TACTIC-KG: Toward Small Agent Teams for Cyber Threat Intelligence Knowledge Graph Construction

Mouhamed Amine Bouchiha[1][0000−0001−6142−6855] and Gregory Blanc[1][0000−0001−8150−6617]

SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris
mbouchiha@telecom-sudparis.eu, gregory.blanc@telecom-sudparis.eu

Abstract. Cyber Threat Intelligence (CTI) reports are predominantly unstructured, heterogeneous, and noisy, which limits their direct usability for automated analysis and reasoning. Cybersecurity Knowledge Graphs (CSKGs) provide a structured representation of adversarial entities, actions, and relations, but constructing such graphs from free-text CTI remains a challenge. Recent approaches rely on monolithic Large Language Models (LLMs) to perform end-to-end extraction and completion, leading to high cost, limited controllability, and unstable performance. This paper introduces **TACTIC-KG**, an agentic framework for CSKG construction that decomposes the task into modular, specialized LLM agents responsible for extraction, typing, verification, and curation. Using lightweight models (3B–8B), TACTIC-KG improves stability, recall, and graph consistency while reducing deployment cost. We implement and evaluate TACTIC-KG against recent state-of-the-art systems. Experiments on human-annotated CTI reports show that agent specialization consistently outperforms larger monolithic in-context-learning (ICL) baselines in extraction F1-score, typing accuracy, and structural graph similarity.

Keywords: Cyber Threat Intelligence · Knowledge Graphs · Language Models · Agentic Systems · Information Extraction.

1 Introduction

Cyber Threat Intelligence (CTI) reports contain rich textual narratives describing adversarial operations, tactics, and observable indicators. However, transforming these heterogeneous and unstructured reports into processable representations remains a fundamental challenge [7,41]. Traditional natural language processing techniques, such as named entity recognition (NER) [17] or classification-based pipelines [3], often fail to capture the nuanced and implicit relationships present in CTI, particularly for complex or fine-grained adversarial behaviors [7]. Cybersecurity or CTI Knowledge Graphs (CSKGs) have emerged as a promising solution to structure CTI into entities, relationships, and contextual knowledge, supporting tasks such as visualization, attack-path reasoning, and automated

Table 1: Comparison of cybersecurity IE and CSKG construction properties.

Property	CTINEXUS [10]	IntelEX [36]	AECR [8]	CRUcialG [9]	AttacKG [22]	TACTIC-KG
Faithfulness verification	✗	△	△	✗	✗	✓
Hallucination mitigation	△	△	✓	△	✗	✓
Over-completion control	✗	✗	✗	✗	✗	✓
Ontology compliance	✓	△	✗	✓	✓	✓
Structural consistency	✓	✗	△	△	✓	✓
Small-model compatibility	△	✓	✓	✓	✓	✓
Incomplete-evidence handling	✓	✗	✗	△	△	✓

correlation [10, 22, 41]. Recent research has leveraged Large Language Models (LLMs) to automate CSKG construction, taking advantage of their ability to process free-form text and capture latent semantic patterns. Prompt-based systems such as aCTIon [31] demonstrated the feasibility of LLM-driven document-level extraction of ATT&CK techniques, although static prompts can suffer from hallucinations and limited precision on fine-grained entities. Ontology-grounded approaches, exemplified by CTINEXUS [10] and IntelEX [36], improve semantic grounding and entity canonicalization through in-context learning (ICL) [13] and retrieval-augmented generation (RAG) [4], respectively, but at the cost of increased system complexity and sensitivity to retrieval quality.

Complementary efforts focus on fine-tuning and domain adaptation of LLMs. Systems such as AECR [8] and Fengrui et al. [15] show that supervised and synthetic fine-tuning can enhance precision and reduce hallucinations, even with smaller model sizes. Comparative studies indicate that, while LLMs generally achieve higher recall than transformer-based classifiers, precision can remain a challenge, motivating hybrid or multi-model architectures [14, 20]. Non-LLM approaches, including AttacKG+ [22], LADDER [3], and CRUcialG [9], highlight the continued relevance of structured pipelines and expert rules, although these methods can be brittle to linguistic variation.

In this work, we ask the question of whether constructing a CSKG from CTI should be treated as a *multi-agent reasoning problem*, rather than a single monolithic inference task to minimize hallucination and improve stability.

We introduce **TACTIC-KG**[1], an agentic framework that decomposes **extraction**, **typing**, **verification**, and **curation** into specialized small LLM agents. By combining multi-agent orchestration with ontology grounding, TACTIC-KG improves robustness and stability, while enabling the use of lightweight models. Unlike static or monolithic LLM approaches, our framework explicitly manages uncertainty, enforces ontology compliance to construct accurate and actionable CSKGs from unstructured CTI reports.

The remainder of this paper is structured as follows. Sect. 2 reviews related work. Sect. 3 formalizes the CSKG construction problem. Sect. 4 presents the TACTIC-KG framework. Sect. 5 describes the experimental setup and evaluation. Sect. 6 discusses insights and limitations, and Sect. 7 concludes the paper.

[1] Code and datasets: https://github.com/mohaminemed/TACTIC-KG.

2 Related work

This section reviews relevant studies on cybersecurity information extraction (IE *e.g.*, TTPs) and CSKGs construction from unstructured CTI reports. Table 1 compares TACTIC-KG with the most recent studies.

Ontology-Grounded & Prompting LLM Systems. aCTIon [31] represents one of the earliest LLM-based systems, relying on GPT-3.5-Turbo with static prompts to perform document-level summarization and ATT&CK technique extraction. While flexible and easy to deploy, static prompting strategies suffer from limited precision and weak control over hallucinations, particularly for fine-grained techniques. CTINEXUS [10] introduces a three-phase pipeline for constructing CSKGs using LLMs through ICL [13]. The workflow integrates triplet extraction from CTI reports, hierarchical entity alignment, and long-distance relation inference to merge disconnected subgraphs. The system uses a k-Nearest Neighbors (kNN) retriever for dynamic ICL demo [40] selection, executes a single-inference triplet extraction prompt, and combines LLM-based typing with embedding-based merging to produce CSKG graphs aligned with the the MALOnt ontology [29]. CTINEXUS provides improved semantic grounding compared to static prompting, but notes increased system complexity and sensitivity to retrieval quality. Similarly, IntelEX [36] adopts GPT-4o-mini with RAG and a multi-agent design to improve precision. While effective for extraction, its performance remains dependent on the accuracy of the retrieved context.

Fine-Tuned and Domain-Adaptive LLMs. AECR [8] explores fine-tuning ChatGLM3-6B using supervised fine-tuning combined with a classification head. This approach outperforms larger general-purpose LLMs in precision while significantly reducing hallucinations, illustrating the benefits of domain adaptation despite model size constraints. Fengrui et al. [15] fine-tune Llama2-7B using GPT-4–augmented training data, demonstrating that synthetic supervision can partially compensate for the scarcity of expert-labeled CTI datasets. The key advantage of fine-tuning lies in its reduced dependence on prompt design, as the model learns during training what information should be extracted from the input. Fieblinger et al. [16] further explore the use of open-source LLMs (*e.g.*, Llama 2, Mistral 7B Instruct, and Zephyr) for extracting structured triplets from CTI reports and constructing KGs. Their study compares prompt engineering, guided generation, and fine-tuning, showing that fine-tuning outperforms prompt-based approaches in extraction quality. However, they also highlight persistent challenges in scaling LLM-based pipelines for large-scale KG construction and downstream tasks (*e.g.*, link prediction).

Non-LLM Recent Approaches. Beyond LLM-based extraction, AttacKG [22] proposes a multi-stage pipeline that combines TTP extraction with knowledge graph construction, forming an end-to-end CTI structuring system rather than

a pure extraction approach. In a similar vein, LADDER [3] introduces a scalable framework for extracting text-based attack patterns from CTI reports by modeling attack phases across Android and enterprise environments, and mapping them to the MITRE ATT&CK framework. CRUcialG [9] further extends this direction with a unified framework for reconstructing integrated and rational attack graphs, targeting connectivity, rationality, and universality. However, these approaches largely depend on predefined schemas, pattern matching, or manually curated expert rules. As a result, they are brittle to linguistic variation: paraphrasing, implicit relations, or diverse narrative styles in CTI reports can break rule-based matching and lead to missed or incorrect extractions. Additionally, temporal order is often inferred from textual sequence, which is unreliable since CTI reports frequently describe events in a non-chronological manner.

Cyber Threat Ontologies. A variety of ontologies have been proposed by both academia and industry to describe adversarial actions and cyber threat intelligence. Among them, STIX 2.1 [27] has been widely adopted as a standard representation format, as it provides a generic, extensible, and interoperable framework for modeling threat intelligence. As a result, many organizations and platforms rely on STIX to exchange and operationalize CTI data. Beyond this generic representation layer, several more specialized models and ontologies have been developed, often encoded using STIX 2.1. Notable examples include the MISP [25] data model for representing threat objects and galaxies, as well as MITRE frameworks such as ATT&CK [26], D3FEND, CAPEC, CWE, and CVE. These frameworks differ in scope and intent, covering defensive techniques, attack patterns, software weaknesses, and known vulnerabilities. Among these, the ATT&CK framework has emerged as the de facto standard for describing adversary Tactics, Techniques, and Procedures (TTPs). MALOnt [29] is another comprehensive ontology that defines 33 entity types (including 17 primary types and 16 sub-types) and 27 relation types, enabling rich and expressive knowledge modeling. It covers a wide spectrum of cyber threat intelligence entities (*e.g.*, *Account*, *Threat Actor*, *Event*, ...). Furthermore, MALOnt's schema is well aligned with natural language descriptions commonly found in cyber threat reports. This makes it suitable for automated knowledge extraction pipelines [10, 33].

3 Problem Definition

Let R denote an unstructured CTI report in free-form natural language. The objective of CSKG construction is to transform R into a structured graph

$$G = (V, E),$$

where V is a set of typed entities and E is a set of typed relations between entities, subject to ontology constraints $\mathcal{O}$ (*e.g.*, MALOnt [29], ATT&CK [26], STIX [27]). Formally, the construction process consists of three coupled tasks:

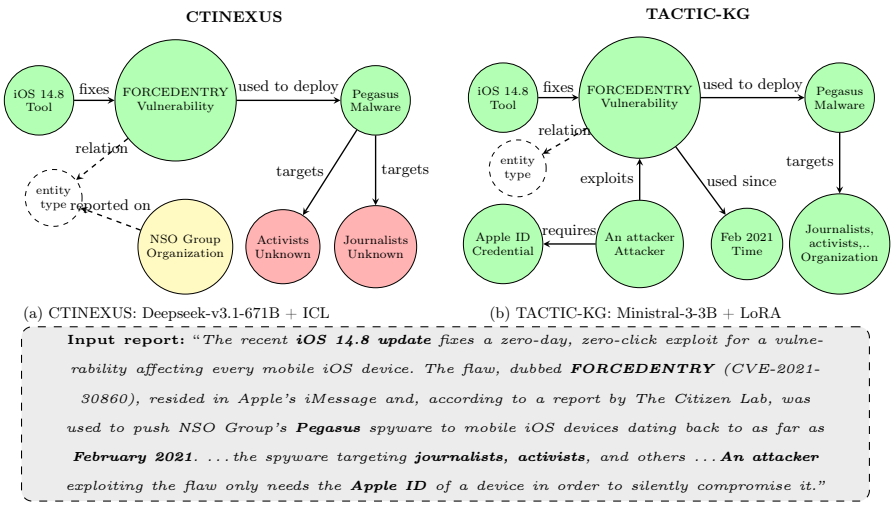

Fig. 1: CSKG construction comparison for a real-world CTI report: **Green** nodes = correctly typed; **yellow** = partially correct; **red** = missing or mis-typed. CTINEXUS misses attacker, temporal, and some entity typing; TACTIC-KG captures all entities.

1. **Extraction:** Identify candidate triples (h, r, t) such that $h, t \in V$ and $r \in \mathcal{R}$ are supported by textual evidence in R.
2. **Typing:** Assign ontology-compliant semantic types to h, r, and t.
3. **Graph Assembly:** Integrate validated triples into a consistent graph G.

While this formulation appears straightforward, constructing G from R is inherently challenging due to several interacting sources of uncertainty. As illustrated in Fig. 1, even when powered by a strong LLM such as DeepSeek-V3.1-671B, CTINEXUS [10] fails to capture key elements, including attacker roles, temporal information, and parts of target typing, whereas TACTIC-KG extracts and types all entities correctly. Moreover, as we demonstrate in Sect. 5, CTINEXUS exhibits significantly degraded CSKG quality when using smaller models. We detail the underlying sources of this uncertainty and instability in the following.

Hallucination and Faithfulness. LLMs may generate triples that are *plausible* but not *entailed* by R. We define a triple (h, r, t) as *faithful* if and only if it is semantically supported by explicit or implicit evidence in R. Otherwise, it is considered a *hallucinated triple*. Hallucinations arise from (i) *domain knowledge gaps:* the model lacks prior exposure to specific malware families, vulnerabilities, or niche TTP patterns; (ii) *insufficient context:* retrieval errors, truncated or incomplete passages (*e.g.*, naive chunking) lead to unsupported inference; or (iii) *context overload:* when excessive or irrelevant tokens dilute attention, critical evidence loses representational weight, causing entity substitutions or fabricated relations. Unlike traditional extraction errors, hallucinated triples may appear

syntactically and semantically coherent, making them difficult to detect through automatic validation. In a pipeline setting, such errors propagate and may distort the final graph structure.

Over-Completion and Artificial Connectivity. Many CSKG construction systems [9, 10] implicitly favor the production of connected graphs. However, CTI reports do not necessarily describe fully connected structures (*e.g.*, complete attack chains). Human-annotated datasets [12] frequently contain disconnected components, reflecting partial observations or incomplete forensic evidence. We therefore distinguish between (i) *evidence-grounded inference:* relations supported by textual or logically entailed evidence; and (ii) *graph over-completion:* artificially introducing edges to increase connectivity without textual support. Over-completion is a structural manifestation of hallucination: the system prioritizes graph connection over faithfulness.

Context-Size Trade-Off. Let $\mathcal{C}(R)$ denote the context window provided to the LLM. There exists a non-trivial trade-off: (i) small $\mathcal{C}(R)$ increases omission errors due to missing evidence; and (ii) large $\mathcal{C}(R)$ increases attention diffusion, reducing signal-to-noise ratio and increasing hallucination risk. This trade-off is particularly acute for lightweight models with limited capacity. While techniques such as ICL [13], RAG [4], or even dynamic context construction [39] partially mitigate this issue for large models, their effectiveness largely depends on the underlying model capacity and the retrieval quality. In practice, these techniques provide substantial gains for large, high-capacity models, but offer limited benefits for smaller models, where constrained reasoning and context hinder their impact [1, 35].

Problem Statement. Given an unstructured CTI report R and ontology constraints $\mathcal{O}$, construct a knowledge graph $G = (V, E)$ that satisfies the following constraints:

Faithfulness	$\forall (h, r, t) \in E,\ (h, r, t) \models R$
Ontology Compliance	$\text{types}(h, r, t) \in \mathcal{O}$
Minimal Over-Completion	E contains no unsupported edges
Structural Consistency	G satisfies global logical constraints

In other words, the constructed graph must remain fully grounded in the input report, respect ontology constraints, and avoid introducing spurious relations. The central research question of this work is therefore:

> *Can CSKG construction be reformulated as a controlled multi-agent reasoning process that verifies faithfulness and limits hallucination propagation, rather than relying on monolithic generation or connectivity-driven inference?*

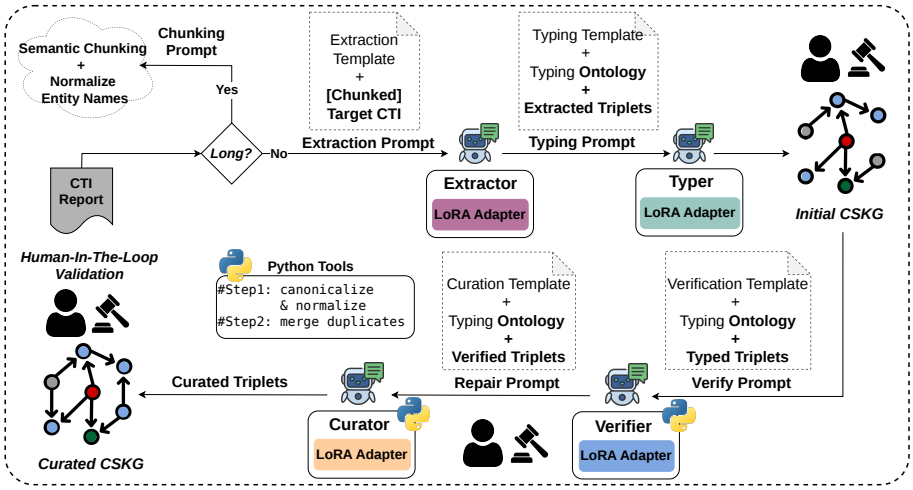

Fig. 2: TACTIC-KG framework.

4 TACTIC-KG Framework

TACTIC-KG reformulates CSKG construction as a controlled multi-stage reasoning process. Instead of relying on a single monolithic LLM to jointly extract, type, and assemble the graph, we decompose the problem into specialized agents operating under constrained objectives. The core design principle is *faithfulness before connectivity*. Each stage is responsible for a narrowly defined transformation, and agents are not allowed to introduce information not grounded in the input report. This modularization reduces hallucination propagation and enables fine-grained supervision. The decomposition yields three robustness advantages:

1. **Reduced hallucination surface:** Each agent operates within a constrained reasoning space.
2. **Error containment:** Hallucinated triples do not automatically propagate.
3. **Task entropy reduction:** Smaller models perform better when solving narrowly scoped tasks.

Empirically (Sect. 5), we observe that even strong large-scale models can hallucinate entity substitutions or unsupported relations. In monolithic pipelines, such errors propagate and distort graph structure. TACTIC-KG localizes these risks and enforces faithfulness as a first-class constraint.

4.1 High-Level Workflow

As shown in Fig. 2, given a CTI report R, the system first applies *semantic chunking* to handle long-form documents that exceed model context limits. Rather than naïve fixed-length segmentation, chunking is performed at sentence

or discourse boundaries with controlled overlap, preserving local semantic coherence while preventing context dilution. The resulting set of chunked reports $\{R_1, R_2, ..., R_n\}$ is processed sequentially through a multi-agent pipeline:

$$\texttt{Raw Report} \rightarrow \textbf{Semantic Chunking} \rightarrow \texttt{Chunked Reports} \rightarrow \textbf{Extractor} \rightarrow \textbf{Typer} \rightarrow \texttt{Initial CSKG} \rightarrow \textbf{Verifier} \rightarrow \textbf{Curator} \rightarrow \texttt{Curated CSKG}$$

A long raw CTI report is first segmented using semantic chunking to preserve discourse boundaries. Next, the agents interact sequentially under an auditable and Human-in-the-loop (HITL)-friendly protocol such that: (i) intermediate outputs are serialized in JSON format, (ii) no agent can modify upstream spans, and (iii) partial re-execution is supported. In this setting, the pipeline includes:

- **Extractor Agent:** Generates candidate triples (h, r, t) grounded in each chunk R_i. No typing or global reasoning is performed at this stage.
- **Typer Agent:** Assigns ontology-compliant types (τ_h, τ_t) to entities using local context and relation semantics.
- **Verifier Agent:** Operates at the triplet level. Removes unsupported or low-confidence triples by structural and ontology validation.
- **Curator Agent:** Operates at the document level after merging verification outputs. Introduces only logically necessary structural edges (*e.g.*, normalization links, alias resolution: "TrickBot malware and TrickBot").

4.2 Faithful Span-Level Generation: Extractor Agent

The Extractor Agent is responsible solely for identifying candidate triples (h, r, t) from raw CTI text. It does *not* perform typing or canonicalization. This separation prevents premature abstraction, which is a major source of hallucinated structure in end-to-end systems.

Confidence estimation. After generation, token-level log-likelihood scores are computed. Triple-level confidence is derived via aggregation and propagated downstream. Low-confidence triples can later be flagged by the verifier.

Context management. To mitigate context overload (Sect. 3), long reports are segmented using sentence-level chunking with controlled overlap. Triples are generated per chunk and later merged. This reduces attention dilution while preserving local semantic coherence.

4.3 Ontology-Constrained Classification: Typer Agent

Joint extraction and typing forces the Typer Agent model to reason over both span detection and ontology inference simultaneously, increasing cognitive load and hallucination risk. Decoupling reduces task entropy and allows targeted fine-tuning. The Typer Agent assigns semantic types to extracted entities using: (i) local textual context, (ii) relation semantics, (iii) and a fixed ontology schema. Formally, for each extracted triple (h, r, t), the typer predicts:

$$(h, r, t) \rightarrow (h, \tau_h, r, t, \tau_t)$$

where $\tau_h, \tau_t \in \mathcal{O}$. The typer is restricted to a predefined class set. If the evidence is insufficient, it assigns `Uncategorized`, which prevents type ontology drift.

4.4 Evidence-Grounded Triplet Validation: Verifier Agent

The Verifier Agent acts as a filtering stage between semantic typing and graph curation. Its role is to ensure that every candidate triplet is *faithfully grounded in the source text*, thereby preventing unsupported or hallucinated relations from propagating downstream. Formally, given a typed triplet $(h, \tau_h, r, t, \tau_t)$ and an input text x, the verifier performs a binary classification:

$$\mathcal{V}(h, \tau_h, r, t, \tau_t, x) \rightarrow \{\texttt{SUPPORTED}, \texttt{NOT\_SUPPORTED}\}$$

A triplet is labeled `SUPPORTED` if the relation between h and t is either explicitly stated or implicitly entailed in x. We define (i) *explicit support* as cases where the relation is directly expressed in the text via a lexical or syntactic realization of the predicate (or its paraphrase). For example, in the sentence *"The malware Emotet communicates with its C2 server at domain X"*, the triplet (Emotet, communicates_with, C2 server) is explicitly supported.
(ii) *Implicit support* refers to cases where the relation is not directly stated but can be inferred from context or domain knowledge. For example, from *"The attacker used TrickBot to deploy ransomware on the victim system"*, we infer (TrickBot, enables, ransomware deployment), even though the enabling relation is not explicitly expressed. Otherwise, the triplet is labeled `NOT_SUPPORTED`.
Evidence extraction. Beyond classification, the verifier enhances interpretability by extracting a minimal supporting sentence $e \subset x$ whenever possible, providing a verifiable trace for downstream HITL validation:

$$\mathcal{V}_e(h, r, t, x) \rightarrow e \;\text{ or }\; \varnothing$$

Strict grounding constraints. The verifier operates under a closed-world assumption with respect to the input text: (i) only the provided text x may be used as evidence, (ii) external knowledge is forbidden, (iii) plausible but unstated relations must be rejected. By explicitly enforcing textual entailment at the triplet level, the verifier filters hallucinations introduced during extraction and typing. This transforms the pipeline into a *generate-then-verify* paradigm, where unsupported structures are pruned before reaching the curator.

4.5 Structural Completion and Sanitization: Curator Agent

The Curator Agent performs the final sanitization pass before graph insertion. Its responsibilities include: (i) deduplication and canonicalization, (ii) entity normalization (case, formatting, alias merging), (iii) ontology compliance enforcement, (iv) graph-level constraint validation. The curator does not perform unconstrained generation. Instead, it operates under a strictly bounded repair objective: it may introduce new triplets only when they are supported by textual evidence. In addition to constrained completion, the curator acts as a structural firewall by filtering unsupported or low-confidence triplets propagated from upstream agents. Thus, it prevents hallucinated or inconsistent edges from entering the final graph. This dual role—verification and minimal constrained repair—preserves global graph coherence.

4.6 Parameter-Efficient Specialization via LoRA

To enable efficient specialization without full-model fine-tuning, we employ Low-Rank Adaptation (LoRA) [19] on lightweight 3B–8B parameter models. Large frontier models reduce hallucination, but incur prohibitive cost and latency [10]. Instead, we adapt smaller instruction-tuned models using LoRA to specialize them for structured JSON generation, faithful span extraction, and ontology-constrained typing. LoRA offers [30]: (i) reduced GPU memory footprint, (ii) modular agent-specific adapters, and (iii) independent upgrading of agents.

Extractor Fine-Tuning. The extractor is trained using: (i) prompt-target concatenation, (ii) masked[2] prompt tokens, (iii) chunked reports with overlap, (iv) strict JSON-only output format. LoRA adapters are injected into attention projection layers (*e.g.*, q_proj, v_proj), which allows task adaptation without modifying the base model weights.

Typer Fine-Tuning. The typer is trained under a closed label space. The ontology is embedded directly in the prompt instruction. This effectively transforms typing into constrained structured generation rather than open-ended inference.

Verifier Fine-Tuning. The verifier is trained as a triplet-level binary classifier under a strict evidence-grounding constraint. Each training instance consists of (i) the source text x, and (ii) a single candidate triplet (h, r, t). The model is optimized to generate a structured JSON output containing the label $\{\texttt{SUPPORTED}, \texttt{NOT\_SUPPORTED}\}$. To ensure precise supervision and reduce ambiguity, training is performed at the *single-triplet level*. The input is formatted as a constrained prompt embedding the text and the triplet, while the target output is the corresponding labeled JSON structure. The model is trained to rely on the provided text. This transforms verification into a textual entailment under strict grounding. Supervision is derived automatically from annotated CTI corpora. Positive samples correspond to supported triplets, while negative samples are generated from non-supported or weakly implied relations. This construction aims at improving robustness against over-acceptance bias.

Curator Fine-Tuning. The curator is fine-tuned to perform constrained graph repair. Given (i) the source text, (ii) the current graph, the model is trained to predict the *minimal set of triplets* that complete the graph. Training targets are formed automatically. For each report, explicit triplets are partitioned into components. Ground-truth bridging triplets are derived from implicit (latent) relations that connect nodes across components. Only triplets that introduce cross-component edges are used as supervision. The model is required to output valid JSON containing only new bridging triplets. Loss is masked over the prompt portion (*i.e.*, optimization applies only to the predicted repair edges). The curator is trained under a *minimality objective*: it must introduce the smallest set of structurally necessary connections without repeating existing edges or hallucinating unsupported relations. This formulation transforms graph completion from open-ended generation into constrained repair grounded in textual evidence.

[2] During training, the model sees both the input prompt and the expected output. But we do not want the model to learn to "predict" the prompt itself, only the answer.

5 Experimental Evaluation

We evaluate TACTIC-KG on human-annotated CTI reports using semantic similarity matching [21, 24]. Metrics include extraction precision, recall, F1-score, typing accuracy [10], and graph structural similarity [24]. Baselines include multiple CTINEXUS [10] configurations. We structure our analysis around a set of research questions that investigate the effectiveness of TACTIC-KG across extraction quality, typing accuracy, and graph-level consistency. Our evaluation addresses the following research questions:

- **RQ1:** Does agentic decomposition improve extraction quality?
- **RQ2:** How does agent specialization affect precision-recall (PR) trade-offs?
- **RQ3:** Does TACTIC-KG improve typing and semantic grounding and produce more structurally consistent graphs?
- **RQ4:** What is the impact of a hybrid agentic design?
- **RQ5:** How much data does TACTIC-KG need to reach monolithic model performance?

5.1 Experimental Setup

Datasets. We construct a comprehensive CTI benchmark using multiple recent human-annotated resources. In particular, we built our dataset using two complementary corpora: (i) CTI-HAL [12], a human-annotated CTI dataset, and (ii) the CTINEXUS benchmark [10], which provides fine-grained annotations for triplet extraction, entity alignment, and prediction tasks. CTI-HAL represents a large collection of CTI reports from multiple APT groups and cybercrime campaigns. The data includes detailed technical reports from well-known security vendors and threat intelligence platforms, spanning several prominent threat actors such as APT29, CARBANAK, FIN6, FIN7, OILRIG (APT34), SANDWORM, and WIZARD SPIDER. Each report contains rich unstructured descriptions of TTPs, malware families, infrastructure, and victimology. The aggregated corpus consists of over 230 human-annotated CTI reports.
Ontology. As in [10], we use MALOnt [29] for this evaluation, as it represents one of the most comprehensive open-source ontologies available. MALOnt provides fine-grained sub-typing, particularly for *Indicator* and *Malware Characteristics*, allowing for more precise semantic grounding. We use the revised version of MALOnt[3] that is aligned with STIX 2.1, which further improves its practical applicability for structured threat intelligence extraction.

Annotation and Task Design. The CTINEXUS dataset provides annotations for triplet extraction, entity typing, and inter-entity relations. We ensure consistency across datasets via entity normalization (alias resolution and canonical mapping) and relation alignment to a unified CTINEXUS schema. For CTI-HAL reports, we apply distant supervision using ontology-guided pattern matching and threat-intelligence lexicons to generate noisy candidate triplets. These are

[3] https://github.com/aiforsec/MALOnt

then refined through manual curation, where a human annotator validates entities, corrects entity linking errors, and removes noisy relations.

Data Splits and Usage. We adopt the following (default) task-specific data partitioning strategy for our multi-agent pipeline:

- **Extractor & Typer Training:** up to 187 reports (539 chunks) are used to fine-tune the information extraction and entity typing modules.
- **Verifier & Curator Training:** 85 reports are used for training verification and curation components.
- **Evaluation:** Three held-out test sets are constructed: TEST0 with the 11 test reports from [10], TEST1 with 25 reports and TEST2 with 45 reports, sampled to ensure diversity in report styles and extraction difficulty. TEST2 extends TEST1 with more challenging reports.

Baselines. We compare TACTIC-KG against CTINEXUS [10], which introduces a three-phase pipeline for constructing CSKGs using LLMs through ICL [13]. The workflow integrates (1) triplet extraction from CTI reports, (2) hierarchical entity alignment and canonicalization, and (3) long-distance relation inference to merge disconnected subgraphs. It uses a k-Nearest Neighbor (kNN) retriever for dynamic demonstration selection, executes a single-inference triplet extraction prompt, and combines LLM-based typing with embedding-based merging to produce connected graphs aligned with MALOnt ontology [29].
We evaluate a diverse set of recent LLMs, including large-scale monolithic models (*e.g.*, DeepSeek-V3.1-671B, Kimi-K2-1T) and other open-weight models (*e.g.*, GPT-OSS-20B). To this end, we adapt the CTINEXUS[4] benchmark to support recent LLMs deployed via Ollama Cloud[5]. The CTINEXUS baselines follow a single-model paradigm, where extraction, typing and link prediction are performed jointly in a monolithic pipeline.
TACTIC-KG Configuration. As presented in Sect. 4, TACTIC-KG decomposes the pipeline into four specialized agents. In our default setup, only extraction and typing are agent-specific, while verification and curation rely on a *single fine-tuned reasoning model*. All tests are repeated 3 times on different seeds on two identical NVIDIA L40S GPUs. We run experiments with the following backbone models[6]: (i) Foundation-Sec-8B-Reasoning [34], (ii) Ministral-3-8B-Reasoning [23], (iii) Ministral-3-3B-Instruct [23], (iv) Qwen3-8B [37].
Metrics. We evaluate the quality of extracted knowledge graphs along three complementary dimensions:
Triplet Extraction. Given a set of gold triplets $\mathcal{G}$ and predicted triplets $\mathcal{P}$, we adopt a *semantic matching* [18, 42]. Each triplet $t = (s, r, o)$ is encoded into a dense vector representation $\phi(t)$ using a sentence embedding model (*e.g.*, all-mpnet-base-v2[7]) applied to the concatenation of its components. Pairwise similarities are computed using cosine similarity [6,21] with a threshold, $T = 0.6$ (see

[4] https://github.com/peng-gao-lab/CTINexus
[5] https://docs.ollama.com/cloud
[6] *Model suffixes (e.g., "Reasoning", "Instruct") are omitted in tables for brevity.*
[7] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Table 2: Best performance comparison on TEST0, TEST1 and TEST2.

Category	Model(s)	TEST0		TEST1		TEST2	
		F1	Graph	F1	Graph	F1	Graph
Best CTINEXUS	DeepSeek-V3.1	74	56	72	58	66	52
	Devstral-2	73	60	72	58	71	58
Best TACTIC-KG	Ministral-3-3B	73	62	79	63	76	61
	Ministral-3-8B	**77**	**62**	**80**	**67**	**78**	**63**

Table 3: TACTIC-KG performance (Model + LoRA) on TEST1.

Model	Role	Precision	Recall	F1	FTA	PTA	Graph
Ministral-3-3B [23]	Ext./Typ.	80	79	**79**	51	74	**63**
	Verifier	**83**	68	74	**52**	**75**	58
	Curator	76	**77**	75	51	74	61
Ministral-3-8B [23]	Ext./Typ.	73	**90**	**80**	**52**	**72**	67
	Verifier	**78**	85	**80**	50	71	**67**
	Curator	71	**90**	78	48	70	63

the Appendix A and B for ablation). We then compute an optimal one-to-one alignment between $\mathcal{G}$ and $\mathcal{P}$. Based on this alignment, we compute Precision (%), Recall (%), and F1-score (%):

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN}, \text{F1} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}.$$

Entity Typing. We evaluate typing quality at two levels. First, we report *Partial Typing Accuracy (PTA)*, which corresponds to the standard entity typing (or classification) accuracy commonly used in prior work [9,10]. PTA evaluates entity types independently and assigns credit when only one of the two (subject or object) is correctly predicted:

$$\text{PTA} = \frac{N_{\text{correct}} + 0.5 \cdot N_{\text{partial}}}{N}.$$

Second, we introduce a stricter metric, *Full Typing Accuracy (FTA)*, which requires both subject and object types of a triplet to be correctly predicted. This provides a more holistic assessment of typing quality at the triplet level.

Graph Structural Similarity. To assess the overall structural quality of the predicted graph, we compute a semantic variant of the Jaccard similarity between the sets of gold and predicted triplets. Let M denote the number of semantically matched triplets under threshold T. The graph similarity (%) is defined as:

$$\text{GraphSim} = \frac{M}{|\mathcal{G}| + |\mathcal{P}| - M}.$$

This metric captures both structural overlap and semantic equivalence between the predicted and gold KGs. All metrics are reported as mean values (%) over three runs (max std $\leq 1.0\%$).

Computational Efficiency. We evaluate CSKG construction time. Detailed results are provided in Appendix C.

5.2 RQ1: Does agentic decomposition improve extraction quality?

Table 2 compares TACTIC-KG with monolithic baselines.

Table 4: CTINEXUS baselines (Model + ICL) on TEST1.

Model	Precision	Recall	F1	FTA	PTA	Graph
GPT-OSS (20B) [2]	64	40	47	28	48	36
Nemotron-3 Nano (30B) [5]	64	57	58	16	44	44
GPT-OSS (120B) [2]	76	56	62	43	67	48
GLM-5 [38]	71	61	65	46	69	50
Kimi-K2 (1T) [32]	82	66	72	47	71	57
Devstral-2 (123B) [28]	**83**	65	72	**50**	**72**	58
DeepSeek-V3.1 (671B) [11]	**83**	**66**	**72**	45	69	**58**

Table 5: TACTIC-KG performance with hybrid models on TEST1.

Model	Role	Precision	Recall	F1	FTA	PTA	Graph
Foundation-Sec-8B [34]	Ext./Typ.	81	76	77	52	74	63
	Verifier	**86**	73	**78**	**53**	**75**	63
	Curator	77	**80**	78	48	72	**64**
Ministral-3-3B [23]	Ext./Typ.	80	79	79	**51**	**74**	63
	Verifier	**82**	75	78	50	73	63
	Curator	70	**83**	**75**	47	72	**69**
Qwen3-8B [37]	Ext./Typ.	84	75	78	50	71	68
	Verifier	**89**	74	**80**	**52**	**72**	**67**
	Curator	77	**85**	**80**	50	**72**	66

Observation 1: Higher overall performance. TACTIC-KG achieves up to **80** F1 and **67** GraphSim with Ministral-8B, outperforming large monolithic models such as DeepSeek-V3.1 (72 / 58).
Observation 2: Significant recall improvement. Agent-based extractors reach up to **90** recall, compared to ≤ 66 for monolithic baselines (Table 4).
Observation 3: Limitations of ICL for smaller models. ICL benefits large models, but smaller ones struggle (*e.g.*, GPT-OSS-20B reaches 47 F1), which proves the limited generalization from prompts (Sect.3).
*Answer to **RQ1.*** Agentic decomposition with parameter specialization via LoRA significantly improves extraction quality by increasing recall and overall F1, enabling smaller models to match or outperform much larger ICL systems.

5.3 RQ2: How does agent specialization affect PR trade-offs?

TACTIC-KG exhibits distinct behaviors across agent roles (Table 3).
Observation 1: Extractors favor recall. Extractors may aggressively explore candidate relations, achieving high recall (up to **90**) at moderate precision.
Observation 2: Verifiers balance precision and recall. Verifiers filter hallucinations and improve F1 (up to **80**) while maintaining strong recall.
Observation 3: Curators improve completeness. Curators recover missing links and further increase recall (up to **85**), improving graph completeness.
*Answer to **RQ2.*** Agent specialization enables explicit control of the precision–recall trade-off by distributing complementary roles across stages, achieving high recall while maintaining stable precision. In contrast, ICL monolithic models rely on single-pass reasoning, which struggles to reliably infer unstated relations. This often leads to over-completion or artificial connectivity, where spurious links are introduced between subgraph nodes and the main topic node (Sect. 3).

5.4 RQ3: Does TACTIC-KG improve typing, semantic grounding, and structural consistency?

TACTIC-KG demonstrates consistent improvements in typing and semantic grounding (Tables 3 and 4).
Observation 1: Higher typing accuracy. TACTIC-KG achieves up to **72** PTA and **52** FTA, outperforming strong ICL baselines such as DeepSeek (≈69 PTA).

Table 6: Impact of training data volume and diversity on TACTIC-KG (Extractor & Typer) performance on TEST2.

Version	Model	Precision	Recall	F1	FTA	PTA	Graph
TC-1: Diverse, 539 chunks	Ministral-3-8B	72	**89**	**78**	**48**	**70**	**63**
	Ministral-3-3B	71	85	76	42	66	61
TC-2: Single, 201 chunks	Ministral-3-8B	73	85	76	47	68	60
	Ministral-3-3B	78	77	76	48	70	61
TC-3: Single, 100 chunks	Ministral-3-3B	69	79	72	44	68	58
TC-4: Diverse, 192 chunks	Ministral-3-3B	**79**	79	**78**	47	70	**63**
CTINEXUS: Models + ICL	Devstral-2:123b	80	68	72	48	71	58
	Deepseek-v3.1:671b	**82**	57	66	53	74	52
	Kimi-k2:1t	80	63	69	47	68	56

Observation 2: Improved semantic grounding. Reduced cognitive load (LoRa vs. ICL demos) on individual agents and cleaner intermediate representations with ontology-aware verification reduce hallucinations and improve typing.

Observation 3: Better structural consistency. Graph scores reach up to **68**, compared to 58 for DeepSeek, indicating more coherent graph construction.

*Answer to **RQ3.*** TACTIC-KG improves typing accuracy, semantic grounding, and structural consistency by enforcing ontology-aware reasoning and reducing error propagation across stages (Sect. 3). The observed gains are a direct result of the reduced load and the constrained verification and curation (*i.e.*, the ontology is used during both fine-tuning and inference). Thus, any new entity or relation must remain consistent with ontology schema, which in turn limits type ambiguity and reduces hallucinated links.

5.5 RQ4: What is the impact of a hybrid agentic design?

Table 5 shows that TACTIC-KG improves performance with shared reasoning (Ministral-8B + LoRA for verification and curation).

Observation 1: Improved stability across roles. Hybrid configurations maintain consistently high precision, recall, and graph scores across all roles.

Observation 2: Enhanced reasoning consistency. Shared reasoning models improve typing and partial typing (up to **72** PTA), while stabilizing outputs.

Observation 3: Balanced performance. For example, Qwen3 achieves strong precision (**89**), recall (**85**), and graph scores (**67**).

*Answer to **RQ4.*** Hybrid agentic design improves robustness and consistency by decoupling *generation* from *validation*. Using a stronger, shared reasoning model for verification and curation allows consistent filtering over heterogeneous extractor outputs. This model enforces ontology constraints and corrects local extraction errors. As a result, different extractors can either prioritize recall or precision, while the centralized verifier–curator pair suppresses hallucinations and resolves conflicts, leading to more stable and higher-quality final graphs.

5.6 RQ5: How much training data does TACTIC-KG need to reach monolithic model performance?

Table 6 reports the performance of TACTIC-KG under different training configurations (TC) and model sizes on TEST2. We compare (i) TC-1, where models are trained on a larger and more diverse dataset (185 reports from [12] + [10]), (ii) TC-2, where the extractor and the typer are fine-tuned on 108 reports from [10] data only, (iii) TC-3, where the training is restricted to a fraction of [10] only and, (iv) TC-4, where the training uses a fraction of both datasets.
Observation 1: Performance improves with more data. Increasing training data from 100 to 201 chunks improves F1 (72 → 76) and recall (79 → 85).
Observation 2: Diversity matters more than volume. Using more diverse data (TC-1, TC-4) yields the best performance (F1 **78**, PTA **70**).
*Answer to **RQ5.*** TACTIC-KG requires only moderate training data (∼200 chunks) to reach strong performance, with data diversity playing a more critical role than volume.

6 Discussion

Results demonstrate that agent specialization via LoRA improves both stability and extraction quality, even when using smaller models. TACTIC-KG consistently outperforms larger ICL monolithic pipelines across F1, recall, and graph quality, highlighting the benefits of task decomposition. By separating roles, the framework enables finer control over the precision–recall trade-off, allowing extractors to prioritize coverage while downstream stages refine correctness. Graph consistency is further enhanced through verification and curation, which reduce error propagation and enforce coherence. Moreover, a hybrid design combining specialized extractors and typers with a shared reasoning model (verifier and curator) provides additional gains in overall performance. Despite these improvements, full typing accuracy remains a bottleneck across all systems.

7 Conclusion

This paper introduced TACTIC-KG, an agentic framework for constructing Cyber Threat Intelligence Knowledge Graphs. By decomposing CSKG construction into specialized small LLM agents, TACTIC-KG achieves higher accuracy, stability, and cost-efficiency than larger monolithic LLM-based approaches. Although this work focuses on publicly available CTI reports, the proposed framework naturally extends to enterprise settings, enabling organizations to build and maintain CSKGs while keeping sensitive data local, with minimal deployment overhead.
Future work will focus on robust graph completion, stronger ontology constraints, and CSKG repair under noisy and adversarial CTI inputs.

Acknowledgments. This research is supported by the CKRISP project (ANR-23-IAS4-0001).

Table 7: Impact of similarity threshold on entity and relation merging (embedding model: *all-MiniLM-L12-v2*).

T	Verifier			Curator		
	Precision	Recall	F1	Precision	Recall	F1
0.6	83	66	73	**73**	79	75
0.7	**84**	72	77	72	80	75
0.8	81	75	78	69	84	75
0.9	81	**77**	**78**	70	**84**	**76**

Table 8: The impact of the evaluation embedding models on TACTIC-KG (Ministral-3B).

Embedding Model (T)	Extraction			Verification			Curation		
	P	R	F1	P	R	F1	P	R	F1
MiniLM-L6 (0.6)	77	77	76	79	75	76	70	83	75
MPNet (0.6)	79	78	78	79	75	76	70	84	76
MiniLM-L12 (0.6)	80	79	79	81	77	78	70	84	76
MiniLM-L12 (0.7)	69	69	69	71	65	68	62	73	66
Paraphrase (0.6)	**81**	**81**	**80**	**82**	**78**	**80**	**72**	**86**	**77**

Table 9: Impact of the evaluation embedding models on CTINEXUS (DeepSeek).

Embedding Model (T)	P	R	F1
all-MiniLM-L6-v2 (0.6)	78	62	67
all-mpnet-base-v2 (0.6)	83	66	72
all-MiniLM-L12-v2 (0.6)	83	66	72
all-MiniLM-L12-v2 (0.7)	69	54	59
paraphrase-MiniLM-L6-v2 (0.6)	**83**	**67**	**72**

Table 10: Per-stage and end-to-end time overhead of models (seconds).

Model	IE	ET	VER/EA	CUR/LP	E2E
CTINEXUS					
Kimi-k2:1t	9	14	0-1	0-1	23
Deepseek-v3.1	22	47	0-1	0-1	72
Devstral-2:123b	**5-6**	**6-7**	**0-1**	**0-1**	**12**
TACTIC-KG					
Foundation-Sec-8B	31	53	58	18	155
Qwen3-8B	46	183	59	17	297
Ministral-3-8B	26	26	67	25	145
Ministral-3-3B	**12**	**23**	**58**	**21**	**115**

A Impact of Similarity Threshold on Entity/Relation Merging.

We analyze the effect of the similarity threshold $T \in \{0.6, 0.7, 0.8, 0.9\}$ used for clustering semantically similar entities and relations (Table 7). This threshold controls the granularity of canonicalization: lower values encourage aggressive merging, while higher values enforce stricter matching. Increasing T consistently improves recall (*e.g.*, curated recall rises from 79 to 84), as stricter merging preserves distinct semantics. Precision shows a non-monotonic trend—peaking at moderate thresholds and slightly decreasing at higher T due to under-merging. Overall, F1 improves with higher T (up to 78 in verification and 76 in curation), indicating a better precision–recall balance.

B Impact of Embedding Models on Semantic Evaluation

Tables 8 and 9 evaluate the effect of the embedding model used to compute semantic similarity between predicted and gold triplets ($T \in [0.6, 0.7]$). For TACTIC-KG, the extraction (Ministral-3B) and verification (Ministral-8B) models remain fixed; only the similarity function varies. Across both our pipeline and CTINEXUS (with *Deepseek*), results show a consistent performance gap between embedding models. *paraphrase-MiniLM-L6-v2* achieves the best overall performance, with gains of up to +1.5 F1, while *all-MiniLM-L6-v2* yields the lowest scores, due to limited representational capacity. Variations are primarily driven by recall rather than precision: precision remains relatively stable, whereas recall increases with paraphrase embeddings (*e.g.*, from 83 to

86), which means that more predictions are considered semantically correct. In contrast, *all-mpnet-base-v2* consistently produces slightly lower recall and F1 with a stricter semantic matching criterion. This trend holds across both pipelines, which highlights the importance of reporting results with multiple embedding models.

C Execution Time and E2E Analysis

Table 10 reports per-stage and end-to-end (E2E) execution times. We note that these timings are indicative only: CTINEXUS pipline was executed on the OLLAMA cloud, whereas TACTIC-KG models ran locally on NVIDIA L40S GPUs. Differences in hardware, concurrency, and runtime configurations can therefore impact direct comparisons. CTINEXUS large models (*e.g.*, Deepseek-v3.1) achieve fast inference for individual tasks. Specialized agents demonstrate higher per-stage costs for reasoning and verification due to their multi-step design, but they benefit from improved extraction quality and graph consistency. Lightweight or smaller models (*e.g.*, Ministral-3B) achieve competitive times with robust construction performance.

References

1. Agarwal, R., Singh, A., Zhang, L., Bohnet, B., Rosias, L., Chan, S., Zhang, B., Anand, A., Abbas, Z., Nova, A., et al.: Many-shot in-context learning. Advances in Neural Information Processing Systems **37**, 76930–76966 (2024)
2. Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R.K., Bai, Y., Baker, B., Bao, H., et al.: gpt-oss-120b & gpt-oss-20b model card. arXiv preprint arXiv:2508.10925 (2025)
3. Alam, M.T., Bhusal, D., Park, Y., Rastogi, N.: Looking beyond iocs: Automatically extracting attack patterns from external cti. In: 26th international symposium on research in attacks, intrusions and defenses. pp. 92–108 (2023)
4. Arslan, M., Ghanem, H., Munawar, S., Cruz, C.: A survey on rag with llms. Procedia computer science **246**, 3781–3790 (2024)
5. Blakeman, A., Grattafiori, A., Basant, A., Gupta, A., Khattar, A., Renduchintala, A., Vavre, A., Shukla, A., Bercovich, A., Ficek, A., et al.: Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. arXiv preprint arXiv:2512.20848 (2025)
6. Bouchiha, M.A., Telnoff, Q., Bakkali, S., Champagnat, R., Rabah, M., Coustaty, M., Ghamri-Doudane, Y.: Llmchain: Blockchain-based reputation system for sharing and evaluating large language models. arXiv preprint arXiv:2404.13236 (2024)
7. Büchel, M., Paladini, T., Longari, S., Carminati, M., Zanero, S., Binyamini, H., Engelberg, G., Klein, D., Guizzardi, G., Caselli, M., et al.: {SoK}: Automated {TTP} extraction from {CTI} reports–are we there yet? In: 34th USENIX security symposium (USENIX Security 25). pp. 4621–4641 (2025)
8. Chen, M., Zhu, K., Lu, B., Li, D., Yuan, Q., Zhu, Y.: Aecr: Automatic attack technique intelligence extraction based on fine-tuned large language model. Computers & Security **150**, 104213 (2025)
9. Cheng, W., Zhu, T., Chen, T., Yuan, Q., Ying, J., Li, H., et al.: CRUcialG: Reconstruct Integrated Attack Scenario Graphs by Cyber Threat Intelligence Reports. IEEE Transactions on Dependable and Secure Computing (2025)

10. Cheng, Y., Bajaber, O., Tsegai, S.A., et al.: Ctinexus: Automatic cyber threat intelligence knowledge graph construction using large language models. In: European Symposium on Security and Privacy (EuroS&P). pp. 923–938. IEEE (2025)
11. DeepSeek-AI: Deepseek-v3 technical report (2024), https://arxiv.org/abs/2412.19437
12. Della Penna, S., Natella, R., Orbinato, V., Parracino, L., et al.: Cti-hal: A human-annotated dataset for cyber threat intelligence analysis. In: European Symposium on Security and Privacy Workshops (EuroS&PW). pp. 69–78. IEEE (2025)
13. Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., et al.: A survey on in-context learning. In: Proceedings of the 2024 conference on empirical methods in natural language processing. pp. 1107–1128 (2024)
14. Fayyazi, R., Taghdimi, R., Yang, S.J.: Advancing ttp analysis: Harnessing the power of large language models with retrieval augmented generation. In: Annual Computer Security Applications Conference Workshops (ACSAC Workshops). pp. 255–261. IEEE (2024)
15. Fengrui, Y., Du, Y.: Few-shot learning of ttps classification using large language models (2024)
16. Fieblinger, R., Alam, M.T., Rastogi, N.: Actionable cyber threat intelligence using knowledge graphs and large language models. In: European symposium on security and privacy workshops (EuroS&PW). pp. 100–111. IEEE (2024)
17. Gao, P., Shao, F., Liu, X., Xiao, X., Qin, Z., Xu, F., Mittal, P., Kulkarni, S.R., Song, D.: Enabling efficient cyber threat hunting with cyber threat intelligence. In: 37th International Conference on Data Engineering. pp. 193–204. IEEE (2021)
18. Ge, X., Wang, Y.C., Wang, B., Kuo, C.C.J.: Knowledge graph embedding: An overview. arXiv preprint arXiv:2309.12501 (2023)
19. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al.: Lora: Low-rank adaptation of large language models. Iclr **1**(2), 3 (2022)
20. Kumarasinghe, U., Lekssays, A., Sencar, H.T., Boughorbel, S., Elvitigala, C., Nakov, P.: Semantic ranking for automated adversarial technique annotation in security text. In: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. pp. 49–62 (2024)
21. Lairgi, Y., Moncla, L., Benabdeslem, K., Cazabet, R., Cléau, P.: Atom: Adaptive and optimized dynamic temporal knowledge graph construction using llms. In: Findings of the Association for Computational Linguistics. pp. 950–966 (2026)
22. Li, Z., Zeng, J., Chen, Y., Liang, Z.: Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In: European Symposium on Research in Computer Security. pp. 589–609. Springer (2022)
23. Liu, A.H., Khandelwal, K., Subramanian, S., Jouault, V., Rastogi, A., et al.: Ministral 3. arXiv preprint arXiv:2601.08584 (2026)
24. Liu, L., Omidvar, A., Ma, Z., Agrawal, A., An, A.: Unsupervised knowledge graph generation using semantic similarity matching. In: 3rd workshop on deep learning for low-resource natural language processing. pp. 169–179 (2022)
25. MISP Project: Misp: Open source threat intelligence platform and open standards for threat information sharing. https://www.misp-project.org/ (2026)
26. MITRE Corporation: Mitre att&ck framework. https://attack.mitre.org/ (2023), accessed: 2025-09-20
27. OASIS Cyber Threat Intelligence (CTI) Technical Committee: Stix version 2.1. https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html (2021), edited by Bret Jordan, Rich Piazza, and Trey Darley. OASIS Standard

28. Rastogi, A., Yang, A., Jiang, A.Q., Liu, A.H., Sablayrolles, A., Héliou, A., Martin, A., Agarwal, A., Ehrenberg, A., Lo, A., et al.: Devstral: Fine-tuning language models for coding agent applications. arXiv preprint arXiv:2509.25193 (2025)
29. Rastogi, N., Dutta, S., Zaki, M.J., Gittens, A., Aggarwal, C.: Malont: An ontology for malware threat intelligence. In: International workshop on deployable machine learning for security defense. pp. 28–44. Springer (2020)
30. Shuttleworth, R., Andreas, J., Torralba, A., Sharma, P.: Lora vs full fine-tuning: An illusion of equivalence. arXiv preprint arXiv:2410.21228 (2024)
31. Siracusano, G., Sanvito, D., Gonzalez, R., Srinivasan, M., Kamatchi, S., Takahashi, W., Kawakita, M., Kakumaru, T., Bifulco, R.: Time for action: Automated analysis of cyber threat intelligence in the wild. arXiv preprint arXiv:2307.10214 (2023)
32. Team, K., Bai, Y., Bao, Y., Charles, Y., et al.: Kimi k2: Open agentic intelligence. arXiv preprint arXiv:2507.20534 (2025)
33. Waldrop, S., Dogdu, E., Choupani, R., Mitchell, W., et al.: Semantic chunking and consensus filtering for structured extraction of cyber threat intelligence. In: 5th International Conference on AI in Cybersecurity (ICAIC). pp. 1–11. IEEE (2026)
34. Weerawardhena, S., Kassianik, P., Nelson, B., Saglam, B., Vellore, A., Priyanshu, A., Vijay, S., Aufiero, M., Goldblatt, A., Burch, F., et al.: Llama-3.1-foundationai-securityllm-8b-instruct technical report. arXiv preprint arXiv:2508.01059 (2025)
35. Wei, J., Wei, J., Tay, Y., Tran, D., Webson, A., Lu, Y., Chen, X., Liu, H., Huang, D., Zhou, D., et al.: Larger language models do in-context learning differently. arXiv preprint arXiv:2303.03846 (2023)
36. Xu, M., Wang, H., Liu, J., Lin, Y., Yingshi Liu, C.X., Lim, H.W., Dong, J.S.: Intelex: A llm-driven attack-level threat intelligence extraction framework. arXiv e-prints pp. arXiv–2412 (2024)
37. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025)
38. Zeng, A., Lv, X., Hou, Z., Du, Z., Zheng, Q., et al.: Glm-5: from vibe coding to agentic engineering. arXiv preprint arXiv:2602.15763 (2026)
39. Zhang, Q., Hu, C., Upasani, S., Ma, B., Hong, F., Kamanuru, V., Rainton, J., Wu, C., Ji, M., Li, H., et al.: Agentic context engineering: Evolving contexts for self-improving language models. arXiv preprint arXiv:2510.04618 (2025)
40. Zhao, W., Liu, Y., Wan, Y., Wang, Y., Wu, Q., Deng, Z., Du, J., Liu, S., Xu, Y., Yu, P.S.: knn-icl: Compositional task-oriented parsing generalization with nearest neighbor in-context learning. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 326–337 (2024)
41. Zhao, X., Jiang, R., Han, Y., Li, A., Peng, Z.: A survey on cybersecurity knowledge graph construction. Computers & Security **136**, 103524 (2024)
42. Zhu, G., Iglesias, C.A.: Sematch: Semantic similarity framework for knowledge graphs. Knowledge-Based Systems **130**, 30–32 (2017)