CTIFOUNDRY: AN AGENT-NATIVE CORPUS SCAFFOLD FOR CYBER THREAT INTELLIGENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Cyber threat intelligence (CTI) is increasingly consumed not by human analysts but by LLM agents that compose multi-step investigations at query time. The harness side of this shift has matured rapidly (planning loops, tool protocols, context management), but the corpus side has not: threat reports and vulnerability databases are still packaged for retrieval-augmented generation, as opaque chunks behind an embedding index. We argue that this substrate, not model capability, is the bottleneck on agentic CTI investigation, and present CTIFoundry, an *agent-native corpus scaffold*. At build time, CTIFoundry materializes the latent structure of a CTI corpus: a deterministic ontology graph over four authoritative knowledge bases (CVE, CWE, CAPEC, ATT&CK) whose official cross-references become typed, traversable edges; a span-grounded report layer whose canonical, alias-resolved cross-vendor entities index provenance-carrying chunks; and hybrid dense+lexical retrieval surfaces. At query time this structure is exposed through seven typed tools and three procedural skills mounted on a stock, widely-used open-source agent harness. On the public CTIConnect benchmark (nine tasks over entity linking, attribution, and multi-document synthesis), swapping only the action surface lifts the identically-harnessed agent from 0.610 to 0.829 overall F1 with `gpt-5.4` and from 0.470 to 0.745 with `claude-haiku-4-5`: a small model on CTIFoundry surpasses a flagship model on the flat substrate. The scaffolded agent is simultaneously more accurate and more efficient: on both Claude models it answers with roughly half the tool calls per question. The ablation distills design principles for matching corpus scaffolding to data modality, in CTI and beyond. Build-time validation guarantees zero fabricated identifiers by construction, and the scaffold sustains 1,168 investigations end-to-end at ≈2.6 cents each.

1 INTRODUCTION

A cyber threat intelligence (CTI) investigation is intrinsically multi-step. The same adversary appears under different names across vendor reports (*Lazarus*, *Hidden Cobra*, *APT38*), so any question about it first requires resolving aliases to one canonical entity. A behavioral description must then be linked to an authoritative taxonomy (CVE, CWE, CAPEC, ATT&CK), usually through an *official cross-reference*: the CVE record names its CWE weakness, the CAPEC pattern names the ATT&CK technique it maps to. And a campaign profile is scattered across vendors, each holding a fragment. Resolve, traverse, collect, reconcile: exactly the kind of tool-mediated procedure that large language model (LLM) agents are built to compose. The interleaved reason–act loop (Yao et al., 2023) and learned tool invocation (Schick et al., 2023) have hardened into a commodity stack of open-source harnesses (Yang & mini-swe-agent contributors, 2025; Anthropic, 2024; 2025b;a), and purpose-built agent–computer interfaces have delivered striking results elsewhere (Yang et al., 2024).

This progress is unbalanced. The *harness* improves with every release, but dropping a more capable agent into a vertical domain does not produce a capable domain investigator: the agent inherits whatever substrate the domain's corpora are packaged in. In CTI that packaging is inherited from retrieval-augmented generation (RAG), opaque chunks behind a single similarity-search interface (Lewis et al., 2020), with three consequences. Vendor aliases are never resolved, so reports about one actor shard across names sharing no surface vocabulary; the official cross-references that authoritatively answer entity-linking questions survive only as text inside record blobs; and derived claims carry no span-level provenance, so nothing separates an authoritative cross-reference from a

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

textual co-occurrence. Putting an agent on top repairs none of it: iterating over an opaque substrate only re-retrieves, and cannot recover structure that indexing discarded. The public CTIConnect benchmark (Cheng et al., 2026b) measured the consequence, and in our own runs a state-of-practice harness over the flat corpus cannot follow an official cross-reference even when it has already retrieved the record carrying it (Appendix E.4). We distill these gaps into four challenges (C1–C4, §2). The binding constraint is the *substrate*, not the agent.

We present CTIFoundry, an agent-native corpus scaffold with two halves (Figure 1, left). At *build time* it materializes the corpus's latent structure into typed, validated artifacts (Table 4): a deterministic *ontology graph* whose nodes are the entries of four authoritative knowledge bases and whose typed edges are their official cross-references, built under a zero-fabrication invariant (C1); a *span-grounded report layer* that chunks vendor reports with exact character-offset provenance and resolves typed mentions, deterministic signals first, into canonical cross-vendor entities that key the chunks (C2); and *hybrid retrieval surfaces* fusing dense and lexical search under a term filter whose pool-size feedback the agent can steer (C3). At *query time* this is exposed through *seven typed tools*, each covering one non-overlapping capability and self-described with usage guidance and cost, and *three procedural skills* encoding investigation discipline: resolve before searching, traverse official edges before trusting text similarity, verify every candidate (C4).

We evaluate on CTIConnect's nine tasks under a deliberately controlled methodology: both arms run on mini-swe-agent (Yang & mini-swe-agent contributors, 2025), the baseline with its native bash tool over the corpus dumped to flat files, the CTIFoundry arm with only the action surface swapped. Loop, step budget, temperature, and model are identical, so any gap is attributable to the substrate. The swap lifts overall F1 from 0.610 to 0.829 for `gpt-5.4` and from 0.470 to 0.745 for `claude-haiku-4-5`, by $+0.19$ to $+0.28$ across a four-model, two-provider panel (Table 2, Figure 1, right). The *shape* matters as much as the size: the gain concentrates where the benchmark located the RAG bottleneck and vanishes on the one task with no authoritative structure to materialize. Structure moreover partially substitutes for scale, a small model on CTIFoundry surpasses the flagship on flat files, and the scaffolded agent is simultaneously more accurate and more efficient, answering with roughly half the tool calls on both Claude models.

Contributions.

1. *The substrate bottleneck.* Under a fixed third-party harness in which the action surface is the sole experimental variable, we show the binding constraint on agentic CTI investigation is the corpus substrate, not agent capability: iteration over a flat substrate cannot recover structure its packaging discarded (§2, §4.1).
2. *An agent-native corpus scaffold.* We define the scaffold and the harness boundary and realize it as a build-time pipeline (ontology graph, span-grounded canonical entities, hybrid retrieval) under validated zero-fabrication invariants, exposed through seven typed tools and three procedural skills (§3).
3. *An empirical study, and what it generalizes to.* Across nine tasks the swap lifts F1 by $+0.19$ to $+0.28$ at zero fabricated identifiers and roughly half the tool calls (§4), and a 2×2 ablation shows the two halves compose super-additively (§4.5); structure makes the right investigation possible, procedure makes it reliable, and neither substitutes for the other. The lesson transfers to any vertical whose corpora carry authoritative reference structure.

2 Background and Motivation

Operational CTI knowledge lives in two sources of sharply different shape. Four community-maintained taxonomies form the reference backbone (CVE (vulnerability instances), CWE (weakness classes), CAPEC (attack patterns), and MITRE ATT&CK (adversary techniques) (Strom et al., 2018)), and crucially they are not four independent lists: they *officially cross-reference* one another, a CVE record naming the CWE it instantiates, a CAPEC pattern the CWEs it exploits and the ATT&CK techniques it maps to. For a large class of analyst questions these curated edges are *the* authoritative answer: "which weakness underlies this vulnerability" is not a matter of textual similarity but a recorded edge. The second source is the narrative layer written by security vendors, whose central entities (threat actors, malware families, campaigns) carry *vendor-specific naming*, so intelligence about one campaign is sharded across reports that share no surface vocabulary (Appendix B).

CTIConnect (Cheng et al., 2026b) operationalizes this workflow as a public benchmark, and is to date the only CTI benchmark that evaluates LLMs with retrieval access to the domain's knowledge sources rather than closed-book: 1,859 expert-verified questions over the four knowledge bases plus 321 report summaries, in nine tasks over three families, *entity linking* (EL), *entity attribution* (EA), and *multi-document synthesis* (MDS). Its evaluation is confined to the RAG setting, and its authors name agentic design over the corpus as the open direction. Its published diagnostics establish *where* LLM-over-CTI fails, and we build on that measurement rather than repeat it (Appendix C): a cross-source semantic gap widens with the heterogeneity a task must bridge, sinking gold evidence past the typical top-k window through *aliasing*, *register mismatch* between narrative prose and taxonomy terminology, and *sibling confusion* among lexically adjacent entries, failures that are structural rather than incidental, since general-purpose retrieval upgrades recover only a fraction of what interventions on vocabulary and entity structure do. Joined by two demands operational CTI adds (that every claim be auditable back to the vendor and sentence asserting it, and that the analyst's procedural discipline is written down nowhere in the corpus) they give four challenges an agent-facing substrate must meet, each answered by one CTIFoundry component:

- **C1** *Materialize the latent structure.* Canonical cross-vendor aliases and official cross-references must become typed records and traversable edges, not phrases to rediscover by similarity search: closing the aliasing and sibling-confusion gaps at their source. ⇒ the deterministic ontology graph and the canonical entity layer (§3.2).
- **C2** *Ground every derived assertion in provenance.* Extracted entities and groundings must point back to exact source spans with vendor attribution, so the agent (and the build validator) can verify rather than trust. ⇒ span-grounded chunks under a zero-fabrication validation regime (§3.2).
- **C3** *Speak both retrieval languages.* Dense search bridges paraphrase; lexical filtering pins rare discriminative tokens; CTI questions routinely need both, and each covers the other's failure mode, the register mismatch measured above. ⇒ hybrid dense+BM25 surfaces with a steerable term filter (§3.2).
- **C4** *Ship the analyst's procedure with the interface.* Which tool to call first, when an official edge outranks a text match, how to verify a candidate; this discipline is corpus-specific and must accompany the scaffold, not be rediscovered per query. ⇒ per-task-family procedural skills over self-described typed tools (§3.3, §3.3).

CTIFoundry's build-time layers answer C1–C3; its query-time skill layer answers C4. The next section presents both.

3 CTIFOUNDRY

3.1 PROBLEM FORMULATION

We study *agent-native corpus scaffolding*: given a domain corpus and a fixed agent loop, build a derived representation of the corpus, and an action surface over it, that maximizes investigation accuracy without modifying the agent. Every deployed system already has such a representation, however thin (flat files are a degenerate one, classic RAG's dense index a weak one, preserving similarity geometry while discarding every other structure the corpus carries), so what varies is not whether a scaffold exists but how much of the corpus it keeps reachable. The agent loop above it is increasingly a commodity (§5), and the corpus below is given by the domain; those we hold fixed.

Corpus. A CTI corpus is $C = R \cup K$, where R is a set of vendor reports (free text with vendor metadata), and $K = K_{\text{cve}} \cup K_{\text{cwe}} \cup K_{\text{capec}} \cup K_{\text{att}}$ is a set of knowledge-base records over four authoritative taxonomies, each record carrying a canonical identifier and cross-references to other taxonomies.

Definition 1 (Scaffold)**.** A *scaffold* over C is a derived, validated, indexed representation through which an agent accesses the corpus,

$$\mathsf{S}(C) \;=\; \big(G,\; X,\; \mathcal{E},\; \Pi,\; \mathcal{I}\big),$$

the *ontology graph* $G = (V, E)$ over KB records and their typed official cross-reference edges; provenance-carrying *chunks* X, each a document span with character offsets; *canonical entities* $\mathcal{E}$,

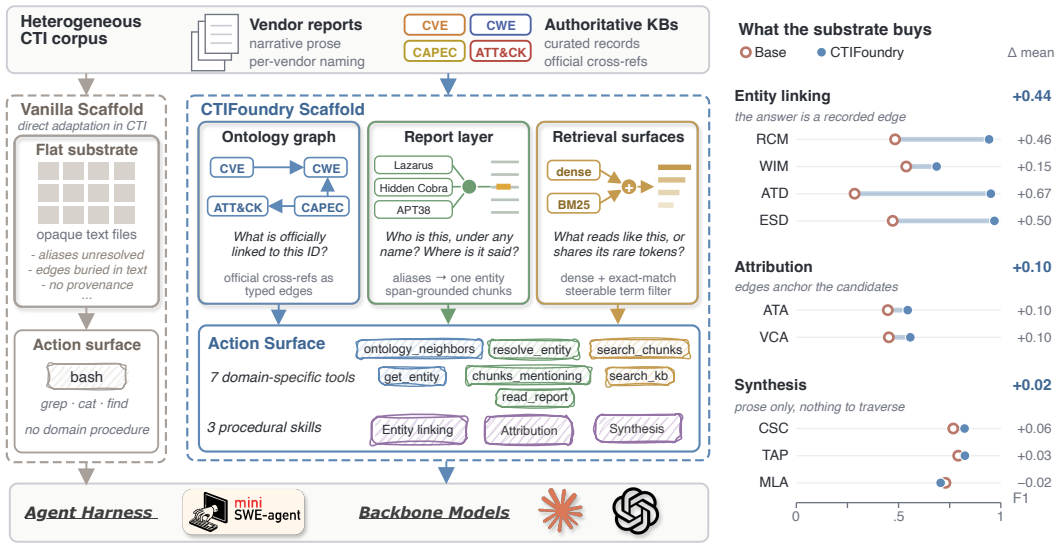


Figure 1: **What the substrate buys, and what produces it.** Left: the vanilla scaffold, a direct adaptation in CTI, dumps the heterogeneous corpus as opaque files that the agent reads through `bash`, while CTIFoundry materializes it at build time into an ontology graph, a span-grounded report layer, and hybrid retrieval surfaces, exposed at query time through seven typed tools and three procedural skills. Right: per-task F1 of the two arms, averaged over the four-model panel, with the mean gain per task and per family (§4.3, Table 2).

each a cluster of report mentions with a vendor-attributed alias set and optional grounding in V; the entity index $\Pi : \mathcal{E} \to 2^X$; and dense and lexical indexes $\mathcal{I}$. It is *admissible* if it fabricates no identifier: $\mathrm{ids}(\mathsf{S}(C)) \subseteq \mathrm{ids}(C)$.

Definition 2 (Harness and action surface)**.** A *harness* is an agent loop $L\langle m, b, A\rangle$: a fixed control program parameterized by a model m, a step budget b, and an *action surface* A, the operations it may invoke against the scaffold. It is *corpus-agnostic* (L, m, b carry no knowledge derived from C), so corpus-derived artifacts reach the agent only through A or through prompt-injected procedural text, a *skill* σ.

3.2 The Scaffold

The build produces the three artifacts of Definition 1 under a single rule: *the mechanism follows the evidence*. Where the corpus already records the answer the build parses it, nothing generative intervening; where only prose carries it, in-context extraction runs bracketed by deterministic guards; where the question merely paraphrases its evidence, matching is delegated to a commodity embedding model. Appendix A gives the pipeline stage by stage, with artifact counts, the resolution algorithm, and the operator prompts.

Ontology graph (C1). The cross-references that answer entity-linking questions are curated by the taxonomy maintainers and shipped inside the released records, so this layer's task is *lossless preservation*, deterministic parsing from pinned snapshots, no model in the loop. Two invariants close it. *Zero fabrication* holds by construction rather than by post-hoc filtering: an identifier is written only after it validates against the snapshot. *Closure over the released corpus*: every edge originates in a KB record the baseline retrieves over as well, so the layer contributes representation rather than information, and §4.5 prices that representation on its own.

Report layer (C2). Vendor prose records nothing explicitly, so where the ontology layer preserves structure this layer must *recover* it. Two commitments distinguish it from the extraction line it builds on (Cheng et al., 2025). The output schema is dictated by the consumer, an investigating agent retrieves entities and reads chunk text, so the layer emits typed mentions, TTP groundings, and the entity→chunk index, and *no* relational triples, which nothing downstream would follow. And every generative step is bracketed by a deterministic guard wherever determinism is available, so what the model contributes is recall and what the build guarantees is validity: identifier-bearing mentions are captured by regex, every T-id is validated against the ATT&CK snapshot, and entity resolution is a

union-find ordered so deterministic evidence dominates, only exact and span-verified alias evidence enters the union, which is what stops distinct state actors collapsing into mega-clusters (§4.2).

Retrieval surfaces (C3). The remaining access mode is the one neither parsed structure nor extracted entities can serve: questions that *paraphrase* their evidence. Paraphrase matching is a commodity, delegated to an off-the-shelf embedding model; what the scaffold contributes is the surface around it. Knowledge-base search fuses dense and BM25 rankings by reciprocal-rank fusion (Cormack et al., 2009; Robertson & Zaragoza, 2009), over which the caller may pass `must_terms`, a conjunctive filter, and read back the surviving pool size. That one field makes the surface *steerable* (too many hits means add a term, zero means swap a synonym), turning a one-shot ranking into an operator the agent controls, and closing the failure mode that defeats purely dense retrieval here: a gold entry sharing rare discriminative tokens with the query yet sitting far from it in embedding space.

3.3 The Action Surface

The scaffold is consumed through seven typed tools that make the right investigation *possible* and three procedural skills that make it *likely*. The two are not independent contributions: A is constrained by S, so a skill prescribing "traverse the official edge" is inert unless that edge exists, an interaction §4.5 measures.

Seven typed tools. Access goes through seven typed tools (Appendix D) designed under three rules. *Non-overlap*: each exposes exactly one scaffold capability (resolution, record fetch, ontology traversal, KB search, chunk search, entity-indexed collection, document read), so tool choice is never ambiguous. *Self-description*: each states when to use it and what it costs (Anthropic, 2025d). *Structure before similarity*: descriptions encode the substrate's priority order (resolve names before querying, prefer an authoritative edge over text search whenever an identifier is known), so the ordering the build makes possible is the one the surface advertises.

Procedural skills. No corpus writes down the analyst's procedure (C4). CTIFoundry ships it as three markdown skill files, one per task family, injected into the user turn. They are advice, not workflow engines, but encode discipline distilled from trajectory analysis of agent failures: for *entity linking*, never search the target taxonomy first, since the question paraphrases one source entry whose official cross-reference gives the answer; for *attribution*, restate each behavior in the target taxonomy's idiom and emit a calibrated minimal covering set, because under identifier F1 a spurious identifier costs what a miss does; for *synthesis*, cover the report cluster exhaustively and merge field-by-field across vendors. Each prescription names an action the build made available, binding the two query-time layers by construction. The playbooks are reproduced in Appendix H.

4 Evaluation

We ask four questions on the public CTIConnect benchmark (Cheng et al., 2026b): is the build sound (**RQ1**, §4.2); does it make an *identically-harnessed* agent more accurate across model families and scales (**RQ2**, §4.3); is that accuracy bought with search effort, and at what cost (**RQ3**, §4.4); and which half, typed structure or procedural skill, carries the gain (**RQ4**, §4.5)? Appendix E adds accuracy and cost at $1.7\times$ the question volume and a single trajectory traced on both arms.

4.1 Experimental Setup

Benchmark, corpus, and harness. CTIConnect contains 1,859 expert-verified questions over nine tasks in three families (§2): entity linking (EL: RCM, WIM, ATD, ESD), attribution (EA: ATA, VCA), and multi-document synthesis (MDS: CSC, TAP, MLA), over the four knowledge bases and 321 report summaries (counts in Table 4). We follow its two-set protocol: a *main set* of 691 questions carries the controlled comparisons (RQ1, RQ2, RQ4), a *scale set* of 1,168 the cost and volume studies (RQ3, Appendix E.2); prompts, tools, and skills were frozen on a held-out development slice. The obvious threat to any "our agent wins" claim is a harness tuned to the proposed substrate, and we remove it by construction: both arms run the stock mini-swe-agent (Yang & mini-swe-agent contributors, 2025) `DefaultAgent` over exactly this corpus. The *base agent* is the harness out of the box, with its native single-bash surface over the corpus dumped to disk, what a practitioner gets today; the

CTIFoundry agent is the same harness with *only* the action surface swapped for the seven typed tools and three skills of §3.3. Loop, prompt style, step budget (20), temperature, and model are identical, so the action surface is the sole independent variable. The panel spans two providers and two tiers (`gpt-5.4`, `gpt-5.4-mini`, `claude-sonnet-4-6`, `claude-haiku-4-5`), and the build is compiled once under fixed operators, so every CTIFoundry row reads byte-identical artifacts.

Metrics. All query-time scores are the benchmark's, computed identically for every arm. EL and EA use *identifier-normalized F1*: identifiers in the final answer are normalized (case, prefix, sub-technique suffix, `T1059.001` vs. `T1059`) into a predicted set and compared against gold. Two properties carry weight below: the metric is *set-valued*, so multi-answer attribution is scored element-wise, and *symmetric in error type*, so a hedged extra identifier costs exactly what a miss does, recall cannot be bought with unresolved candidates (Appendix E.4). MDS is scored by the benchmark's claim-level judge (`gpt-5.4`) with fixed prompt across arms, read at the resolution we audit below. *Overall* is the unweighted nine-task mean. Effort and cost are *measured, not budgeted*: every run serializes its trajectory, from which we recompute calls per question and provider-reported tokens at list rates.

4.2 RQ1: Build-Time Quality

Structural validity. From the 321 reports and four KB snapshots the build materializes 6,044 ontology nodes, 7,097 official edges, 688 provenance-carrying chunks, and 4,868 canonical entities. A validator *blocks the build* on three violation classes (fabricated identifiers, orphan edges, and span violations (recorded offsets that do not re-verify byte-for-byte against the frozen source)), and the shipped build passes with zero. The first class is impossible by construction rather than filtered post hoc: every identifier is checked against the snapshots at the moment it is written. End-to-end build cost is 1.42M tokens ($1.86), linear in corpus size.

Entity resolution. Against the benchmark's 50 adversary-centric report clusters (used only as labels, never at build time), we select for each gold cluster g the canonical entity ε whose report set best matches it and report coverage $= |R(\varepsilon) \cap g|/|g|$, purity $= |R(\varepsilon) \cap g|/|R(\varepsilon)|$, and their harmonic mean. CTIFoundry reaches 0.900 / 0.927 (F1 0.913) with 27 clusters reconstructed exactly, against 0.840 / 0.920 (F1 0.878), and 21 for the extraction graph shipped with the benchmark. The gain is *coverage at unchanged purity*, which is the non-trivial direction: the cheap way to raise coverage is transitive merging, and it is exactly what the merge discipline of §3.2 forbids, admitting only exact and span-verified alias evidence into the union rules out the mega-cluster failure mode that sinks the benchmark graph (its best `APT42` entity spans nine reports at purity 0.33).

Extraction quality across operators. Table 1 varies the extraction operator over six models (protocol in Appendix E.1): flagships reach F1 0.859 / 0.792 (entity / TTP), and small operators trail by 10–20 points, but *almost entirely in recall*: they under-extract rather than invent, the one degradation mode a build pipeline can absorb, since a missing mention costs coverage while a fabricated one would breach the invariant the whole scaffold rests on.

Judge reliability. Two auditors with 3+ years of professional CTI experience independently re-scored a 50-item MDS sample blind to the judge's verdicts. Agreement is close (Pearson r 0.85; mean per-item absolute difference 0.06; mean 0.705 vs. 0.713), i.e. agreement on ranking with no strictness offset. MDS numbers are therefore comparable *across configurations under one judge*, the only comparison we make, and 0.06 is the resolution at which we read MDS gaps below.

4.3 RQ2: Query-Time Quality

The substrate swap lifts overall F1 by +0.219 (`gpt-5.4`, 0.610→0.829), +0.190 (`gpt-5.4-mini`), +0.222 (`claude-sonnet-4-6`), and +0.275 (`claude-haiku-4-5`) (Table 2, Figure 1, right). The headline is not the magnitude but the *shape*: the tasks where the gain fails to appear are as informative as those where it saturates.

The gain tracks materialized structure, and stops where it stops. On the three forward EL tasks whose official cross-references the build materializes, the plan collapses to resolve-then-traverse and approaches ceiling (`gpt-5.4`: RCM 0.59→1.00, ATD 0.32→1.00, ESD 0.43→0.95), with `claude-haiku-4-5` reaching a full 1.00 on ATD and ESD, the smallest model in the panel

Table 1: Build-time extraction quality as the operator model is varied. **best** and second mark the best and runner-up operator per column.

Operator model	Entity extraction			TTP grounding		
	Prec	Rec	F1	Prec	Rec	F1
gpt-5.4	0.846	**0.872**	**0.859**	**0.749**	**0.841**	**0.792**
gpt-5.4-mini	0.708	0.781	0.743	0.611	0.692	0.649
gpt-5.4-nano	0.691	0.712	0.701	0.574	0.580	0.577
claude-opus-4-1	**0.849**	0.748	0.795	0.727	0.735	0.731
claude-sonnet-4-6	0.778	0.771	0.775	0.627	0.806	0.705
claude-haiku-4-5	0.780	0.653	0.711	0.551	0.599	0.574

Table 2: Main-set per-subtask scores under the two action surfaces; harness, model, budget, and temperature are identical within a model. **better** and **best** mark the better arm per task and overall.

Model	Config	EL				EA		MDS			Overall
		RCM	WIM	ATD	ESD	ATA	VCA	CSC	TAP	MLA	
gpt-5.4	Base	0.588	0.660	0.317	0.425	0.523	0.497	0.845	0.830	**0.804**	0.610
	CTIFoundry	**1.000**	**0.723**	**1.000**	**0.950**	**0.631**	**0.625**	**0.874**	**0.852**	0.802	**0.829**
gpt-5.4-mini	Base	0.575	0.511	0.317	0.350	0.456	**0.633**	0.776	**0.793**	**0.698**	0.568
	CTIFoundry	**0.900**	**0.681**	**0.925**	**0.925**	**0.581**	0.533	**0.796**	**0.793**	0.687	**0.758**
claude-sonnet-4-6	Base	0.278	0.766	0.326	0.611	**0.473**	0.320	0.753	0.758	**0.745**	0.559
	CTIFoundry	**0.926**	**0.808**	**0.880**	**1.000**	0.469	**0.593**	**0.828**	**0.840**	0.685	**0.781**
claude-haiku-4-5	Base	0.491	0.213	0.180	0.500	0.337	0.360	0.696	0.785	**0.670**	0.470
	CTIFoundry	**0.944**	**0.532**	**1.000**	**1.000**	**0.498**	**0.479**	**0.793**	**0.813**	0.649	**0.745**

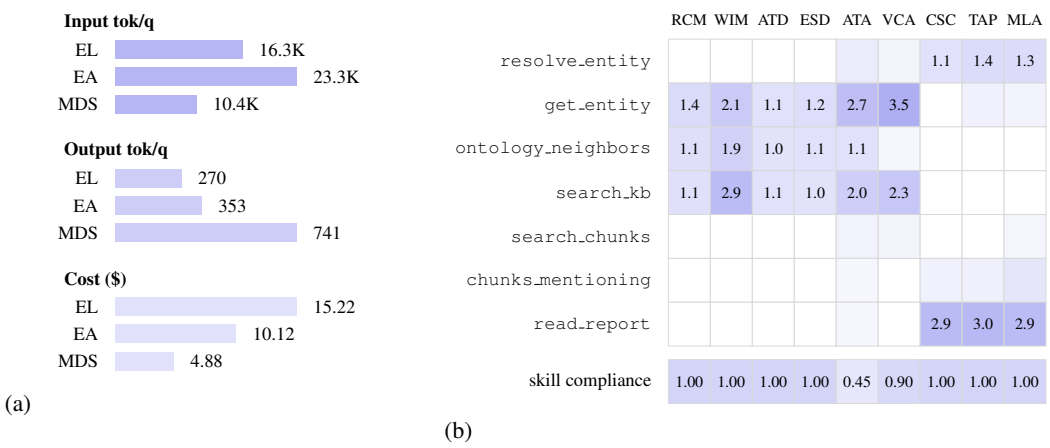
saturating tasks on which the flagship scored 0.32 and 0.43 over flat files. At the other extreme, MLA has no authoritative structure to materialize and the task reduces to rewriting the same report text under either surface: the arms are level, within the judge's 0.06 per-item resolution. A substrate contribution on MLA too would be the result we could not explain; its absence is the control the argument needs.

Structure pays even where it does not hold the answer. EA's answer is *not* a recorded edge, the scaffold can only anchor the candidate set the model must then discriminate among, yet it still yields +0.12 to +0.14 pooled F1 on three of four models. The single exception, `gpt-5.4-mini` on VCA, is also the one cell where a base arm wins outright. WIM makes the same point from the other side: it is the one EL task with no forward edge, and base-arm scores span 0.66 to 0.21 across the panel, a 0.45 spread on a fixed retrieval interface that retrieval cannot explain and *parametric CVE memory* can. CTIFoundry lifts every model regardless of what it memorized, converting a capability the flat corpus can only borrow from the model into one the substrate supplies.

The substrate outweighs a capability tier. Read across rows: `claude-haiku-4-5` (0.745), and `claude-sonnet-4-6` (0.781) on CTIFoundry both beat flagship `gpt-5.4` on the flat substrate (0.610), by a wider margin than separates flagship from small model *within* either arm. The effect is largest where model capability is smallest (+0.275 vs. +0.219), so the scaffold does most work where the model does least while remaining large at the frontier, the same accuracy at a fraction of the per-query model cost, bought once, offline. The obvious alternative explanation, that CTIFoundry simply searches harder, is tested next and does not survive the trajectories.

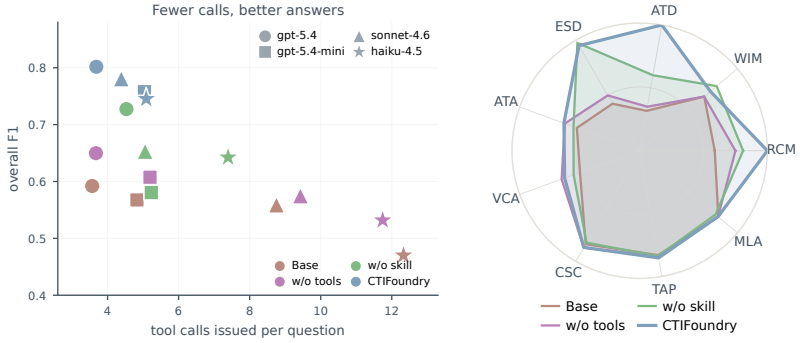
4.4 RQ3: Search Effort and Operating Cost

Figure 3a places every (model, arm) cell on the cost/accuracy plane. On the GPT models the move is essentially vertical (+0.12 calls per question for +0.219 F1 on `gpt-5.4`, +0.21 for +0.190 on `gpt-5.4-mini`), so a 3% change in effort does not buy a 36% change in accuracy. On the Claude models it inverts outright: `claude-haiku-4-5` issues 12.33 calls per question on flat files against 5.09 on CTIFoundry, less than half, while gaining +0.275 F1; `claude-sonnet-4-6` goes 8.75 → 4.39 for +0.222. The flat-substrate agent is therefore not under-searching but *over*-searching: lacking a traversable structure it re-probes the corpus and still lands on plausible-but-wrong entries. Within



(a) (b)

Figure 2: **Where the query-time budget goes.** (a) Per-family cost profile over the 1,168-question scale set (`gpt-5.4`), one shade per metric. (b) Mean calls per question by tool and subtask, with skill first-call compliance beneath.



(a) (b)

Figure 3: **Two diagnostic views of the same runs.** (a) Overall F1 against tool calls per question, main set (§4.4). (b) The 2×2 ablation over the nine task axes, `gpt-5.4` (§4.5).

every arm accuracy *decreases* monotonically with call count, so a long call sequence marks an item the agent could not resolve, never an investigation that paid off: and the ablation locates the mechanism, since *w/o skill* issues the most calls of any configuration (4.53 against CTIFoundry's 3.68), and still trails by 0.083 overall. Cost follows the same logic (Figure 2a): at ≈2.6 cents and ≈7 agent-seconds per investigation a single sweep recovers the one-time $1.86 build many times over, and the per-family profile tracks the design rather than the corpus size. Figure 2b shows the intended plans are the plans the agent runs: forward EL is a tight `search_kb`→`ontology_neighbors`→`get_entity` spine, while MDS leans on `read_report` and the entity index and never touches the ontology tools; structure used where it exists and ignored where it does not, unprompted by any router (trajectory in Appendix E.4).

4.5 RQ4: Ablation

CTIFoundry adds exactly two things to the stock harness (the typed-tool surface in place of bash, and the per-task skill), and a 2×2 over the same harness, model, and metrics separates them (Table 3, Figure 3b). Taken alone, neither is the system. *Procedure alone* recovers +0.062 (0.672 vs. 0.610): its reading and verification discipline transfers to bash, but its central prescriptions (resolve, traverse an official edge, consult an alias set) name actions the flat surface cannot perform. *Structure alone* reaches 0.746, with the deficit concentrated exactly where discipline decides the outcome (ATD 0.60 vs. 1.00, RCM 0.81 vs. 1.00 once the skill is added); trajectories show the undisciplined agent searching the *target* taxonomy directly and landing on plausible-but-wrong entries. Read per family, the table maps data modality to the scaffold it needs. EL, whose answers are recorded cross-references, depends on materialized structure and collapses without it (ATD 1.000→0.350, ESD 0.950→0.500 w/o tools). EA grounds narrative prose to a taxonomy in which the scaffold cannot look the answer

Table 3: 2×2 ablation of the two additions to the stock harness (main set, `gpt-5.4`); the all-off corner is mini-swe-agent itself. **best** and second mark best and runner-up per task.

Config	EL				EA		MDS			Overall
	RCM	WIM	ATD	ESD	ATA	VCA	CSC	TAP	MLA	
CTIFoundry (full)	**1.000**	0.723	**1.000**	0.950	**0.631**	0.625	0.874	**0.852**	0.802	**0.829**
w/o skill	0.810	**0.787**	0.600	**0.975**	0.551	0.550	0.831	0.837	0.776	0.746
w/o tools	0.750	0.660	0.350	0.500	0.623	**0.650**	**0.876**	0.842	0.801	0.672
w/o skill & tools	0.588	0.660	0.317	0.425	0.523	0.497	0.845	0.830	**0.804**	0.610

up, and is carried instead by the skill: w/o tools it essentially matches the full system (ATA 0.623 vs. 0.631, VCA 0.650 vs. 0.625), while w/o skill it falls to 0.551 and 0.550. MDS needs neither, its four configurations falling inside the judge's resolution. Where both matter the composition is the finding: +0.062 and +0.136 separately but +0.219 together (0.610 → 0.746 → 0.829), super-additive rather than the +0.198 independent contributions would predict, the interaction §3.3 anticipated, and the reason neither an off-the-shelf skill library nor a richer index would substitute for the other half. A deployment finding falls out alongside: identical skill text placed in the system prompt was under-followed by smaller models yet followed reliably in the user turn, which we attribute to instruction-following asymmetries in current models rather than to anything CTI-specific; all runs inject skills in the user turn.

5 Related Work

Structuring CTI. Turning reports into structure has moved from indicator mining (Liao et al., 2016) through bespoke behavior-extraction pipelines (Husari et al., 2017; Zhu & Dumitras, 2018; Satvat et al., 2021; Li et al., 2022; Alam et al., 2023) to LLM-based construction (Cheng et al., 2025), alongside unified graphs over the taxonomies themselves (Strom et al., 2018; OASIS, 2021; Hemberg et al., 2021). All treat extraction as the endpoint, so the product is a static analytic asset. CTIFoundry takes extraction quality as a solved input and asks instead what shape structure must take to be *traversed* by an investigating agent. Evaluation has meanwhile moved from representation quality (Ranade et al., 2021) through closed-book probes (Alam et al., 2024) to CTIConnect (Cheng et al., 2026b), the corpus-grounded setting we adopt unchanged.

Agent harnesses and scaffolds. The reason–act loop (Yao et al., 2023) and learned tool invocation (Schick et al., 2023) have hardened into purpose-built agent–computer interfaces (Yang et al., 2024; Yang & mini-swe-agent contributors, 2025; Anthropic, 2024; 2025d;a) and deployable skills (Anthropic, 2025c; Wang et al., 2024; Zhang et al., 2025; Cheng et al., 2026a; Wang et al., 2023; Shinn et al., 2023), which self-evolving agents now rewrite for themselves (Lee et al., 2026; Zhang et al., 2026; Lou et al., 2026; Lin et al., 2026; Chen et al., 2026a). Throughout, the editable layer is the *harness* (prompts, skills, control logic, and the tools' code) while the substrate underneath stays whatever the corpus was packaged as. A tool an agent writes for itself composes only what that substrate affords; it cannot author an edge the corpus never materialized. CTIFoundry therefore holds the harness fixed and rebuilds what it acts on. Appendix F situates this against agentic and deep-research retrieval, LLM-augmented data management and knowledge-base construction, and conversational memory stores.

6 Conclusion

CTI investigation is multi-step by nature, and the agents now asked to perform it are only as good as the substrate they investigate. This paper presented CTIFoundry, an agent-native corpus scaffold that materializes at build time what analysts traverse at query time (authoritative cross-reference edges, canonical cross-vendor entities with span-level provenance, and dual dense+lexical retrieval surfaces), exposed through typed tools and procedural skills on a stock agent harness. Swapping only the action surface improves the same agent by +0.19 to +0.28 overall F1 across a four-model panel, at matched or lower search effort, and the ablation completes the account: structure makes the right investigation possible, procedure makes it reliable, neither substitutes for the other. The claim extends past CTI: for any domain whose corpora carry authoritative reference structure, the highest-leverage investment in agent quality may not be a better agent at all, but a corpus deliberately built to be investigated.

AI USE STATEMENT

In this work, we used generative AI tools as an object of study and as a component of the system: the models named in §4.1 are the operator models of the build pipeline and the agents under evaluation, and their use is documented in full in §3 and §4. We additionally used generative AI assistance for writing support (copy-editing and LaTeX formatting), and for coding support during implementation of the build pipeline and evaluation scripts. We did not use generative AI tools to generate research ideas, to produce experimental results, or to write or select the related work. All AI-assisted code was reviewed and tested by the authors, and all reported numbers come from executed runs. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, and artifacts.

ETHICS STATEMENT

This work studies public cyber threat intelligence: the four community-maintained taxonomies (CVE, CWE, CAPEC, ATT&CK), and the vendor report summaries released with the public CTIConnect benchmark (Cheng et al., 2026b). No human subjects, no private or personally identifying data, and no proprietary victim data are involved, and no new attack capability is created: CTIFoundry reorganizes already-public defensive reference material into a form an analyst-facing agent can traverse, and the artifact is intended for defensive triage and attribution. The residual dual-use consideration is that faster, better-grounded navigation of public CTI is available to any reader, which we judge to be outweighed by the defensive benefit, since the same material is already public and the scaffold adds no non-public knowledge. CTIFoundry's outputs are decision support, not adjudication: attribution claims carry span-level provenance precisely so that a human analyst can audit them, and should not be treated as established fact without that review. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

The scaffold construction is specified in §3.2 (ontology graph, span-grounded report layer, hybrid retrieval surfaces) with the validation invariants it is built under, and the query-time surface, the seven typed tools and the three procedural skills, in §3.3 and Table 5. The evaluation protocol, including the harness (mini-swe-agent), the step budget, the temperature, the model versions, and the scoring procedure for each of the nine CTIConnect tasks, is given in §4.1; both arms share every setting except the action surface, which is the sole experimental variable. All corpora are public: the CTIConnect benchmark and its released corpus (Cheng et al., 2026b), and the four upstream knowledge bases. Source code for the build pipeline, the tool server, and the skill files, together with the evaluation harness, is submitted as anonymized supplementary material and will be released publicly upon publication.

ACKNOWLEDGMENTS

Omitted for double-blind review.

REFERENCES

Md Tanvirul Alam, Dipkamal Bhusal, Youngja Park, and Nidhi Rastogi. Looking beyond IoCs: Automatically extracting attack patterns from external CTI. In *RAID*, 2023.

Md Tanvirul Alam, Dipkamal Bhusal, Le Nguyen, and Nidhi Rastogi. CTIBench: A benchmark for evaluating LLMs in cyber threat intelligence. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Anthropic. Model context protocol. https://modelcontextprotocol.io, 2024.

Anthropic. Effective context engineering for AI agents. https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents, 2025a.

Anthropic. Effective harnesses for long-running agents. https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents, 2025b.

Anthropic. Equipping agents for the real world with agent skills. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills, 2025c.

Anthropic. Writing effective tools for agents — with agents. https://www.anthropic.com/engineering/writing-tools-for-agents, 2025d.

Mingju Chen, Can Lv, Guibin Zhang, Heng Chang, and Shiji Zhou. Harnessforge: Joint harness and policy evolution for adaptive agent systems. *arXiv preprint arXiv:2606.01779*, 2026a.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026b.

Yutong Cheng, Osama Bajaber, Saimon Amanuel Tsegai, Dawn Song, and Peng Gao. CTINexus: Automatic cyber threat intelligence knowledge graph construction using large language models. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 923–938, 2025.

Yutong Cheng, Haifeng Chen, Wenchao Yu, Xujiang Zhao, Peng Gao, and Wei Cheng. Escaping whack-a-mole: Optimizing documentation as repo-specific playbooks for coding agents. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026a.

Yutong Cheng, Yang Liu, Changze Li, Dawn Song, and Peng Gao. CTIConnect: A benchmark for retrieval-augmented LLMs over heterogeneous cyber threat intelligence. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '26)*. ACM, 2026b. doi: 10.1145/3770855.3817527.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv:2504.19413*, 2025.

Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys*, 53(6), 2020.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *SIGIR*, 2009.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv:2404.16130*, 2024.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. *arXiv:2410.05779*, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *NeurIPS*, pp. 59532–59569, 2024.

Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. Linking threat tactics, techniques, and patterns with defensive weaknesses, vulnerabilities and affected platform configurations for cyber hunting (BRON). *arXiv:2010.00533*, 2021.

Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources. In *ACSAC*, pp. 103–115, 2017.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9237–9251, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, pp. 9459–9474, 2020.

Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. LLM for data management. *Proceedings of the VLDB Endowment*, 17(12):4213–4216, 2024. doi: 10.14778/3685800.3685838.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. WebThinker: Empowering large reasoning models with deep research capability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. AttacKG: Constructing technique knowledge graph from cyber threat intelligence reports. In *ESORICS*, pp. 589–609, 2022.

Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *ACM CCS*, 2016.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving LLM agents. *arXiv preprint arXiv:2605.30621*, 2026.

Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. Towards accurate and efficient document analytics with large language models. *arXiv preprint arXiv:2405.04674*, 2024.

Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, and Gerardo Vitagliano. Palimpzest: Optimizing AI-powered analytics with declarative query processing. In *Conference on Innovative Data Systems Research (CIDR)*, 2025.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. Autoharness: Improving LLM agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026.

OASIS. STIX version 2.1: Structured threat information expression. OASIS Standard, 2021.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv:2310.08560*, 2023.

Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. Semantic operators and their optimization: Enabling LLM-based data processing with accuracy guarantees in LOTUS. *Proceedings of the VLDB Endowment*, 18(11):4171–4184, 2025. doi: 10.14778/3749646.3749685.

Rizky Ramadhana Putra, Raihan Sultan Pasha Basuki, Yutong Cheng, and Peng Gao. NL2Logic: AST-guided translation of natural language into first-order logic with large language models. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6035–6051. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-eacl.317. URL http://dx.doi.org/10.18653/v1/2026.findings-eacl.317.

Priyanka Ranade, Aritran Piplai, Anupam Joshi, and Tim Finin. CyBERT: Contextualized embeddings for the cybersecurity domain. In *IEEE International Conference on Big Data (Big Data)*, pp. 3334–3342, 2021. doi: 10.1109/BigData52589.2021.9671824.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv:2501.13956*, 2025.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

Matthew Russo, Sivaprasad Sudhir, Gerardo Vitagliano, Chunwei Liu, Tim Kraska, Samuel Madden, and Michael Cafarella. Abacus: A cost-based optimizer for semantic operator systems. *arXiv preprint arXiv:2505.14661*, 2025.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *ICLR*, 2024.

Kiavash Satvat, Rigel Gjomemo, and V. N. Venkatakrishnan. EXTRACTOR: Extracting attack behavior from threat reports. In *IEEE EuroS&P*, pp. 598–615, 2021.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. DocETL: Agentic query rewriting and evaluation for complex document processing. *Proceedings of the VLDB Endowment*, 18(9):3035–3048, 2025. doi: 10.14778/3746405.3746426.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv:2501.09136*, 2025.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Blake E. Strom, Andy Applebaum, Doug P. Miller, Kathryn C. Nickels, Adam G. Pennington, and Cody B. Thomas. MITRE ATT&CK: Design and philosophy. Technical Report MTR170302, The MITRE Corporation, 2018.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the search capability of LLMs without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations (ICLR)*, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv:2305.16291*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. Fonduer: Knowledge base construction from richly formatted data. In *SIGMOD*, pp. 1301–1316, 2018.

John Yang and mini-swe-agent contributors. mini-swe-agent: a minimal, standard agent harness. https://github.com/SWE-agent/mini-swe-agent, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *NeurIPS*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

Ce Zhang, Christopher Ré, Michael Cafarella, Christopher De Sa, Alex Ratner, Jaeho Shin, Feiran Wang, and Sen Wu. DeepDive: Declarative knowledge base construction. *Communications of the ACM*, 60(5):93–102, 2017.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.

Ziyun Zhu and Tudor Dumitras. ChainSmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In *IEEE EuroS&P*, pp. 458–472, 2018.

Appendix Contents

A The Build Pipeline in Detail **16**

A.1 Report Layer: the Four Stages 16

A.2 Embedding Model and Indexes 16

A.3 Deterministic-First Entity Resolution 17

B The CTI Ecosystem **17**

C CTIConnect's Measured Failure Diagnostics **17**

D The Seven Typed Tools **18**

E Additional Experimental Results **18**

E.1 RQ1: Extraction Quality Across Operator Models 18

E.2 Scalability at $1.7\times$ the Question Volume 19

E.3 RQ4: Scalability 19

E.4 A Qualitative Case Study 19

E.5 RQ6: A Qualitative Case Study 20

F Extended Related Work **20**

F.1 CTI Knowledge Extraction and Representation 20

F.2 From Retrieval Pipelines to Agentic Search 21

F.3 LLM-Augmented Data Management and Knowledge-Base Construction 21

F.4 Agent Interfaces, Skills, and Context Engineering 22

F.5 CTI and Security Benchmarks on LLM 22

G Prompt Templates **24**

G.1 Agent System Prompts: the Two Arms 24

G.2 Build Time: Span-Grounded Extraction 24

G.3 Build Time: TTP Extraction 25

G.4 Build Time: Entity-Resolution Adjudication 25

G.5 Build Time: Triple Validation 26

G.6 Evaluation: Multi-Document Synthesis Judge 26

H Procedural Skill Playbooks **28**

H.1 Entity Linking (RCM, ATD, ESD) 28

H.2 Entity Linking, Reverse Direction (WIM) 29

H.3 Attribution to ATT&CK Techniques (ATA) 30

H.4 Attribution to CWE Weaknesses (VCA) 30

H.5 Multi-Document Synthesis (CSC, TAP, MLA) 31

Table 4: The scaffold at a glance: what the build materializes from the released corpus, and the invariants the validator enforces. Every identifier is checked against the pinned snapshots at write time, so zero fabrication holds by construction rather than by post-hoc filtering.

Ontology nodes		**Typed edges**		**Report layer, cost, validation**	
CVE	3,011	`has_weakness`	3,290	Vendor reports	321
CWE	1,342	`exploits_weakness`	1,214	Span-grounded chunks	688
CAPEC	615	`in_tactic`	1,076	Canonical entities	4,868
ATT&CK	1,076	`child_of`	533	Build tokens (one-time)	1.42M
		`sub_technique_of`	518	Build cost (one-time)	$1.86
		`maps_to_technique`	272	Fabricated identifiers	**0**
		CAPEC ordering	194	Orphan edges	**0**
Total	*6,044*	*Total*	*7,097*	Span violations	**0**

A THE BUILD PIPELINE IN DETAIL

§3.2 states the rule the build follows and the invariants it guarantees. This appendix gives the pipeline itself. It runs once over the 321 vendor reports with a fixed build model (`gpt-5.4-mini`), and is then frozen, so every query-time configuration in §4 consumes byte-identical artifacts. The operator prompts are reproduced in Appendix G.

A.1 REPORT LAYER: THE FOUR STAGES

Step 0: semantic chunking. Reports are segmented at clause boundaries and packed into chunks of 200–900 characters, never splitting mid-clause; the corpus yields 688 chunks. Each chunk records exact character offsets into the frozen source text, making the chunk the unit of both content and provenance (C2). The granularity is chosen to be navigable in both directions: a single clause is too small to ground a synthesis answer and a whole report too coarse to pin a specific fact, while a few-clause chunk resolves up to its document and down to its spans.

Step 1: typed entity mentions. Per chunk, an LLM extracts typed entity mentions plus within-chunk coreference links, which feed cross-vendor resolution downstream. The type system is a deliberately small, STIX-aligned set of eight types (`threat_actor`, `malware`, `tool`, `technique`, `vulnerability`, `campaign`, `identity`, `indicator`), chosen for extraction precision: mutually distinct, clearly realized on the surface, retrieval-relevant. Identifier-bearing mentions (CVE ids, ATT&CK T-ids, hashes, IPs) are captured by *deterministic regex guards* rather than trusted to the LLM, the classes where fabrication is cheapest to prevent are prevented outright.

Step 2: TTP grounding. A single-pass extractor maps each chunk's behavioral statements to ATT&CK technique ids, scaffolded by the fourteen tactics, with explicit exclusion of defender-side behavior and worked examples for canonically under-extracted techniques. Every emitted T-id is validated against the ATT&CK snapshot.

Step 3: entity resolution. Canonical entities are formed by a union-find over four equivalence signals, ordered so that deterministic evidence dominates: (a) equality of grounded external identifiers; (b) TTP groundings (natural-language technique surfaces that ground to the same T-id); (c) normalized surface-form equality within a type; (d) within-chunk coreference aliases from Step 1 (Algorithm 1). The output is 4,868 canonical entities, each carrying its vendor-attributed alias set and grounded external id where one exists, plus a bidirectional chunk↔entity index. That index is the load-bearing artifact of the layer: an entity's chunks are reachable *with their full text* (the index-to-content link), and a chunk's entities are its retrieval keys. Merge discipline mirrors the guard philosophy above: only exact and span-verified alias evidence enters the union, while weak "possibly-same" verdicts are retained as soft links outside the transitive closure, the discipline that keeps distinct state actors from collapsing into mega-clusters, and the source of the resolution margin measured in §4.2.

A.2 EMBEDDING MODEL AND INDEXES

Paraphrase matching is a commodity, so CTIFoundry delegates it to an off-the-shelf embedding model (`text-embedding-3-large`, disk-cached), serving a chunk-level index over the report corpus and one index per knowledge base.

Algorithm 1 RESOLVEENTITIES: deterministic-first union of mentions.

Require: mentions $\mathcal{M}$ (typed, with grounded ids and coref links); TTP groundings $\mathcal{T}$
Ensure: canonical entities $\mathcal{E}$; entity→chunk index Π

$U \leftarrow$ UNIONFIND($\mathcal{M}$)
for mentions $a, b \in \mathcal{M}$ of the same type **do**
  if $\text{extid}(a) = \text{extid}(b) \neq \perp$ **then** ▷ (a) external-id equality
    U.UNION(a, b)
  else if $\mathcal{T}(a) = \mathcal{T}(b) \neq \perp$ **then** ▷ (b) same grounded T-id
    U.UNION(a, b)
  else if NORM(a) = NORM(b) **then** ▷ (c) normalized surface form
    U.UNION(a, b)
  end if
end for
for coref link $(a, b) \in \mathcal{M}$ **do** ▷ (d) within-chunk coreference
  U.UNION(a, b)
end for
$\mathcal{E} \leftarrow \varnothing$; $\Pi \leftarrow \varnothing$
for cluster $c \in U$.SETS() **do**
  $\varepsilon \leftarrow \langle \text{aliases}(c),\ \text{vendors}(c),\ g{=}\text{GROUNDEDID}(c) \rangle$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{\varepsilon\}$; $\Pi[\varepsilon] \leftarrow \{\, \text{chunk}(m) : m \in c \,\}$
end for
invariant: only exact/span-verified evidence enters U; weak "possibly-same" verdicts are kept as soft links, never unioned
return $\mathcal{E}, \Pi$

A.3 DETERMINISTIC-FIRST ENTITY RESOLUTION

Algorithm 1 gives the merge procedure summarized in §3.2. Its discipline is the invariant on the last line: only exact and span-verified alias evidence enters the union, and weak "possibly-same" verdicts are retained as soft links the agent can see but the build never merges on. That is what rules out the transitive mega-cluster failure mode quantified in §4.2.

B THE CTI ECOSYSTEM

§2 compresses this to a paragraph. The two source types in full:

Authoritative knowledge bases. Four community-maintained taxonomies form the reference backbone of the field: CVE (specific vulnerability instances), CWE (weakness classes), CAPEC (attack patterns), and MITRE ATT&CK (adversary techniques, organized under fourteen tactics) (Strom et al., 2018). Crucially, these are not four independent lists: the bases *officially cross-reference* one another. A CVE record names the CWE weakness it instantiates; a CAPEC pattern lists the CWE weaknesses it exploits and the ATT&CK techniques it maps to; techniques nest under parent techniques and tactics. These cross-references are curated by the taxonomy maintainers and are, for a large class of analyst questions, *the* authoritative answer: "which weakness underlies this vulnerability" is not a matter of textual similarity but a recorded edge.

Vendor threat reports. The narrative layer is written by security vendors: incident write-ups, actor profiles, malware analyses. Reports are prose, and their central entities (threat actors, malware families, campaigns) carry *vendor-specific naming*: each vendor maintains its own nomenclature, and the same actor routinely has three or more names across the reporting landscape. Intelligence about one campaign is therefore sharded across reports that do not share surface vocabulary.

C CTICONNECT'S MEASURED FAILURE DIAGNOSTICS

§2 summarizes the benchmark's published diagnostics in three sentences and derives C1–C4 from them. This appendix reproduces the account in full, since C1–C4 are motivated by these measurements rather than by argument.

CTIConnect (Cheng et al., 2026b) operationalizes the workflow of §2 as a public benchmark, and is to date the only CTI benchmark that evaluates LLMs with retrieval access to the domain's knowledge

sources rather than closed-book: 1,859 expert-verified questions over a released corpus of the four knowledge bases (3,011 CVE, 1,342 CWE, 615 CAPEC, 1,076 ATT&CK entries) plus 321 multi-vendor report summaries, organized into nine tasks in three families. *Entity linking* (EL: RCM, WIM, ATD, ESD) maps a behavioral description across taxonomies; *entity attribution* (EA: ATA, VCA) grounds report narratives to the sets of ATT&CK techniques or CWE weaknesses they describe; *multi-document synthesis* (MDS: CSC, TAP, MLA) assembles campaign summaries, actor profiles, and malware lineages across report clusters. EL and EA are scored by identifier-normalized F1, MDS by a claim-level LLM judge. The benchmark's evaluation, however, is confined to the *RAG setting*, fixed retrieve-then-generate pipelines over chunk-and-embed indexes, whereas the progress surveyed in §1 has since made the *agentic* setting, multi-step tool-mediated investigation at query time, the operationally dominant way LLMs consume CTI in industry; the benchmark authors themselves name agentic design over the corpus as the open direction.

What the benchmark's published diagnostics (Cheng et al., 2026b) do establish, and what this paper builds on, is a *measured* account of where LLM-over-CTI fails. They quantify a cross-source semantic gap, the difference between a query's embedding similarity to its gold evidence and to its top-retrieved candidate, that widens systematically with the heterogeneity a task must bridge (0.06 within taxonomy vocabulary, 0.31 from narrative to taxonomy terminology, 0.43 across alias-sharded vendor reports), sinking gold evidence from mean rank 4.2 to 6.5 to 9.2, the latter two beyond the typical top-k window. They isolate the mechanisms: *aliasing* (reports naming one actor under different vendor names share no surface vocabulary, so near-miss distractors outscore the gold cluster; *register mismatch*) reports describe behavior in action-oriented prose while taxonomies encode it in technique-oriented terminology, so embeddings miss links that an official cross-reference already records; and *sibling confusion*, among lexically adjacent taxonomy entries, retrieval surfaces candidates the model then fails to discriminate, and in attribution every incorrectly retrieved entry becomes a wrong answer element outright, at times dragging retrieval-augmented accuracy *below* the closed-book baseline. They further establish that these failures are structural rather than incidental: general-purpose retrieval upgrades (retrieve-then-rerank, iterative retrieval) recover only a small fraction of the gap that interventions on vocabulary and entity structure recover. These documented gaps, joined by two demands operational CTI adds on top of any benchmark, namely that every claim be auditable back to the vendor and sentence that asserted it, and that the analyst's procedural discipline (resolve names before searching, trust a recorded edge over textual similarity, verify every candidate) is written down nowhere in the corpus, translate into four challenges that an agent-facing substrate must meet, each answered by one CTIFoundry component:

D THE SEVEN TYPED TOOLS

§3.3 states the three rules the action surface is designed under, non-overlap, self-description, and structure before similarity. Table 5 is the surface itself: each tool's capability and the cost class shipped to the agent in its description. The exact tool descriptions the agent sees are reproduced in Appendix G.1.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 RQ1: EXTRACTION QUALITY ACROSS OPERATOR MODELS

Extraction quality across operators. Table 1 varies the extraction operator over six models under two metrics: typed-mention extraction against a stratified human-audited gold set (∼60 chunks, all eight types; a type error is both a miss and a false positive), and TTP grounding on groundable gold (chunks stating a behavior with an explicit T-id, stripped before extraction) augmented by ∼40 vague or defender-side phrases whose correct output is abstention. Flagships reach F1 0.859 / 0.792 (entity / TTP), and small operators trail by 10–20 points: but *almost entirely in recall*: they under-extract rather than invent, the one degradation mode a build pipeline can absorb, since a missing mention costs coverage while a fabricated one would breach the invariant the whole scaffold rests on.

Table 5: The seven typed tools of the CTIFoundry action surface, each exposing one non-overlapping scaffold capability with its usage guidance and cost shipped to the agent.

Tool	Capability	Cost
`resolve_entity`	name/alias/id → canonical entities with cross-vendor alias lists and grounded external ids; prescribed first call	cheap
`get_entity`	full record of one ontology node or report entity	cheap
`ontology_neighbors`	traverse official cross-reference edges (`has_weakness`, `exploits_weakness`, `maps_to_technique`, ...); deterministic	cheap
`search_kb`	hybrid dense+BM25 search over one KB, with `must_terms` conjunctive filter and pool-size feedback	moderate
`search_chunks`	dense search over report chunks; returns text with the entities/TTPs that index it (pivot into the graph)	moderate
`chunks_mentioning`	all chunks of every report where an entity appears, with full text, the cross-vendor collection primitive	cheap
`read_report`	full text of one report by document id	cheap

Table 6: CTIFoundry accuracy on the scale set (`gpt-5.4`); the base arm is not re-run at scale.

Family	Task	n	CTIFoundry F1
EL	RCM	190	0.988
	WIM	208	0.702
	ATD	161	0.957
	ESD	180	0.834
EA	ATA	100	0.586
	VCA	129	0.500
MDS	CSC	60	0.917
	TAP	80	0.826
	MLA	60	0.848

E.2 Scalability at 1.7× the Question Volume

Dropped from the main text for length; §4.4 states the result in one sentence. The controlled comparison is established on the main set, so at scale we measure CTIFoundry only, asking whether accuracy or cost degrades at 1.7× the question volume.

E.3 RQ4: Scalability

At 1.7× the question volume neither property degrades (Table 6 and Figure 2a; discussion in Appendix E.2). Cost stays linear, the same ≈2.6 cents and ≈7 seconds per investigation as on the main set, as a build-once substrate should be. Accuracy holds: forward EL persists at the main-set ceiling (RCM 0.988, ATD 0.957, ESD 0.834), reverse linking holds once the playbook routes it through the authoritative edge (WIM 0.702 vs. 0.723), and synthesis is the *strongest* family at scale (CSC 0.917, TAP 0.826, MLA 0.848), volume stresses retrieval recall, which is exactly what the entity index supplies. Attribution is drawn harder at scale and still lands at or above what the flat substrate reached on the main set's easier attribution questions.

E.4 A Qualitative Case Study

Dropped from the main text for length; §4.4 states the two findings it contributes. Figure 2b in the main text aggregates the tool-call profile this trace exemplifies.

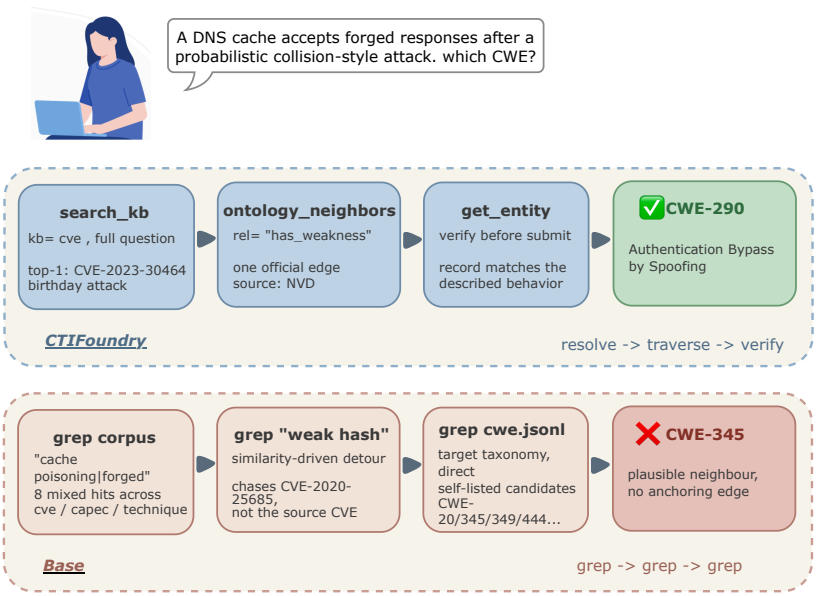


Figure 4: Both arms on entity-linking item `rcm-005` (`gpt-5.4`). Four calls and ≈8 seconds on either arm: the gap is direction, not effort.

E.5 RQ6: A Qualitative Case Study

Figure 2b gives the per-tool call profile, and Appendix E.4 traces one item, `rcm-005`, on both arms (Figure 4). Two findings carry into the argument. First, the base agent does not fail to *find* the source CVE, its very first `grep` surfaces it, it fails because the flat substrate offers no operation for *using* it, and the wrong answer it commits to lies on a real authoritative edge from a different source entity. Semantic similarity to the target therefore carries no information about which official edge a question was generated from, which is what the skill's first rule encodes. Second, the aggregate profile shows the design's intended plans are the plans the agent actually runs: forward EL is a tight `search_kb`→`ontology_neighbors`→`get_entity` spine at compliance 1.00, while MDS leans on `read_report` and the entity index and *never touches the ontology tools*, the agent uses structure where it exists and ignores it where it does not, unprompted by any router.

F Extended Related Work

§5 states this paper's position against each neighboring line in compressed form. This appendix gives the same five comparisons at length, with the positioning arguments spelled out.

F.1 CTI Knowledge Extraction and Representation

Turning threat reports into structure has evolved through three generations. The first targeted *indicators of compromise*: systems such as iACE (Liao et al., 2016) mined IPs, hashes, and domains from open-source reporting, shallow artifacts with no behavioral semantics. The second generation lifted extraction to *behavior*: TTPDrill (Husari et al., 2017) mapped report sentences to attack techniques, ChainSmith (Zhu & Dumitras, 2018) learned campaign-stage semantics, EXTRACTOR (Satvat et al., 2021) distilled attack behavior graphs from prose, AttacKG (Li et al., 2022) assembled technique-level attack graphs, and LADDER (Alam et al., 2023) extracted attack patterns beyond IoCs, each a bespoke NLP pipeline with its own schema, and each brittle in the way supervised pipelines over adversarial prose tend to be. The third generation replaced the pipelines with LLMs: CTINexus (Cheng et al., 2025) showed that optimized in-context learning constructs CTI knowledge graphs with minimal supervision, largely closing the extraction-quality question.

This line treats extraction as the endpoint: the product is a graph for human inspection or a downstream classifier. What it leaves open is the question this paper starts from: *what shape must extracted structure take to be consumed by an investigating agent?* Our answer is deliberately

subtractive: the CTIFoundry report layer extracts typed entities and groundings but no relational triples (§3.2), because its consumer reads provenance text rather than reasoning over extracted edges. CTIFoundry is thus complementary to this line: it takes extraction-quality as a solved input (RQ1 quantifies the residual model-dependence), and contributes the consumption-side design.

A parallel representational line connects the authoritative taxonomies themselves (CVE, CWE, CAPEC, ATT&CK (Strom et al., 2018), exchanged under standards such as STIX (OASIS, 2021)) into unified graphs, with BRON (Hemberg et al., 2021) linking tactics through vulnerabilities into one bidirectional graph for offline threat hunting, and follow-on work densifying its mappings. These graphs are *static analytic assets*: consumed by human queries outside any retrieval or generation loop, with no integrity guarantee on what enters them. CTIFoundry's ontology layer is materially the same data, the contribution is its *operationalization* as an agent action surface: a deterministic rebuild from pinned snapshots under a zero-fabrication invariant (§3.2), a traversal tool whose self-description teaches the agent to prefer recorded edges over text similarity, and the canonical-entity bridge into the report layer that the static-graph line does not provide.

F.2 From Retrieval Pipelines to Agentic Search

Classic RAG retrieves top-k chunks by embedding similarity and generates once (Lewis et al., 2020); its failures on structure-heavy corpora drew two pipeline-side responses: interposing derived structure between corpus and query (GraphRAG (Edge et al., 2024), LightRAG (Guo et al., 2024), RAPTOR (Sarthi et al., 2024), HippoRAG (Gutiérrez et al., 2024)), and fusing lexical with dense evidence (Robertson & Zaragoza, 2009; Cormack et al., 2009), which CTIFoundry adopts for its KB surface (§3.2). The field has since moved the retrieval decision itself into an LLM loop (Singh et al., 2025), and the current frontier is *deep research*: agents that interleave reasoning with multi-round search over an external environment, increasingly trained end-to-end with reinforcement learning, Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025) learn when and what to query, ZeroSearch (Sun et al., 2025) trains the capability without a live engine, DeepResearcher (Zheng et al., 2025) scales the loop to the open web, and WebThinker (Li et al., 2025) couples search with report drafting; BrowseComp-Plus (Chen et al., 2026b) pins such agents to a fixed corpus so that retrieval choices become comparable, the same control our methodology imposes. A neighboring line has agents traverse *existing* knowledge graphs at query time (StructGPT (Jiang et al., 2023), Think-on-Graph (Sun et al., 2024)), assuming a curated open-domain graph as given.

As attention has concentrated on agentic search, the capability being built has begun to turn on its own components: the same reason–act competence that lets an agent plan a multi-round investigation also lets it inspect, diagnose, and rewrite the machinery conducting that investigation. This is the *self-evolving agent*, a frozen model that improves by editing the scaffolding around itself, up to and including authoring new tools for its own use. Meta-Harness searches over harness code with an agentic proposer reading prior traces off a filesystem (Lee et al., 2026); Self-Harness closes a mine–propose–validate loop on model-specific failure patterns (Zhang et al., 2026); AutoHarness has the model synthesize its harness as code against environment feedback (Lou et al., 2026); and follow-on work asks which capability the loop actually requires (Lin et al., 2026; Chen et al., 2026a). The searcher thus optimizes itself: but what it searches remains outside the loop. The editable surface throughout is the harness (prompts, skills, memory, control logic, and the tools' code), while the substrate underneath is still whatever the corpus was packaged as: a generic search API, a single retrieve tool, or an already-built graph. A tool the agent writes for itself can only compose operations the substrate already affords; it cannot author an edge the corpus never materialized. CTIFoundry therefore moves one layer down and holds the harness fixed (a stock loop, no training, no evolution), rebuilding instead what the harness acts *on*: typed traversal over validated official edges, alias resolution, hybrid search with an explicit iteration signal, and entity-indexed collection. On this corpus the resulting seven-tool surface is self-sufficient (§4.3); letting the scaffold and its action surface evolve themselves is a natural extension we leave to future work.

F.3 LLM-Augmented Data Management and Knowledge-Base Construction

The database community's own answer to unstructured corpora is to move LLM operators inside the data processing loop. Semantic operators supply the declarative formalism, with per-operator optimization under accuracy guarantees in LOTUS (Patel et al., 2025); Palimpzest (Liu et al., 2025)

and Abacus (Russo et al., 2025) cast plan selection as cost–quality optimization; DocETL (Shankar et al., 2025) rewrites and validates document pipelines agentically; ZenDB (Lin et al., 2024) builds semantic indexes for document analytics; and LLM agents for data-management tasks are emerging as an architecture of their own (Li et al., 2024). The discipline these systems inherit is older, and is the one CTIFoundry's build inherits directly: knowledge-base construction from dark data, where DeepDive (Zhang et al., 2017) and Fonduer (Wu et al., 2018) established that reliability comes from declarative structure, provenance, and validation rather than from any single extractor, with entity resolution as its classical core (Christophides et al., 2020). Running through both generations is one requirement: a text-to-structure operator is trustworthy only when its output is checkable against a grammar or schema rather than accepted on the model's word. NL2Logic (Putra et al., 2026) makes this explicit in the translation setting, using the target formalism's abstract syntax tree to steer an LLM into first-order logic that is well-formed by construction.

CTIFoundry is this discipline pointed at a new consumer. Semantic-operator systems execute per query, re-optimizing each user pipeline for cost and accuracy; CTIFoundry runs once, offline, and its product is not a query answer but a substrate: span-grounded provenance on every assertion, deterministic guards wherever determinism is available (identifier regexes, snapshot validation), deterministic-first entity resolution (Alg. 1), and a blocking validator enforcing the zero-fabrication invariant (§3.2). The consumer shift is what changes the design: classical KBC targets a schema a human analyst or downstream classifier will query, so its output is optimized for relational completeness, whereas CTIFoundry's output schema is dictated by what an autonomous investigator can traverse and verify, which is why the report layer extracts typed entities and groundings but deliberately no relational triples (§3.2). The nearest LLM-era stores structure *conversational memory* (MemGPT's paged context (Packer et al., 2023), Zep's temporal knowledge graph (Rasmussen et al., 2025), Mem0's long-term store (Chhikara et al., 2025)) whereas CTIFoundry structures a *domain corpus* anchored to external authoritative taxonomies, where fabrication is definable and measurable.

F.4 Agent Interfaces, Skills, and Context Engineering

The reason–act loop (Yao et al., 2023) and learned tool invocation (Schick et al., 2023) have hardened into engineering practice: MCP standardizes tool protocols (Anthropic, 2024), and tool and context design now carry their own guidance literature (Anthropic, 2025d;a). The result our methodology leans on is SWE-agent's: a purpose-built *agent–computer interface* over a code repository outperforms a raw shell at fixed model (Yang et al., 2024), and its minimal successor mini-swe-agent (Yang & mini-swe-agent contributors, 2025) reduces the harness to the commodity our controlled design requires (§4.1). On the procedural side, skills have become first-class deployable artifacts: Agent Skills package procedural knowledge as files an agent loads on demand (Anthropic, 2025c), Agent Workflow Memory induces reusable workflows from an agent's own trajectories (Wang et al., 2024), and agentic context engineering evolves the context itself as an updatable playbook (Zhang et al., 2025; Cheng et al., 2026a), maturing the direction opened by Voyager's skill library (Wang et al., 2023) and Reflexion's verbal feedback (Shinn et al., 2023).

Both threads take the environment as given: interface work targets the computer layer, and skill work is task-generic or induced against whatever tools exist. CTIFoundry binds both to a corpus. The interface *is* a data substrate, built offline under validated invariants; the skills are corpus-bound: their central prescriptions (resolve, traverse an official edge, consult an alias set) name actions that exist only because the build materialized them. The 2×2 design of §4.5 turns this binding from a design intuition into a measured result: skills alone recover $+0.062$, tools alone $+0.136$, and together $+0.219$, procedural advice binds only to structure that exists, a corpus-side controlled question the harness literature has not posed. The same experiments contribute a deployment finding for skill engineering: identical skill text is under-followed in the system prompt by smaller models yet followed reliably in the user turn.

F.5 CTI and Security Benchmarks on LLM

Evaluation of LLMs on CTI has progressed from representation quality (CyBERT (Ranade et al., 2021)) to knowledge-and-reasoning probes (CTIBench (Alam et al., 2024)), which test what a model knows without grounding it in a corpus. CTIConnect (Cheng et al., 2026b) is the setting closest to ours and the one we adopt: corpus-grounded, expert-verified tasks spanning entity linking, attribution,

and multi-document synthesis, together with the measurement that chunk-and-embed RAG stalls exactly on the cross-source tasks, and an explicit call for agentic harness design as future work. We answer that call with a reframing: the binding side is not the harness but the corpus. We use the benchmark's tasks and released corpus unchanged, re-measure every baseline under our own fixed harness (no numbers are imported from prior work), and show the bottleneck it measured is a substrate property that build-time scaffolding removes.

G PROMPT TEMPLATES

This appendix reproduces, verbatim, every prompt used by CTIFoundry: the two agent system prompts that constitute the paper's sole experimental variable (§G.1), the build-time operator prompts that materialize the scaffold (§G.2–§G.5), and the judge prompt behind the MDS metric (§G.6). The five procedural skill playbooks follow in Appendix H. Placeholders in {} (Python `str.format`) and {{ }} (Jinja2) are substituted at runtime. Throughout, system prompts are shown in blue, user prompts in green, and skill playbooks in rose.

G.1 AGENT SYSTEM PROMPTS: THE TWO ARMS

These two prompts are the experiment of §4.3. Both arms run the same harness, model, step budget, and temperature; the only difference between them is which of the following is installed as the system prompt and which action surface it describes, the seven typed tools of Table 5 for CTIFoundry, a single `bash` tool over the corpus dumped to flat files for the base arm. Each prompt is deliberately short: the substrate, not the prompt, is what the paper varies.

System Prompt: CTIFoundry Arm (typed tools)

```
You are a CTI analyst. Investigate the question using the available tools, one or more tool
     calls per turn, then call submit_answer with the final answer. Resolve names to
    canonical entities first; prefer authoritative structure over inference; state exact
    identifiers (CVE-/CWE-/CAPEC-/T-ids, canonical names) explicitly.
```

System Prompt: Base Arm (bash over flat files)

```
You are a CTI analyst with a bash shell. The current directory holds a flat CTI corpus:
    reports/<doc_id>.txt (321 vendor threat reports) and kb/{cve,cwe,capec,technique,
    tactic}.jsonl (one JSON entry per line). Investigate with one bash command per turn (
    grep/cat/awk; each command runs in a fresh shell, so chain with pipes). State exact
    identifiers (CVE-/CWE-/CAPEC-/T-ids, canonical names) explicitly. When you have the
    answer, run a single command whose FIRST output line is exactly
    COMPLETE_TASK_AND_SUBMIT_FINAL_OUTPUT followed by your answer text, e.g.
  echo COMPLETE_TASK_AND_SUBMIT_FINAL_OUTPUT; echo 'The weakness is CWE-384.'
```

The user turn is identically templated in both arms; the per-task skill of Appendix H is prepended to it for the CTIFoundry arm, and the answer-format template for the MDS tasks is appended in both arms so that the two arms are scored on the same output shape.

User Turn Template: Both Arms

```
## Question
{{task}}
```

G.2 BUILD TIME: SPAN-GROUNDED EXTRACTION

The report layer's extraction operator (§3.2). The report is presented as numbered spans and every emitted surface must be a verbatim substring of the span it is attributed to, which is what makes the character-offset provenance of C2 checkable rather than merely asserted: the build validator re-locates each surface in its span and drops what it cannot find.

System Prompt: Span-Grounded Extraction

```
You are a precise CTI information-extraction engine. Output only JSON.
```

User Prompt: Span-Grounded Extraction

```
Extract a span-grounded knowledge graph from this cyber threat intelligence report.

The report is segmented into numbered spans:
{spans_block}

Emit a JSON object:
{{
  "mentions": [
    {{"id": "m1", "entity_type": "<one of {etypes}>",
      "surface": "<verbatim substring of the span>", "sent_idx": <span number>}}
  ],
```

```
  "triples": [
    {{"src": "m1", "rel": "<one of {rels}>", "dst": "m2", "sent_idx": <span number where
    this relation is asserted>}}
  ]
}}

Rules:
- "surface" MUST be copied verbatim (case included) from the span numbered sent_idx.
- COMPLETENESS MATTERS: extract EVERY named threat actor, malware family, attack tool,
    technique, vulnerability, campaign, indicator (hash/IP/domain), targeted sector/region
    /organization | including ones mentioned in passing (e.g. "recruited affiliates from
    BlackMatter, REvil and DarkSide" names three threat actors).
- A triple's relation must be explicitly asserted in its span, not inferred from co-
    occurrence.
- "alias_of" triples capture EVERY naming statement: "X (also known as Y)", "X, tracked as
    Y", "X aka Y", "the X group operates the Y ransomware" does NOT make X an alias of Y,
    but "Y (formerly X)" does. Never miss an alias statement | cross-vendor naming is
    critical downstream.
- Entities, not descriptions: surfaces should be proper names or identifiers, not generic
    phrases ("the malware", "a phishing campaign" are NOT mentions; ransom amounts are NOT
     indicators).
- Prefer specific relations; emit nothing you cannot anchor to a span.

Worked example. Span "[2] DarkGate (also known as MehCrypter), operated by the Rastafareye
    persona, exploited CVE-2024-21412 to target financial organizations in Europe" yields
    mentions m1=DarkGate/malware, m2=MehCrypter/malware, m3=Rastafareye/threat_actor, m4=
    CVE-2024-21412/vulnerability, m5=financial/sector, m6=Europe/region and triples (m2
    alias_of m1), (m3 operates m1), (m1 exploits m4), (m1 targets m5), (m1 targets m6),
    all with sent_idx=2.
```

G.3 Build Time: TTP Extraction

Grounds report chunks to ATT&CK techniques (§3.2). This operator runs without a system prompt. The rules encode the two error modes that dominate uncurated ATT&CK tagging (defensive recommendations read as attacker behavior, and outcome-phrased behaviors missed entirely) and the no-invention rule feeds the zero-fabrication invariant, since every emitted id is checked against the materialized ATT&CK node set at validation time.

User Prompt: TTP Extraction

```
Extract MITRE ATT&CK techniques from this CTI report chunk.

Chunk:
"""{chunk}"""

The 14 ATT&CK tactics (for scope): {tactics}

List every ATTACKER behavior in the chunk that corresponds to a real ATT&CK
technique. For each: "behavior" = the verbatim phrase; "technique_id" = the
ATT&CK id (e.g. T1486 or T1059.001), most specific you are confident in;
"technique_name" = official name.

STRICT rules:
- Attacker techniques ONLY. EXCLUDE defenses/mitigations/recommendations
  ("enforce MFA", "monitor logs", "apply patches", "network segmentation",
  "third-party risk management"), generic outcomes ("data theft", "extortion"),
  and bare tool/malware names.
- Include techniques stated concisely as OUTCOMES. Examples:
  "encrypts files / systems" -> T1486 Data Encrypted for Impact;
  "deletes volume shadow copies / inhibits recovery" -> T1490;
  "disables/kills security tools" -> T1562.001 Disable or Modify Tools;
  "uses valid/compromised/stolen accounts" -> T1078 Valid Accounts;
  "exploits a public-facing app / VPN / web vulnerability" -> T1190;
  "spearphishing / phishing email" -> T1566;
  "PowerShell / command-line execution" -> T1059.
- Do NOT invent ids. If unsure of the exact id, omit the behavior.

Return JSON: {{"ttps": [{{"behavior": "...", "technique_id": "T....", "technique_name":
    "..."}}]}}.
```

G.4 Build Time: Entity-Resolution Adjudication

Deterministic signals resolve most mentions; this judge adjudicates only the residue (§3.2). It is the operator behind the canonical cross-vendor entities of C1, and its strictness is the safeguard

against the mega-cluster failure discussed in §4.2: near-miss names (BlackCat vs. BlackMatter) and shared-sponsor actors (Lazarus, Kimsuky, Andariel) must stay distinct, and a span that merely lists both names is treated as evidence of difference rather than sameness. The three-valued `kind` field lets downstream merging accept only the authoritative tier.

System Prompt: Entity-Resolution Judge

```
You are a CTI entity-resolution judge. Decide whether two entity mentions from different
    threat reports refer to the same real-world entity. Output only JSON.
```

User Prompt: Entity-Resolution Judge

```
Mention A: "{sa}" (type: {ta}, vendor: {va})
  supporting span: "{ctx_a}"

Mention B: "{sb}" (type: {tb}, vendor: {vb})
  supporting span: "{ctx_b}"

Are A and B the same real-world entity? Cross-vendor naming differs (e.g. APT29 = Cozy Bear
     = Midnight Blizzard), but distinct entities often have similar names (e.g. BlackCat
    vs BlackMatter are DIFFERENT ransomware families).

Strict rules:
- A span that merely LISTS both names among several actors, or says one "overlaps with" / "
    may be confused with" / "is distinct from" the other, asserts they are DIFFERENT
    entities, not the same.
- Answer "exact" only when the names are well-established canonical aliases of one entity,
    or a span contains an explicit naming statement ("aka", "also known as", "tracked as",
     "formerly") directly connecting A and B.
- Sharing a country, sponsor, toolset, or campaign does NOT make two actors the same (
    Lazarus, Kimsuky and Andariel are all DPRK groups and all DIFFERENT).

JSON: {{"same": true/false, "kind": "exact" | "claimed-same" | "possibly-same", "reason":
    "<one sentence>"}}
- "exact": an authoritative naming relationship is evident
- "claimed-same": a span itself asserts the equivalence
- "possibly-same": contextual evidence warrants association but not certainty
```

G.5 Build Time: Triple Validation

Scores each candidate report-layer edge against the single sentence offered as its evidence (§3.2). Restricting admissible evidence to that one span is what separates an asserted relation from a co-occurrence, and the type-correction fields let the judge repair an entity typing without discarding the edge.

System Prompt: Triple-Validation Judge

```
You are a strict CTI fact-verification judge. Score whether a single sentence supports a
    single relational claim. Output only JSON.
```

User Prompt: Triple-Validation Judge

```
Claim: ({src} [{src_type}]) --{rel}--> ({dst} [{dst_type}])

The ONLY admissible evidence is this sentence from a {vendor} report:
"{span_text}"

Score on three criteria:
1. predicate explicitness | is the relation "{rel}" explicitly asserted in the sentence, or
     merely inferred from co-occurrence?
2. entity scope | are both entities within the predicate's syntactic scope in this sentence
    ?
3. CTI semantic validity | is "{rel}" type-compatible with a {src_type} and a {dst_type}?
    Are the entity types themselves correct (e.g. REvil is a threat actor AND a ransomware
    ; "recruits affiliates from X" does not mean "uses X")?

JSON: {{"score": <0.0-1.0>, "reason": "<one sentence>",
       "src_type_correction": null | "<corrected type>",
       "dst_type_correction": null | "<corrected type>"}}
```

G.6 Evaluation: Multi-Document Synthesis Judge

The MDS family (CSC, TAP, MLA) is free-form, so it is scored by the claim-coverage judge below rather than by identifier F1 (§4.1). Matching is many-to-many and semantically tolerant: the judge

decomposes both answers into atomic claims, then judges each side independently, which is what makes the metric robust to a prediction that splits or merges the reference's sentences. It is applied identically to both arms, and §4.2 reports its measured resolution (0.06), the threshold below which we decline to read an MDS gap as real.

Judge Prompt: MDS Claim Coverage

```
You are an expert Cyber Threat Intelligence (CTI) analyst acting as an
impartial judge. You will score a model's free-form answer to a
multi-document synthesis question against a reference (gold) answer, using
claim-level COVERAGE matching (v2: many-to-many, semantically tolerant).

## Method

1. Decompose the REFERENCE answer into a list of atomic claims. An atomic
   claim is a single, self-contained, verifiable statement (one fact about a
   threat actor name/alias, one TTP, one target, one tool, one date, one
   capability, etc.). Do not merge multiple facts into one claim.

2. Decompose the PREDICTION answer into atomic claims using the same rule.
   IMPORTANT: a compound prediction sentence containing several facts MUST be
   split into one claim per fact (same granularity as the reference side).

3. Coverage matching | NOT one-to-one. Judge each side independently:
   - A REFERENCE claim is COVERED if its content is expressed by ANY
     prediction claim, or jointly by SEVERAL prediction claims.
   - A PREDICTION claim is SUPPORTED if its content is expressed by ANY
     reference claim, or is part of the content of ONE reference claim.
     When two or more prediction claims together correspond to one reference
     claim, ALL of them count as supported.

4. Semantic tolerance | the following count as a MATCH:
   - paraphrase and alias equivalence (e.g. "APT29" matches "Cozy Bear";
     "spear-phishing" matches "targeted phishing emails");
   - temporal granularity and qualifier differences when the core fact
     (year, entity, event) agrees: "since 2021" ≈ "late 2021",
     "early 2024" ≈ "January 2024", "April 2025" ≈ "April 23, 2025";
   - singular/plural, word order, and one-word qualifier differences that do
     not change the identified entity, event, or time.
   Do NOT match claims about different entities, different events, or
   clearly different facts.

## Question
{{ QUESTION }}

## Reference (gold) answer
{{ REFERENCE }}

## Prediction (model) answer
{{ PREDICTION }}

## Output

Return ONLY a JSON object, no prose, no code fences:

{
  "reference_claims": ["<atomic claim>", ...],
  "prediction_claims": ["<atomic claim>", ...],
  "covered_reference_indices": [<0-based indices of covered reference claims>],
  "supported_prediction_indices": [<0-based indices of supported prediction claims>]
}
```

H PROCEDURAL SKILL PLAYBOOKS

The three procedural skills of §3.3 ship as five markdown playbooks: entity linking and attribution each carry a specialization for the subtask whose direction or output type differs from its family default, and synthesis serves all three MDS subtasks. The router is a static task-to-playbook map: `rcm`, `atd`, `esd` → entity linking; `wim` → entity linking (reverse); `ata`, `vca` → the two attribution playbooks; `csc`, `tap`, `mla` → synthesis: with no model call and no per-question adaptation. Each playbook is injected verbatim into the user turn rather than the system prompt, for the reason reported in §4.5: identical text placed in the system prompt was under-followed by the smaller models.

These are the files the ablation of §4.5 removes in the *w/o skill* arm; the *w/o tools* arm keeps them but serves them over bash, which is why their central prescriptions (resolve, traverse an official edge, consult an alias set) become inert there.

H.1 ENTITY LINKING (RCM, ATD, ESD)

Skill Playbook: `entity-linking.md`

```
# Skill: cross-taxonomy entity linking

Trigger: a behavioral description must be mapped to an entry in another CTI
taxonomy. The source entry is paraphrased but its ID is hidden.

**The single most important rule: do NOT search the TARGET taxonomy first.**
The description paraphrases one specific SOURCE entry; the authoritative
cross-reference edge from that source entry gives the answer. Searching the
target taxonomy directly falls into the cross-source vocabulary gap and picks
plausible-but-wrong entries.

Routing table | identify the question form, then execute:

| Question form | Step 1 (mandatory first call) | Step 2 |
|---|---|---|
| vulnerability description → "which CWE" | `search_kb(query=<description>, kb="cve")` | `
    ontology_neighbors(node_id=<CVE>, rel="has_weakness")` |
| weakness description → "which CVE instantiates" | keyword-probe loop over `search_kb(kb
    ="cve", must_terms=[...])` | see below | confirm the winner's CWE via `
    ontology_neighbors(node_id=<CVE>, rel="has_weakness")` matches the described weakness
    |
| attack-pattern description → "which ATT&CK technique" | `search_kb(query=<description>,
    kb="capec")` | `ontology_neighbors(node_id=<CAPEC>, rel="maps_to_technique")` |
| weakness description → "which CAPEC exploits it" | `search_kb(query=<description>, kb="
    cwe")` | `ontology_neighbors(node_id=<CWE>, rel="exploits_weakness", direction="in")`
    |

Keyword-probe loop (weakness→CVE): the question paraphrases the target CVE's
description, so they share rare discriminative vocabulary. Run TWO probes and
cross-check | never trust a single retrieval path:
(1) Path A: `search_kb(kb="cve", query=<question>)` with NO must_terms | note
    the top-5 (semantic + lexical ranking).
(2) Path B: pick the 2-3 most specific technical terms in the question
    (component names, mechanism words like "hardware random", "sandbox",
    instruction/protocol names | never generic words like system/attacker),
    then `search_kb(kb="cve", query=<question>, must_terms=[t1, t2])`.
    Iterate on `n_term_matches_in_kb`: >30 hits → add a term; 0 hits → drop
    the weakest term or swap a synonym ("randomness"→"entropy"); give up
    after 3 probe rounds and fall back to Path A's list.
(3) Path C: if your own knowledge suggests a specific CVE id for this
    description, verify it | `get_entity` on that id and check its
    description against the question. Memory is a candidate generator, never
    an answer by itself.
    Candidates appearing in MULTIPLE paths are strongest. A unique single-path
    hit is a good LEAD, not an answer | it still must pass step (4).
(4) FINAL CHECK (mandatory, per candidate): read the description and tick off
    EVERY specific detail of the question | mechanism, component, attack
    consequence. Pick the candidate matching ALL details; if none matches
    all, pick the one matching the mechanism (not the component). Recency or
    CVSS is NOT a tiebreaker. Each question is independent | never reuse the
    previous question's CVE just because the wording feels similar.
(5) Confirm via the CVE's `has_weakness` CWE edge when present (~9% of CVEs have no edge |
     a missing edge is NOT disconfirmation); NEVER answer empty | if
    unresolved after all probes, answer Path A's best mechanism match.

Step 3 | verify before answering: `get_entity` on the candidate target; its
```

```
title/description must actually match the question's behavior. If the top
source candidate's neighbors contain no plausible target, try the next source
candidate from step 1 (the right source is usually in the top 3).

Fallback only when step 2 returns nothing for all plausible sources:
`search_kb` over the TARGET taxonomy with keyphrases restated in that
framework's idiom (CWE: "Improper/Missing/Incorrect X"; ATT&CK:
"Parent: Sub-technique" noun phrases; CAPEC: attack-method verb phrases),
then verify with `get_entity`.

Answer with exactly one identifier, stated explicitly, plus one sentence of
reasoning grounded in the verified entry.
```

H.2 Entity Linking, Reverse Direction (WIM)

WIM inverts the linking direction (weakness description → instantiating CVE) and is the one EL subtask on which the scaffold does not approach ceiling. It gets its own playbook because the family rule, search the source taxonomy, inverts with it: here the source is the CWE, and the answer set is the reverse `has_weakness` edge.

Skill Playbook: `entity-linking-wim.md`

```
# Skill: weakness description → the CVE that instantiates it

Trigger: the question paraphrases one CWE's definition and asks which CVE
instantiates that weakness. The answer is always a CVE identifier.

**The single most important rule: do NOT search the CVE corpus first.**
The question paraphrases a CWE, not a CVE. CVE descriptions name products and
mechanisms, never the weakness class, so lexical overlap between the question
and the right CVE is weak and misleading. Go through the CWE.

Step 1 | locate the SOURCE CWE: `search_kb(query=<the description>, kb="cwe")`.
The description is a close paraphrase of one CWE entry, so the right entry
normally lands in the top 3. Confirm with `get_entity` that its definition
covers every element of the description before moving on.

Step 2 | walk the authoritative edge:
`ontology_neighbors(node_id=<CWE>, rel="has_weakness", direction="in")`.
This enumerates exactly the CVEs that NVD records as instances of that
weakness. It is the answer set, not a hint: every member is a defensible
answer to "which CVE instantiates this weakness". Do not discard it and go
searching the CVE corpus instead | that trades an authoritative answer for a
guess.

Step 3 | pick one from that set. Every member already satisfies the weakness,
so read the candidates' descriptions with `get_entity` and prefer the one that
also matches the question's extra specifics (affected component, attack
consequence, privilege required). With nothing to separate them, answer the
first. Recency and CVSS are NOT tiebreakers.

If step 1 yields no convincing CWE, try the next CWE candidate from the search
(the right source is usually in the top 3) before giving up on this route.

Fallback | ONLY when no plausible CWE exists or its reverse edge is empty
(some CWEs have no linked CVE). Search the CVE corpus directly:
(1) Path A: `search_kb(kb="cve", query=<question>)` with NO must_terms | note
    the top-5.
(2) Path B: pick the 2-3 most specific technical terms in the question
    (component names, mechanism words like "hardware random", "sandbox",
    instruction/protocol names | never generic words like system/attacker),
    then `search_kb(kb="cve", query=<question>, must_terms=[t1, t2])`.
    Iterate on `n_term_matches_in_kb`: >30 hits → add a term; 0 hits → drop
    the weakest term or swap a synonym ("randomness"→"entropy"); give up
    after 3 probe rounds and fall back to Path A's list.
(3) Read each candidate's description and tick off EVERY specific detail of
    the question | mechanism, component, consequence. Pick the candidate
    matching ALL details; if none matches all, pick the one matching the
    mechanism (not the component).

Answer with exactly one CVE identifier, stated explicitly, plus one sentence
of reasoning grounded in the verified entry. The answer to this question is a
CVE: a CWE or CAPEC id is an intermediate hop, never the answer | if the id
you are about to submit does not start with `CVE-`, you stopped early, so go
back to Step 2 and continue. NEVER answer empty.
```

H.3 Attribution to ATT&CK Techniques (ATA)

ATA and VCA share a decompose–restate–verify spine but differ in output cardinality and target taxonomy, so they ship separately. Both encode the scoring geometry explicitly: under symmetric identifier F1 a spurious identifier costs exactly what a miss does, hence the rule that rejected candidates must not appear anywhere in the answer text.

Skill Playbook: `attribution-ata.md`

```
# Skill: grounding a report narrative to its ATT&CK technique

Trigger: a quoted blog passage describes attacker behavior; identify the
ATT&CK technique it maps to. **The answer is exactly one T-id.**

Every question here has a single gold technique. A second plausible T-id
cannot gain you anything and strictly costs precision, so pick the best one
and drop the runner-up -- even when two feel equally good. A passage often
narrates several steps (delivery, execution, evasion); the question asks for
the technique it is *about*, the one the passage spends its detail on, not
every step you can name.

Recommended sequence:
1. **Decompose the passage into atomic behaviors** (each a single actionable
   security event). Prose interleaves several behaviors; one-to-many is common.
2. Per behavior, `search_kb(kb="mitre", top_k=10)` | the right entry often ranks 5-10, NOT
     top-3, because
   famous sibling techniques outrank precise ones. Restate the behavior in
   the taxonomy's own idiom | ATT&CK names are "Parent: Sub-technique" noun
   phrases ("Subvert Trust Controls: Mark-of-the-Web Bypass"); CWE names are
   "Improper/Missing/Incorrect X". Narrative verbs rarely match: canonicalize
   before searching.
   When you pick a parent technique, also call
   `ontology_neighbors(node_id=<T-id>, rel="sub_technique_of",
   direction="in")` and check whether a sub-technique matches the passage's
   specifics better.
3. **Validate each candidate** with `get_entity`: the entry's description must
   cover the described behavior, not merely share words. Reject co-occurrence
   matches. Sub-technique beats parent technique when the detail supports it.
4. If the passage names identifiers (CVE-...), `ontology_neighbors` from them
   (has_weakness) gives authoritative CWE anchors for free.
5. Commit to ONE technique. If two candidates survive verification, choose on
   the passage's own wording rather than listing both. If none convincingly
   matches, re-search with 2-3 alternative phrasings before settling.
   Answer any sub-questions (platforms, data sources) from the `get_entity`
   attrs | they are authoritative fields, not guesses.

Decision points:
- Behavior matches both parent and sub-technique → prefer the sub-technique if
  the passage's specifics warrant it, else the parent.
- Precision over recall: a wrong extra identifier costs exactly as much as a
  miss. Never present "related" or "additionally relevant" identifiers.
- CRITICAL OUTPUT RULE: every CWE-/T- identifier token appearing ANYWHERE in
  your final answer is scored as one of your predictions. Mention ONLY your
  chosen identifier; never name rejected candidates, comparisons, or
  alternatives in the answer text. Exactly one T-id may appear anywhere in
  the answer.
```

H.4 Attribution to CWE Weaknesses (VCA)

Skill Playbook: `attribution-vca.md`

```
# Skill: grounding a vulnerability narrative to its CWE

Trigger: a quoted blog passage describes a vulnerability or exploitation
narrative; identify the CWE weakness it maps to. **The answer is always a CWE.**

A passage about attacker behavior will always suggest plausible ATT&CK
techniques too | ignore them. Search `kb="cwe"` only, and emit only `CWE-nnn`
identifiers. A T-id in the answer is not a partial credit, it is a wrong
prediction. If the passage feels too thin to ground a CWE confidently, still
commit to the best-supported one: abstaining and answering wrongly score the
same, so a calibrated answer strictly dominates.

If the question truncates mid-sentence (e.g. ends at "Please provide: 1)"),
answer with the single CWE identifier plus one sentence of justification.

Recommended sequence:
```

```
1. **Decompose the passage into atomic behaviors** (each a single actionable
   security event). Prose interleaves several behaviors; one-to-many is common.
2. Per behavior, `search_kb(kb="cwe", top_k=10)` | the right entry often
   ranks 5-10, NOT top-3, because famous sibling weaknesses outrank precise
   ones. Restate the behavior in CWE's own idiom: entries are named
   "Improper/Missing/Incorrect X". Narrative verbs rarely match: canonicalize
   before searching.
3. **Validate each candidate** with `get_entity`: the entry's description must
   cover the described behavior, not merely share words. Reject co-occurrence
   matches.
4. If the passage names identifiers (CVE-...), `ontology_neighbors` from them
   (has_weakness) gives authoritative CWE anchors for free.
5. Consolidate to a CALIBRATED set: output the smallest set of identifiers
   that covers every described behavior. One behavior -> usually one identifier;
   but when two candidates BOTH plausibly match a behavior after verification
   and you cannot separate them on the entry text, include both. Do not include
   a third. If NO candidate convincingly matches a behavior, re-search with
   2-3 alternative phrasings (different idiom, different aspect of the
   behavior) before settling.
   Answer any sub-questions (platforms, data sources) from the `get_entity`
   attrs | they are authoritative fields, not guesses.

Decision points:
- The gold answer is usually the weakness CLASS the passage illustrates, not
  the narrowest variant you can find. When a specific CWE and its more general
  parent both fit, prefer the one the passage's own wording supports; do not
  reach for a narrower variant on detail the passage never states.
- Precision over recall: a wrong extra identifier costs exactly as much as a
  miss. Never present "related" or "additionally relevant" identifiers.
- CRITICAL OUTPUT RULE: every CWE-/T- identifier token appearing ANYWHERE in
  your final answer is scored as one of your predictions. Mention ONLY your
  chosen identifier(s); never name rejected candidates, comparisons, or
  alternatives in the answer text. Before submitting, delete every T-id from
  the answer text | including ones cited only as supporting context.
```

H.5 Multi-Document Synthesis (CSC, TAP, MLA)

One playbook serves all three MDS subtasks. It is the only skill whose prescriptions are mostly about *coverage* rather than routing, which matches the per-tool profile of Figure 2b: the MDS agent leans on `read_report` and the entity index and never touches the ontology tools.

Skill Playbook: `synthesis.md`

```
# Skill: multi-report synthesis (actor profiles, malware lineage, campaign timelines)

Trigger: a question about one threat entity whose intelligence is scattered
across several vendor reports | possibly under different names.

Recommended sequence:
1. **Cover the listed cluster first**: if the question lists a report
   cluster (ids like [BLOG-n]), fetch EVERY listed report up front |
   `read_report` on each id (bash arm: cat reports/<id>.txt). These reports
   ARE the question's scope; do not skip any of them.
2. **Resolve the anchor entity**: `resolve_entity` on the entity named in the
   question (or surface it via `search_chunks` first if only behavior is
   given). The result lists the build-aggregated cross-vendor `aliases` and
   `n_docs` | for alias/naming questions, that aliases field is the complete
   authoritative set.
3. **Collect additional content through the entity index**:
   `chunks_mentioning` on the resolved entity id lists every report where
   any alias appears (doc_id, vendor, excerpt) | `read_report` any listed
   report not yet read. Also `resolve_entity` on related names found along
   the way (variants, predecessor families). Stay on the campaign the
   question is about: drop reports that merely share an actor name but
   describe a different campaign.
4. **Per-report checklist extraction** (do NOT summarize the pile in one
   pass): first list the distinct reports you retrieved; then for EACH
   report, go through EVERY field of the required answer format and note what
   that report contributes (names, dates, capabilities, targets).
   `read_report` starts with the report's build-extracted
   `indexed_entities`/`indexed_ttps` | a handy floor for what the report
   mentions (type buckets are heuristic) | then read the full text: facts
   often sit in passages that do not name the anchor entity.
5. **Merge across reports/vendors**: union the per-report notes field by
   field; group by vendor for corroboration. For dates and timelines, use
   only dates stated in the report TEXT (activity dates, disclosure dates in
   prose) | metadata is not provided. Copy date qualifiers verbatim
```

```
   ("late 2021", "early 2024", "April 23, 2025") | do not round or reword.
6. **Synthesize with explicit alias resolution**: name the canonical entity,
   list the aliases and which vendor uses which, order events by the dates
   stated in the text, attribute claims to vendors. Note disagreements
   rather than averaging them.
7. **Answer form**: compact bullets answering EXACTLY the aspects asked
   (e.g. "canonical name / aliases resolved" or "dates / phases"). Every
   bullet must be a claim grounded in a retrieved chunk. No background
   filler, no speculation, no "additionally" padding | extra unsupported
   claims directly lower your score.

Decision points:
- "All / every / distinct X across these reports" → collect candidates from
  EVERY report read (not only those linked to the anchor entity), dedupe by
  canonical name, list the whole set.
- Timeline questions → collect date statements from the chunk texts;
  first-seen claims need the earliest stated date, not the most detailed report.
- Lineage questions → resolve each variant entity, read the chunk introducing
  it for the capability delta.
- Targeting/corroboration questions → group the chunks by vendor; a claim
  backed by multiple vendors is stronger than a single-source one.
```