Craig Ross, Tommy Xaypanya. ThoxWork: One Workplace, One Connected Team. A Policy-Bounded DigitalHuman Work Platform for Enterprise Use. THOX.ai Research technical report

v1.0.1, preprint, 2026-11-04; Zenodo DOI 10.5281/zenodo.23963936. https://doi.org/10.5281/zenodo.23963936

[2] Craig Ross, Tommy Xaypanya. THOX MeshStack: Capacity-Priced Pipeline Sharding, Grant-Gated KV Transfer and Lease-Fenced Device Roles for a Private Household Mesh. THOX.ai Research technical report v1.0.1, preprint, 2026-11-03; Zenodo DOI 10.5281/zenodo.23919580. https://doi.org/10.5281/zenodo.23919580

[3] Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, Graham Neubig. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. NeurIPS 2025 Datasets and Benchmarks Track (poster); arXiv:2412.14161. https://arxiv.org/abs/2412.14161

[4] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. NeurIPS 2024 Datasets and Benchmarks Track (poster); arXiv:2404.07972. https://arxiv.org/abs/2404.07972

[5] Shunyu Yao, Noah Shinn, Pedram Razavi, Karthik Narasimhan. $τ$-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. ICLR 2025 (poster); arXiv:2406.12045. https://arxiv.org/abs/2406.12045

[6] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, Ion Stoica. Why Do Multi-Agent LLM Systems Fail?. NeurIPS 2025 Datasets and Benchmarks Track (spotlight); arXiv:2503.13657. https://arxiv.org/abs/2503.13657

[7] Alex Singla, Alexander Sukharevsky, Bryce Hall, Lareina Yee, Michael Chui, Tara Balakrishnan. The state of AI in 2025: Agents, innovation, and transformation. McKinsey & Company (QuantumBlack, AI by McKinsey), November 5, 2025. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-2025

[8] SailPoint, Dimensional Research. SailPoint research highlights rapid AI agent adoption, driving urgent need for evolved security. SailPoint press release, Austin, TX, May 28, 2025; report 'AI agents: The new attack surface' (survey by Dimensional Research). https://www.sailpoint.com/press-releases/sailpoint-ai-agent-adoption-report

[9] Microsoft WorkLab. Agents, human agency, and the opportunity for every organization (2026 Work Trend Index Annual Report). Microsoft WorkLab, May 5, 2026; survey by Edelman Data x Intelligence. https://www.microsoft.com/en-us/worklab/work-trend-index/agents-human-agency-and-the-opportunity-for-every-organization

[10] Peter Weill, Stephanie L. Woerner. Leveraging Digital Colleagues for Enterprise Value. MIT CISR Research Briefing No. XXVI-4, April 16, 2026. https://cisr.mit.edu/publication/2026_0401_DigitalColleagues_WeillWoerner

[11] National Institute of Standards and Technology, Center for AI Standards and Innovation (CAISI). AI Agent Standards Initiative. NIST initiative page; announced by CAISI on February 17, 2026; page last updated August 14, 2026. https://www.nist.gov/artificial-intelligence/ai-agent-standards-initiative

[12] Harold Booth, William Fisher, Ryan Galluzzo, Joshua Roberts. Accelerating the Adoption of Software and AI Agent Identity and Authorization. NIST National Cybersecurity Center of Excellence, draft concept paper (Initial Public Draft), February 5, 2026; comments closed April 2, 2026. https://csrc.nist.gov/pubs/other/2026/02/05/accelerating-the-adoption-of-software-and-ai-agent/ipd

[13] Josh Woodruff. The Agentic Trust Framework: Zero Trust Governance for AI Agents. Cloud Security Alliance blog, February 2, 2026. https://cloudsecurityalliance.org/blog/2026/02/02/the-agentic-trust-framework-zero-trust-governance-for-ai-agents

[14] Josh Woodruff. Agentic Trust Framework Specification. Version 0.9.1, Public Review Draft, last updated April 2026; GitHub massivescale-ai/agentic-trust-framework; stewardship transferring

to CSAI Foundation (CSA press release, April 29, 2026). https://github.com/massivescale-ai/agentic-trust-framework/blob/main/SPECIFICATION.md

[15] Anthropic. Introducing the Model Context Protocol. Anthropic news, November 25, 2024. https://www.anthropic.com/news/model-context-protocol

[16] Rao Surapaneni, Miku Jha, Michael Vakoc, Todd Segal. Announcing the Agent2Agent Protocol (A2A). Google Developers Blog, April 9, 2025. https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/

[17] Model Context Protocol a Series of LF Projects, LLC. Model Context Protocol Specification, revision 2026-07-28. modelcontextprotocol.io; current protocol version 2026-07-28. https://modelcontextprotocol.io/specification/2026-07-28

[18] A2A Protocol Project (Linux Foundation). Agent2Agent (A2A) Protocol Specification. a2a-protocol.org, Latest Released Version 1.0.0; project under the Linux Foundation-directed Agentic AI Foundation (blog, August 27, 2026). https://a2a-protocol.org/latest/specification/

[19] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, Bernard Ghanem. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. NeurIPS 2023; arXiv:2303.17760. https://arxiv.org/abs/2303.17760

[20] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, Jürgen Schmidhuber. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. ICLR 2024 (oral, per OpenReview); arXiv:2308.00352. https://arxiv.org/abs/2308.00352

[21] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, Maosong Sun. ChatDev: Communicative Agents for Software Development. ACL 2024 (Long Papers), Anthology ID 2024.acl-long.810; arXiv:2307.07924. https://arxiv.org/abs/2307.07924

[22] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. COLM 2024; arXiv:2308.08155. https://arxiv.org/abs/2308.08155

[23] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, Jie Zhou. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. ICLR 2024 (poster); arXiv:2308.10848. https://arxiv.org/abs/2308.10848

[24] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. Proceedings of UIST 2023, ACM; arXiv:2304.03442. https://arxiv.org/abs/2304.03442

[25] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, Ji-Rong Wen. A Survey on Large Language Model based Autonomous Agents. Frontiers of Computer Science 18(6): 186345, 2024; DOI 10.1007/s11704-024-40231-1; arXiv:2308.11432. https://arxiv.org/abs/2308.11432

[26] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Tao Gui. The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv preprint arXiv:2309.07864, 2023. https://arxiv.org/abs/2309.07864

[27] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, Xiangliang Zhang. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. IJCAI 2024 Survey Track, pp. 8048-8057, 2024; DOI 10.24963/ijcai.2024/890; arXiv:2402.01680. https://arxiv.org/abs/2402.01680

[28] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, Hoang D. Nguyen. Multi-Agent Collaboration Mechanisms: A Survey of LLMs. arXiv preprint arXiv:2501.06322, 2025. https://arxiv.org/abs/2501.06322

[29] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang, Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, Saleema Amershi. Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks. arXiv preprint arXiv:2411.04468, 2024. https://arxiv.org/abs/2411.04468

[30] Harsh Rao Dhanyamraju, Leonidas Raghav, Aaron Lee. Autonomous Event-Driven Multi-Agent Orchestration for Enterprise AI at Scale. arXiv preprint arXiv:2606.20058, 2026 (v2, 28 Sep 2026). https://arxiv.org/abs/2606.20058

[31] Ruhao Liu, Qi Li, Shizun Wang, Xinchao Wang. DynaFlow: Learning Free-Form Multi-Agent Orchestration. arXiv preprint arXiv:2610.34467, 2026 (v2, 29 Oct 2026). https://arxiv.org/abs/2610.34467

[32] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, Alexandre Lacoste. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks?. ICML 2024 (poster); arXiv:2403.07718. https://arxiv.org/abs/2403.07718

[33] Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, Chien-Sheng Wu. CRMArena: Understanding the Capacity of LLM Agents to Perform Professional CRM Tasks in Realistic Environments. NAACL 2025; arXiv:2411.02305. https://arxiv.org/abs/2411.02305

[34] Abubakarr Jaye, Nigel Boachie Kumankumah, Chidera Biringa, Anjel Shaileshbhai Patel, Sulaiman Vesal, Dayquan Julienne, Charlotte Siska, Manuel Raúl Meléndez Luján, Anthony Twum-Barimah, Mauricio Velazco, Tianwei Chen. CORPGEN: Simulating Corporate Environments with Autonomous Digital Employees in Multi-Horizon Task Environments. arXiv preprint arXiv:2602.14229, 2026. https://arxiv.org/abs/2602.14229

[35] Hussein Mozannar, Gagan Bansal, Cheng Tan, Adam Fourney, Victor Dibia, Jingya Chen, Jack Gerrits, Tyler Payne, Matheus Kunzler Maldaner, Madeleine Grunde-McLaughlin, Eric Zhu, Griffin Bassman, Jacob Alber, Peter Chang, Ricky Loynd, Friederike Niedtner, Ece Kamar, Maya Murad, Rafah Hosn, Saleema Amershi. Magentic-UI: Towards Human-in-the-loop Agentic Systems. arXiv preprint arXiv:2507.22358, 2025. https://arxiv.org/abs/2507.22358

[36] K. J. Kevin Feng, David W. McDonald, Amy X. Zhang. Levels of Autonomy for AI Agents. Knight First Amendment Institute 'AI and Democratic Freedoms' essay series, 2025; arXiv:2506.12469. https://arxiv.org/abs/2506.12469

[37] K. J. Kevin Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S. Weld, Amy X. Zhang, Joseph Chee Chang. Cocoa: Co-Planning and Co-Execution with AI Agents. CHI 2026; arXiv:2412.10999. https://arxiv.org/abs/2412.10999

[38] Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, Diyi Yang. Collaborative Gym: A Framework for Enabling and Evaluating Human-Agent Collaboration. ICLR 2026 (poster); arXiv:2412.15701. https://arxiv.org/abs/2412.15701

[39] Harang Ju, Sinan Aral. Collaborating with AI Agents: Field Experiments on Teamwork, Productivity, and Performance. arXiv preprint arXiv:2503.18238, 2025 (v4, 15 Oct 2026). https://arxiv.org/abs/2503.18238

[40] Harang Ju, Sinan Aral. Personality pairing improves human-AI collaboration. arXiv preprint arXiv:2511.13979, 2025 (v4, 28 Aug 2026). https://arxiv.org/abs/2511.13979

[41] Arsalan Shahid, Gordon Suttie, Philip Black. Collaborative Human-Agent Protocol (CHAP). arXiv preprint arXiv:2606.09751, 2026 (v2, 20 Jun 2026). https://arxiv.org/abs/2606.09751

[42] Elham Tabassi. Artificial Intelligence Risk Management Framework (AI RMF 1.0). National Institute of Standards and Technology, NIST AI 100-1, January 26, 2023; DOI 10.6028/NIST.AI.100-1. https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-ai-rmf-10

[43] Chloe Autio, Reva Schwartz, Jesse Dunietz, Shomik Jain, Martin Stanley, Elham Tabassi, Patrick Hall, Kamie Roberts. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. National Institute of Standards and Technology, NIST AI 600-1, July 26, 2024; DOI 10.6028/NIST.AI.600-1. https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence

[44] ISO/IEC JTC 1/SC 42. ISO/IEC 42001:2023 Information technology - Artificial intelligence - Management system. International Standard ISO/IEC 42001:2023, Edition 1.0, 2023-12-18. https://webstore.iec.ch/publication/90574

[45] European Parliament and Council. REGULATION (EU) 2024/1689 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act). Official Journal of the European Union, L series, 2024/1689, 12.7.2024; Regulation (EU) 2024/1689; ELI http://data.europa.eu/eli/reg/2024/1689/oj. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[46] European Parliament and Council. REGULATION (EU) 2026/1744 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 8 July 2026 amending Regulations (EU) 2024/1689, (EU) 2018/1139 and (EU) 2023/1230 as regards the simplification of the implementation of harmonised rules on artificial intelligence. Official Journal of the European Union, OJ L 2026/1744, 24.7.2026; Regulation (EU) 2026/1744 (Digital Omnibus on AI). https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L_202601744

[47] OWASP Gen AI Security Project. OWASP Top 10 for LLM Applications 2025. OWASP Gen AI Security Project, resource page dated November 17, 2024; entries LLM01:2025 and LLM06:2025. https://genai.owasp.org/llm-top-10/

[48] OWASP Agentic Security Initiative (Gen AI Security Project). Agentic AI – Threats and Mitigations. OWASP Gen AI Security Project, February 17, 2025. https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

[49] OWASP Gen AI Security Project - Agentic Security Initiative. OWASP Top 10 for Agentic Applications for 2026. OWASP Gen AI Security Project, resource page dated December 9, 2025 (PDF: Version 2026, December 2025); entries ASI01-ASI10. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

[50] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, Mario Fritz. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), pp. 79-90, 2023; DOI 10.1145/3605764.3623985; arXiv:2302.12173. https://arxiv.org/abs/2302.12173

[51] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, Florian Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track, 2024; DOI 10.52202/079017-2636; arXiv:2406.13352. https://arxiv.org/abs/2406.13352

[52] Qiusi Zhan, Zhixiang Liang, Zifan Ying, Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. Findings of the Association for Computational Linguistics: ACL 2024, pp. 10471-10506, 2024; DOI 10.18653/v1/2024.findings-acl.624; arXiv:2403.02691. https://aclanthology.org/2024.findings-acl.624/

[53] Scott Rose, Oliver Borchert, Stu Mitchell, Sean Connelly. Zero Trust Architecture. NIST Special Publication 800-207, August 2020; DOI 10.6028/NIST.SP.800-207. https://csrc.nist.gov/pubs/sp/800/207/final

[54] Vincent Hu, David Ferraiolo, Richard Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162, January 2014 (includes updates as of 08-02-2019); DOI 10.6028/NIST.SP.800-162. https://csrc.nist.gov/pubs/sp/800/162/upd2/final

[55] Tobin South, Samuele Marro, Thomas Hardjono, Robert Mahari, Cedric Deslandes Whitney, Dazza Greenwood, Alan Chan, Alex Pentland. Authenticated Delegation and Authorized AI Agents. arXiv preprint arXiv:2501.09674, 2025. https://arxiv.org/abs/2501.09674

[56] Alan Chan, Carson Ezell, Max Kaufmann, Kevin Wei, Lewis Hammond, Herbie Bradley, Emma Bluemke, Nitarshan Rajkumar, David Krueger, Noam Kolt, Lennart Heim, Markus Anderljung. Visibility into AI Agents. ACM FAccT 2024; arXiv:2401.13138. https://arxiv.org/abs/2401.13138

[57] Alan Chan, Kevin Wei, Sihao Huang, Nitarshan Rajkumar, Elija Perrier, Seth Lazar, Gillian K. Hadfield, Markus Anderljung. Infrastructure for AI Agents. TMLR 2025; arXiv:2501.10114. https://arxiv.org/abs/2501.10114

[58] Bruce Schneier, John Kelsey. Secure audit logs to support computer forensics. ACM Transactions on Information and System Security (TISSEC), 1999, pp. 159-176; DOI 10.1145/317087.317089. https://doi.org/10.1145/317087.317089

[59] Scott A. Crosby, Dan S. Wallach. Efficient Data Structures for Tamper-Evident Logging. USENIX Security '09 (Technical Sessions), USENIX Association, 2009. https://www.usenix.org/legacy/event/sec09/tech/full_papers/crosby.pdf

[60] B. Laurie, A. Langley, E. Kasper. Certificate Transparency. IETF RFC 6962 (Experimental), June 2013; DOI 10.17487/RFC6962. https://www.rfc-editor.org/rfc/rfc6962

[61] B. Laurie, E. Messeri, R. Stradling. Certificate Transparency Version 2.0. IETF RFC 9162 (Experimental), December 2021; DOI 10.17487/RFC9162. https://www.rfc-editor.org/rfc/rfc9162

[62] Luc Moreau, Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, 30 April 2013; REC-prov-dm-20130430. https://www.w3.org/TR/2013/REC-prov-dm-20130430/

[63] Timothy Lebo, Satya Sahoo, Deborah McGuinness. PROV-O: The PROV Ontology. W3C Recommendation, 30 April 2013; REC-prov-o-20130430. https://www.w3.org/TR/2013/REC-prov-o-20130430/

[64] Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, Justin Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USENIX Association, pp. 1393-1410, August 2019; ISBN 978-1-939133-06-9. https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[65] SLSA, The Linux Foundation. Build: Provenance. SLSA specification v1.2, slsa.dev (Linux Foundation); in-toto predicateType https://slsa.dev/provenance/v1. https://slsa.dev/spec/v1.2/build-provenance

[66] Renan Souza, Amal Gueroudji, Stephen DeWitt, Daniel Rosendo, Tirthankar Ghosal, Robert Ross, Prasanna Balaprakash, Rafael Ferreira da Silva. PROV-AGENT: Unified Provenance for Tracking AI Agent Interactions in Agentic Workflows. IEEE e-Science 2025; arXiv:2508.02866. https://arxiv.org/abs/2508.02866

[67] Yi Nian, Aojie Yuan, Haiyue Zhang, Jiate Li, Li Li, Xiyang Hu, Hua Wei, Xiongye Xiao, Chaowei Xiao, Yue Zhao. Auditable Agents. arXiv preprint arXiv:2604.05485 (v4, 27 Sep 2026), 2026; authors state a condensed 5-page version was accepted to ACM AI Summit 2026 (Visionary Papers track). https://arxiv.org/abs/2604.05485

[68] Jeremy Qin, David Schmotz, Derck Prinzhorn, Luca Beurer-Kellner, Ameya Prabhu, Maksym Andriushchenko. LLM Agents Can Easily Tamper With Their Own Traces. arXiv preprint arXiv:2609.30266 (v1, 24 Sep 2026), 2026. https://arxiv.org/abs/2609.30266

[69] National Institute of Standards and Technology. Secure Hash Standard (SHS). NIST FIPS 180-4, August 2015; DOI 10.6028/NIST.FIPS.180-4. https://csrc.nist.gov/pubs/fips/180-4/upd1/final

[70] H. Krawczyk, M. Bellare, R. Canetti. HMAC: Keyed-Hashing for Message Authentication. IETF RFC 2104 (Informational), February 1997; DOI 10.17487/RFC2104. https://www.rfc-editor.org/rfc/rfc2104

[71] Martin Kleppmann, Adam Wiggins, Peter van Hardenberg, Mark McGranaghan. Local-First Software: You Own Your Data, in spite of the Cloud. Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! '19), Athens, Greece, ACM, pp. 154-178, 2019; DOI 10.1145/3359591.3359737. https://doi.org/10.1145/3359591.3359737

[72] Different AI, Inc.. openwork: The open-source alternative to Claude Cowork (powered by opencode). GitHub repository different-ai/openwork; accessed 2026-11-04. https://github.com/different-ai/openwork

[73] anomalyco. opencode: The open source coding agent. GitHub repository anomalyco/opencode, MIT license; accessed 2026-11-04. https://github.com/anomalyco/opencode

[74] Nous Research. hermes-agent: The agent that grows with you. GitHub repository NousResearch/hermes-agent, MIT license; accessed 2026-11-04. https://github.com/NousResearch/hermes-agent