

Schema-Agnostic Knowledge Graph Construction via Hybrid Ontology Discovery for Cyber Threat Intelligence

Seonwoo Kim

Ministry of National Defense
Seoul, Republic of Korea

Jinwoo Kim

Incheon International Airport Corporation
Incheon, Republic of Korea

Daegyung Kang

Financial Security Institute
Yongin, Republic of Korea

Daeseong Kim

Korean National Police Agency
Seoul, Republic of Korea

Insup Lee*

Korea University
Seoul, Republic of Korea

Abstract—Cyber threat intelligence (CTI) reports now serve as essential resources for capturing adversary tactics, techniques, and procedures observed in modern attack campaigns. While traditional CTI platforms reduce this intelligence to isolated indicators through fixed schemas such as STIX, ontology-based representations preserve the semantic relationships needed for structured threat analysis. However, existing approaches for ontology-aligned CTI extraction face three challenges: (i) schema-specific pipelines that require manual reconfiguration whenever the schema changes, (ii) prompt-based schema inclusion that fails to scale on large ontologies such as UCO, and (iii) reliance on enterprise LLM APIs that conflicts with privacy constraints when integrating sensitive internal incident data. In this paper, we present ANCHOR, a *schema-agnostic* CTI knowledge graph construction system that bridges LLMs and formal ontology schemas. At the core of ANCHOR is *hybrid ontology discovery*, a search-and-navigate mechanism that dynamically explores large-scale ontology schemas, combined with SHACL-based validation to enforce schema-compliant type assignments. Experimental results on the UCO, STIX, and MALOnt schemas show that ANCHOR outperforms existing baselines in ontology typing and schema compliance. In addition, ANCHOR with a local LLM closely matches enterprise LLM typing performance, enabling privacy-preserving CTI analysis with high fidelity.

Index Terms—Cyber threat intelligence, ontology, knowledge graph, large language models

I. INTRODUCTION

A comprehensive analysis of cyber campaigns requires more than isolated indicators of compromise (IoC) [1]. Cyber threat intelligence (CTI) reports record the threat actors, attack sequences, and technical evidence, explicitly detailing the *causal* relationships between attack steps. To facilitate automated sharing of such knowledge, standardized data formats such as the structured threat information expression (STIX) [2] and the malware information sharing platform (MISP) [3] have been widely adopted. However, these indicator-oriented formats primarily capture low-level, isolated data points such as IP addresses and file names, discarding the causal dependencies between attack steps and the analytical reasoning that links them [4]–[6].

Ontology-based representations address this semantic gap by defining a formal vocabulary of ontology classes, their

properties (object and datatype), and structural constraints. Notable efforts include the unified cyber ontology (UCO) [7], STUCCO [8], and MALOnt [9]. These frameworks are typically formalized in the web ontology language (OWL) for class hierarchies and properties, supplemented by optional shapes constraint language (SHACL) constraints for structural validation. However, despite the introduction of many security ontologies over the past decade, no single schema has achieved widespread adoption in practice [10]. Recent studies have therefore explored the automated extraction of ontology-aligned knowledge from unstructured CTI text. Conventional approaches rely on rule-based pipelines [11], [12] and classification models [13], [14]. Unfortunately, these methods depend on hand-crafted rules or fixed type inventories, requiring a complete redesign whenever the target schema changes. Large language model (LLM)-based methods [15]–[17] are better suited for schema-agnostic extraction because they can interpret ontology class descriptions directly without schema-specific engineering. However, existing LLM-based approaches have been validated only on small custom schemas and exhibit critical limitations when applied to large-scale, real-world ontologies [18].

We identify three limitations of existing LLM-based automated ontology extraction approaches. First, conventional methods [11], [15]–[17] either define their own custom schemas or hard-code a specific ontology into the extraction pipeline (*schema dependency*). This rigid design requires a complete redesign whenever the underlying schema changes. The resolution of schema dependency alone does not solve the scalability problem, as the system must still present the target ontology to the LLM at inference time. Second, prompt-based schema inclusion strategies fail on large-scale ontologies such as UCO. Hundreds of hierarchical classes exhaust the context window and degrade the model’s ability to distinguish semantically similar types. Third, an inherent reliance on enterprise LLMs complicates the integration of external CTI with sensitive internal incident data due to privacy and data-sovereignty concerns. In this context, we derive the following three challenges from the existing literature.

Challenge 1) How can we extract ontology-aligned knowledge without being tied to a specific schema? To

*Corresponding author: Insup Lee (islee94@korea.ac.kr)

support diverse and evolving security standards [19], the system must decouple extraction logic from schema definitions. This decoupling allows different OWL/SHACL ontologies to be loaded without manually reconfiguring the pipeline.

Challenge 2) How can we handle large-scale ontologies that exceed the capacity of prompt-based schema inclusion? Rather than including the entire schema in the LLM prompt, the system must dynamically discover and retrieve only task-relevant ontology fragments. This dynamic retrieval enables schema-aware reasoning within large and hierarchical ontologies while keeping each query within the context window.

Challenge 3) How can we support privacy-preserving CTI analysis with local LLMs? External CTI provides greater operational value when integrated with internal alerts, logs, and incident reports, but such data is often too sensitive to expose to enterprise LLMs. The system must therefore enable ontology-grounded reasoning with locally deployed open-source LLMs while preserving competitive typing performance.

To this end, we propose ANCHOR (Adaptive Navigation for Cybersecurity Hybrid Ontology Reasoning), a *schema-agnostic* system that builds a structured threat knowledge graph from unstructured CTI reports. The proposed method presents a *hybrid ontology discovery* mechanism, which enables LLMs to dynamically explore relevant sub-graphs of complex ontologies and separates the extraction pipeline from any specific schema. To ensure accurate and schema-compliant ontology alignment, ANCHOR also considers a SHACL-based validation during ontology typing. The main contributions of this paper are summarized as follows.

- We propose ANCHOR, a schema-agnostic CTI knowledge graph construction framework that decouples the extraction pipeline from any specific ontology schema, supporting arbitrary OWL/SHACL ontologies at runtime without manual reconfiguration.
- We introduce *hybrid ontology discovery*, which combines embedding-based semantic search with LLM-guided recursive navigation to retrieve only task-relevant ontologies, integrated with closed-loop SHACL validation to ensure schema-compliant knowledge graphs.
- We demonstrate privacy-preserving CTI knowledge graph construction with locally deployed open-source LLMs, retaining 99.2% (entity) and 97.8% (predicate) of the best enterprise LLM’s performance.

The source code of ANCHOR will be publicly available in the near future to foster further research.

II. BACKGROUND

This section reviews the necessary background on CTI data formats, ontologies, and LLM-based knowledge extraction.

A. CTI Data Formats and Ontologies

CTI records adversary tactics, techniques, and procedures (TTPs) and IoCs. This intelligence serves as a shared resource for organizations to coordinate defenses against rapidly

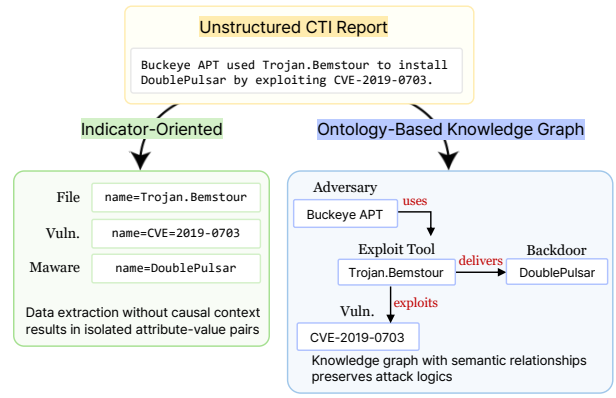


Fig. 1. Comparison of two CTI data formats: indicator-oriented flat data and ontology-based knowledge graph.

changing threats [20]. Open-source CTI (OSCTI) is collected from security blogs, threat reports, and vulnerability databases, providing a critical resource for understanding dynamic threat environments. To standardize the exchange of threat information, the security community widely adopts three data formats and protocols. STIX [2] defines a set of domain objects (e.g., threat actors, malware, vulnerabilities) and their relationships in a JSON-based format. MISP [3] provides a collaborative framework to share IoCs using a predefined attribute taxonomy. The trusted automated exchange of intelligence information (TAXII) [21] acts as the transport protocol to distribute STIX-formatted data between organizations.

While these formats enable efficient indicator sharing, their structures are limited to flat attribute-value pairs and predefined relationship types [4]. Fig. 1 illustrates differences between this indicator-oriented view and an ontology-based knowledge graph. In the threat knowledge graph, semantic relationships connect security entities into a coherent structure. This connectivity allows analysts to trace the full attack flow rather than examine individual indicators in isolation. Ontologies are typically expressed in the web ontology language (OWL), which defines class hierarchies alongside object and datatype properties. These definitions often include optional shapes constraint language (SHACL) constraints for structural validation, such as required attributes and cardinality bounds.

UCO [7] unifies concepts from multiple security standards (e.g., CyBOK, NIST, MITRE ATT&CK) into a single hierarchical framework defined in OWL/SHACL, containing hundreds of classes and properties. In contrast, MALOnt [9] provides a lightweight malware-focused ontology with only 75 classes, 10 relations, and 12 properties. Despite the studies on security ontologies, the construction and maintenance of a comprehensive cybersecurity ontology remains an open problem. A recent survey reports inconsistencies and ambiguities in cybersecurity terminology that hinder communication between security professionals [10]. Furthermore, no existing ontology fully aligns with the major security standards required for seamless cross-platform integration. These practical limitations motivate the scenarios described in Section III.

B. LLM-based Knowledge Extraction

LLMs have demonstrated strong performance in natural language understanding and information extraction tasks [22], [23]. In the CTI domain, LLMs offer a practical alternative to traditional rule-based or keyword-matching systems. While these traditional systems struggle to capture the intent and contextual nuances of attacker behavior, recent studies [5], [15]–[17] show that LLMs can extract key entities and their semantic relationships from threat reports with high performance. This automated extraction reduces the need for manual analyst intervention in knowledge graph construction.

The integration of LLMs with external knowledge sources, such as databases and ontology schemas, has traditionally required separate implementations for each data source. This requirement creates tightly coupled pipelines that are difficult to maintain. Recent agent frameworks address this fragmentation by providing standardized interfaces to access external knowledge dynamically. This interactive approach grounds LLM outputs in verifiable references rather than relying solely on parametric knowledge. Our design adopts this methodology to ensure flexibility and maintainability. A uniform interface between the LLM agent and the external ontology, aligned with open standards such as the model context protocol (MCP) [24], successfully decouples the extraction logic from any specific schema (Section IV).

III. MOTIVATING EXAMPLES

This section illustrates the practical limitations of current automated CTI extraction through two scenarios: (i) information loss issues common to all indicator-oriented pipelines and (ii) a scalability limitation caused by prompt-based schema inclusion in LLM.

A. Information Loss in Indicator-oriented Formats

To illustrate the information loss caused by indicator-oriented CTI data, consider the following passage from typical threat analysis reports:

“In early March 2024, security analysts identified a spear-phishing campaign targeting Southeast Asian financial institutions. The attacker delivered a Microsoft Word document disguised as an invoice via email. When a user opens the document, a macro executes to download a secondary payload from `hxxp://update-check[.]site/loader.exe`. This payload maintained access by creating a scheduled task named “WindowsUpdateCheck” and afterward communicated with a command and control (C2) server at 45.77.23.91 via TCP port 443. Analysis revealed that these techniques align with the activities of APT-X, known for macro-based initial access and invoice-themed lures.”

When this scenario is mapped into indicator-oriented formats such as STIX, most CTI platforms extract only a limited subset of isolated indicators:

- **Indicators:**

TABLE I
COMPARISON OF CTI KNOWLEDGE GRAPH CONSTRUCTION SYSTEMS

System	Method	Schema dependency	Schema size
TTPDrill [11]	Rule	Fixed (ATT&CK)	-
CTIKG [15]	LLM (prompt)	Open-ended	-
CTINexus [16]	LLM (prompt)	Fixed (MALOnt)	60
LLM4CTI [17]	LLM (prompt)	Fixed (STIX)	77
ANCHOR	LLM (hybrid)	Any OWL/SHACL	Up to 997 (UCO)

- URL: “update-check[.]site”
- IP: “45[.]77[.]23[.]91”
- File: “loader.exe”

- **Objects:**

- Scheduled Task: “WindowsUpdateCheck”
- Identity: *Financial Sector*

- **Relationships:**

- Attributed-to *APT-X*

Note that this isolation leads to the loss of semantic connectivity: indicator-oriented formats log discrete values while omitting the causality and rationale that link them together. For instance, the passage above includes a causal chain (e.g., (i) macro execution downloads the secondary payload and (ii) the C2 connection enables remote control) and an attribution rationale (TTP overlap with APT-X), yet the STIX output preserves only flat identifiers without any of this contextual reasoning. As a result, the structured output captures a fragmented snapshot of the original report, with no representation of the attack logic that connects these indicators. To preserve these crucial relationships, an ontology-based knowledge graph offers a promising alternative that explicitly models semantic connectivity between threat entities.

B. Limited Scalability of Prompt-Based Schema Inclusion

Recent LLM-based methods map entities to ontologies using prompt-based schema inclusion, which embeds the entire class list directly into the input prompt. As shown in Table I, existing systems [11], [15]–[17] either hard-code a specific schema, operate without one, or rely on prompt-based inclusion at a small scale (60 to 77 elements). While prompt-based inclusion is feasible for small schemas like MALOnt and STIX, it introduces three severe limitations when applied to large-scale ontologies such as UCO (419 classes):

- **Context exhaustion:** Massive prompts with hundreds of class descriptions trigger the “Lost in the Middle” phenomenon [25], causing the model to miss relevant context.
- **Precision degradation:** An excessive number of candidates degrades semantic precision, causing the model to confuse adjacent types (e.g., *Process*, *Action*, *Event*) or hallucinate non-existent classes.
- **Maintenance burden:** A fixed class list requires manual reconfiguration whenever the schema is updated.

Furthermore, none of the existing baselines verify whether assigned types satisfy formal structural constraints, allowing invalid mappings to enter the knowledge graph undetected. These scalability and validation gaps highlight a critical mis-

alignment between static LLM prompts and complex security ontologies. To process large-scale schemas without context exhaustion, a system requires dynamic exploration that retrieves only task-relevant fragments on demand. Furthermore, to prevent hallucinated mappings from corrupting the output, the system must enforce formal constraints before committing the data. Motivated by these exact requirements, Section IV introduces the architectural design of ANCHOR, which replaces prompt-based full schema inclusion with an effective ontology discovery and validation mechanism.

IV. ANCHOR DESIGN

We propose ANCHOR, a schema-agnostic threat knowledge graph construction system that aligns entities and predicates to large-scale OWL/SHACL ontologies via hybrid ontology discovery. This section describes the system overview, the schema-agnostic extraction pipeline, and the hybrid ontology discovery procedure.

A. System Overview

ANCHOR aims to construct threat knowledge graphs from fragmented CTI data based on the following three principles: (i) the extraction pipeline is decoupled from any specific ontology schema, supporting arbitrary OWL/SHACL ontologies at runtime; (ii) ontology fragments are discovered *dynamically* rather than included as a whole in the LLM prompt, keeping each query within the context window; (iii) during threat knowledge graph construction, every type assignment is verified against formal schema constraints before commitment, ensuring schema compliance. As shown in Fig. 2, the resulting architecture consists of three core components: schema-agnostic extraction, hybrid ontology discovery, and knowledge graph construction.

Schema-agnostic extraction. This stage extracts entities, coreferences, and relation triplets from preprocessed CTI text without binding the extraction logic to any specific ontology, and forwards the structured output to the next stage for type assignment. Section IV-B details the full extraction pipeline.

Hybrid ontology discovery. This stage aligns each extracted element to a formal ontology class or property URI by combining embedding-based semantic search with LLM-guided hierarchical navigation over the target OWL/SHACL schema. When the search confidence falls below a predefined threshold, the system switches to recursive schema traversal, ensuring that domain-specific or novel terminology is correctly resolved. Section IV-C details the discovery algorithm.

Knowledge graph construction. The validated type assignments are assembled into an ontology-aligned knowledge graph through a SHACL-based closed-loop correction process: each candidate assignment is checked against the target schema, and the mapper is re-invoked with the violation report until the output is conformant or a retry budget is exhausted. The resulting graph is serialized as JSON and, optionally, as a Neo4j-importable Cypher file to support downstream multi-hop queries over attack chains.

B. Schema-Agnostic Extraction

To prepare unstructured CTI data for ontology-aligned knowledge extraction, ANCHOR processes raw inputs through a sequential pipeline: (i) input preprocessing, (ii) knowledge extraction, and (iii) post-processing and filtering.

1) *Input Preprocessing:* Given unstructured CTI text such as threat reports, ANCHOR applies a recursive character-based splitting strategy to handle documents that exceed the LLM context window. The splitter respects natural semantic boundaries (paragraph breaks, sentence boundaries, word boundaries) and retains a fixed-size overlap window between consecutive chunks to preserve cross-boundary context.

2) *Knowledge Extraction:* After preprocessing, ANCHOR extracts structured entities and relationships from preprocessed chunks through three phases: entity extraction, coreference resolution, and triplet extraction.

Entity extraction. ANCHOR identifies all named entities from each chunk through a dedicated LLM extraction call. An entity here refers to any phrase that denotes a concrete object or referent in the text, in contrast to descriptive expressions (e.g., adjectives, adverbs) or behavioral phrases that do not stand as standalone referents.

Rather than restricting the process to predefined entity classes, the system captures a broad spectrum of referents and defers formal ontology type assignment to the subsequent discovery stage. Each extracted entity is represented as a tuple $(name, type_hint, properties)$, where *type_hint* is a short phrase that guides ontology class search, and *properties* captures clearly stated literal attributes (e.g., implementation language, alias, first-seen date). To maintain consistent referencing, the extraction enforces name normalization (e.g., stripping leading articles, lowercasing) and performs cross-chunk unification using a normalized key, so that surface variations such as “Fancy Bear Hacking Group” and “Fancy Bear” map to the same canonical entry while non-entity behavioral descriptions are filtered out.

Coreference resolution. After entity extraction, ANCHOR applies an entity-aware coreference resolution pass to each chunk. The LLM replaces anaphoric references (pronouns, role descriptors, near-demonstratives) with the corresponding canonical entity name from the unified list, reducing misattributed or missing relations caused by unresolved anaphora.

Triplet extraction. With the extracted entity list in place, the system extracts structured relationships per chunk, providing the entity list as an explicit restriction so that only known, verified entities appear as relation endpoints. This two-pass design (entities first, relations second) eliminates subject-bias (over-generating relations for prominent entities while neglecting less salient ones) and dangling references (relation endpoints absent from the final inventory), both of which frequently arise when entity discovery and relation extraction are performed simultaneously. ObjectProperty triplets represent entity-to-entity relations of the form (e_s, p, e_o) with *p* a concise predicate phrase (e.g., “uses”, “targets”, “exploits”, “attributed to”) and an evidence sentence from the source text attached for grounding context during ontology alignment;

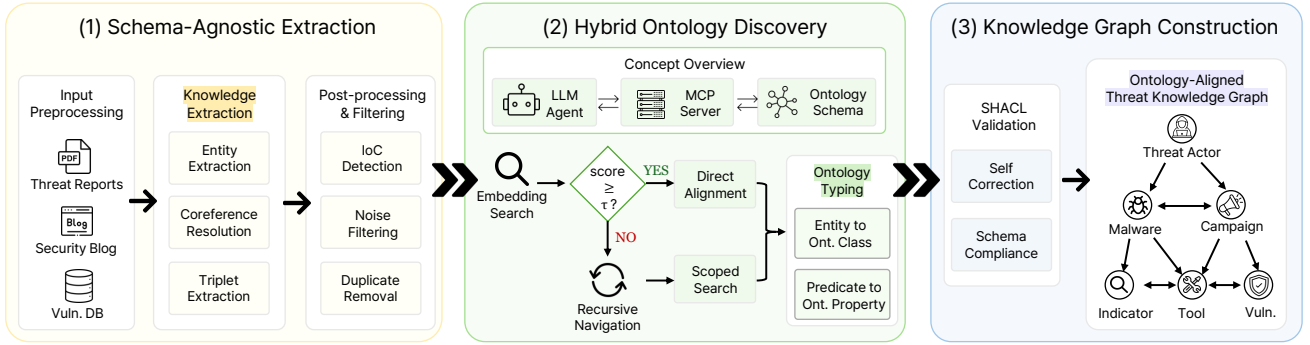


Fig. 2. Architecture of the ANCHOR System: (i) Schema-agnostic extraction preprocesses CTI documents and extracts entities, coreferences, and triplets without binding to a fixed ontology schema; (ii) Hybrid ontology discovery maps each entity and predicate to an ontology class or property URI via embedding search when confidence meets τ , or recursive navigation otherwise; (iii) Knowledge graph construction applies SHACL validation with closed-loop self-correction to build an ontology-aligned threat knowledge graph.

DatatypeProperty triplets capture entity-to-literal associations such as language strings, timestamps, or boolean flags. The LLM is instructed to be exhaustive, evaluating every entity pair and capturing all stated attributes regardless of salience.

3) *Post-processing and Filtering*: Following knowledge extraction, ANCHOR applies three deterministic post-processing steps: IoC detection, noise filtering, and duplicate removal. In IoC detection, ANCHOR matches entity names against regular expression patterns and overrides their type hints with a schema-agnostic descriptor (e.g., “IPv4 address”, “SHA-256 hash”), enabling reliable class resolution in the mapping stage regardless of which ontology is active. In noise filtering, ANCHOR removes non-entity strings such as temporal expressions, short strings, and common descriptors. Finally, in duplicate removal, ANCHOR merges duplicate entities sharing the same normalized key and removes duplicate or dangling triplets. The filtered output is then forwarded to the subsequent hybrid ontology discovery stage.

C. Hybrid Ontology Discovery

With the schema-agnostic knowledge extraction complete, ANCHOR aligns each extracted element to a formal ontology class and property. Hybrid ontology discovery provides this alignment through a uniform tool-based interface between the LLM agent and any OWL/SHACL ontology. At initialization, ANCHOR parses the target ontology file, pre-computes embeddings for all class names and descriptions, and indexes the type hierarchy. It exposes six tools covering three targets (classes, attributes, relations), each with two operations: (i) embedding-based search that retrieves candidates by semantic similarity, and (ii) hierarchical recursive navigation that the LLM invokes when search confidence is below the threshold τ .¹

1) *Entity Ontology Typing*: ANCHOR determines the appropriate formal ontology class for each extracted entity, representing it as a uniform resource identifier (URI), by executing

¹To facilitate broader reuse, the discovery suite is provided as an MCP-compatible server that includes the six core tools used in this work along with auxiliary functions for general-purpose ontology exploration. The underlying hybrid ontology discovery architecture, however, is agnostic to any specific tool-call protocol.

Algorithm 1. For this phase, the algorithm is instantiated using class-specific semantic search, the root class set U_{roots} , and owl:Thing as the default fallback.

Unlike predicate ontology typing, entity ontology typing executes only Steps 1 and 2 of the discovery process, with Algorithm 1 instantiated in ENTITY mode.

Step 1: Embedding-based search. The system first performs embedding-based search, which computes a relevance score for each candidate URI u in the ontology graph G :

$$\text{score}(u) = \frac{1}{2} \text{sim}(\mathbf{q}, \mathbf{e}_u^{\text{name}}) + \frac{1}{2} \text{sim}(\mathbf{q}, \mathbf{e}_u^{\text{desc}}) + \delta_k, \quad (1)$$

where \mathbf{q} is the query embedding derived from the query hint h (e.g., the entity’s *type_hint*), $\mathbf{e}_u^{\text{name}}$ and $\mathbf{e}_u^{\text{desc}}$ are pre-computed embeddings of the class name and description, $\text{sim}(\cdot)$ denotes cosine similarity, and δ_k is a fixed keyword bonus (set to 0.3 in our experiments) applied when the query string appears verbatim in the class name or description. If the top candidate’s score meets or exceeds the entity confidence threshold τ_{entity} (set to 0.45 in our experiments), the class is selected immediately.

Step 2: Hierarchical recursive navigation. If no candidate from Step 1 clears the threshold τ_{entity} , the system activates hierarchical recursive navigation. From the set of root classes U_{roots} , the LLM evaluates semantic definitions of each subclass tier and selects the most logically matching branch, drilling down iteratively until it reaches a leaf node or determines that no sufficiently matching branch remains. This top-down approach compensates for the limitations of embedding models by leveraging LLM reasoning, enabling precise classification of domain-specific or novel terminology. The LLM bases this decision on the candidate subclasses’ names, textual descriptions, and child counts presented at each tier, signaling termination when no candidate aligns with the target query hint h .

Fallback. If the recursive navigation fails to identify a suitable class, the entity defaults to owl:Thing, corresponding to the fallback return in Algorithm 1. This defensive assignment ensures the extracted entity and its associated relations remain

Algorithm 1: Hybrid Ontology Discovery

Input: Query hint h ; root URIs U_{roots} ; confidence threshold τ ; discovery mode $m \in \{\text{ENTITY}, \text{PREDICATE}\}$.

Output: Matched ontology element URI u^* .

```
1: /* Step 1: Embedding-based search */
2:  $U_{\text{cands}} \leftarrow \text{Search}(h)$ 
3: if  $U_{\text{cands}} \neq \emptyset$  and  $\text{score}(\text{top}(U_{\text{cands}})) \geq \tau$  then
4:   return  $\text{top}(U_{\text{cands}})$ 
5: end if
6: /* Step 2: Hierarchical recursive navigation */
7:  $u_{\text{curr}} \leftarrow \text{LlmSelect}(h, U_{\text{roots}}, \text{RetrieveDesc}(U_{\text{roots}}))$ 
8: if  $u_{\text{curr}} = \text{none}$  then
9:   return  $\text{Fallback}(m)$ 
10: end if
11:  $u_{\text{scope}} \leftarrow \emptyset$ 
12: while true do
13:    $S \leftarrow \text{Children}(u_{\text{curr}})$ 
14:   if  $S = \emptyset$  then
15:      $u_{\text{scope}} \leftarrow u_{\text{curr}}$ ; break
16:   end if
17:    $u_{\text{best}} \leftarrow \text{LlmSelect}(h, S, \text{RetrieveDesc}(S))$ 
18:   if  $u_{\text{best}} = \text{none}$  then
19:      $u_{\text{scope}} \leftarrow u_{\text{curr}}$ ; break
20:   end if
21:    $u_{\text{curr}} \leftarrow u_{\text{best}}$ 
22: end while
23: if  $m = \text{ENTITY}$  then
24:   return  $u_{\text{scope}}$ 
25: end if
26: /* Step 3: Scoped embedding search (predicate) */
27: if  $u_{\text{scope}} \neq \emptyset$  then
28:    $U_{\text{cands}} \leftarrow \text{Search}(h, \text{scope} = u_{\text{scope}})$ 
29:   if  $U_{\text{cands}} \neq \emptyset$  and  $\text{score}(\text{top}(U_{\text{cands}})) \geq \tau$  then
30:     return  $\text{top}(U_{\text{cands}})$ 
31:   end if
32: end if
33: return  $\text{Fallback}(m)$ 
```

in the knowledge graph without introducing an incorrect or unverified type.

2) *Predicate Ontology Typing*: To assign a formal property URI to each extracted triplet, ANCHOR extends the search-and-navigate strategy used in entity ontology typing. Depending on the triplet type, it determines either an ObjectProperty URI for entity-to-entity relations or a DatatypeProperty URI for entity-to-literal attributes.

Unlike entity ontology typing, this phase utilizes all three steps of Algorithm 1, with the algorithm instantiated in PREDICATE mode.

Step 1: Embedding-based search. ANCHOR deploys two distinct search tools to handle attributes and relations. For literal attributes, the embedding-based search identifies the optimal DatatypeProperty. It extends Equation 1 with a datatype inference heuristic, applying a score boost when the inferred XSD type (e.g., `xsd:dateTime`, `xsd:integer`) aligns with the

property’s declared XML Schema Definition (XSD) range. For entity connections, the relation search targets ObjectProperty URIs by utilizing the evidence sentence captured during triplet extraction as additional context. It applies direction-aware score adjustments, adding a bonus (0.10) for direct forward relations and a minor penalty (-0.05) for inverse relations. To ensure completeness, both tools evaluate properties inherited from the transitive superclass hierarchy, leveraging SHACL domain annotations back-propagated during schema loading.

Step 2: Hierarchical recursive navigation. If the search score falls below the predicate confidence threshold $\tau_{\text{predicate}}$ (set to 0.30), the system triggers LLM-guided traversal. The traversal starting point depends on the target: data-property navigation roots at the bound entity URI, while object-property navigation roots at the (s, o) URI pair. From these roots, the LLM navigates a structured property list organized by inheritance level and domain class. For object properties, the LLM additionally infers the correct assertion direction (forward or inverse).

Step 3: Scoped embedding search. Large schemas often group properties under intermediate classes, making exhaustive LLM traversal impractical (e.g., UCO contains 578 properties). When navigation reveals a collapsed property group anchored to an intermediate class u_{scope} , ANCHOR re-executes the embedding similarity search restricted solely to the properties of that class. This scoped pass allows the system to handle large hierarchies efficiently once anchored to a specific branch.

Fallback. If all three steps fail to identify a matching property, ANCHOR defaults to `rdfs:label` for literal attributes and `rdfs:seeAlso` for entity relations. Similar to entity ontology typing, this defensive mapping ensures every extracted triplet is preserved in the output graph without fabricating incorrect schema assignments.

D. Knowledge Graph Construction with SHACL Validation

After finalizing the entity and predicate mappings, ANCHOR assembles the typed elements into a unified knowledge graph. To ensure structural integrity, the system applies a SHACL-based validation mechanism that evaluates every entity against the node shapes defined in the target ontology. This validation targets cardinality constraints, verifying that required properties are present and that value counts fall within declared bounds. Because LLM-based extraction often omits mandatory attributes, missing required properties represent the most common source of schema violations in this domain.

For example, under the UCO schema, a cardinality violation occurs if an extracted malware entity lacks the required hash algorithm attribute, a common omission that renders the indicator unusable for downstream matching. Upon detecting a violation, the system generates a structured feedback message detailing the violating entity, the failed constraint, and the expected correction. The LLM agent receives this feedback and re-invokes the appropriate discovery tool to rectify the omission (e.g., searching for a missing required property). ANCHOR limits this closed-loop self-correction to three retries

File: buckeye-windows-zero-day-exploit

Beginning in March 2016, Buckeye began using a variant of DoublePulsar (Backdoor.DoublePulsar), a backdoor that was subsequently released by the Shadow Brokers in 2017. DoublePulsar was delivered to victims using a custom exploit tool (Trojan.Bemstour) that was specifically designed to install DoublePulsar. Bemstour exploits two Windows vulnerabilities in order to achieve remote kernel code execution on targeted computers. One vulnerability is a Windows zero-day vulnerability (CVE-2019-0703) discovered by Symantec. The second Windows vulnerability (CVE-2017-0143) was patched in March 2017 after it was discovered to have been used by two exploit tools – EternalRomance and EternalSynergy – that were also released as part of the Shadow Brokers leak.

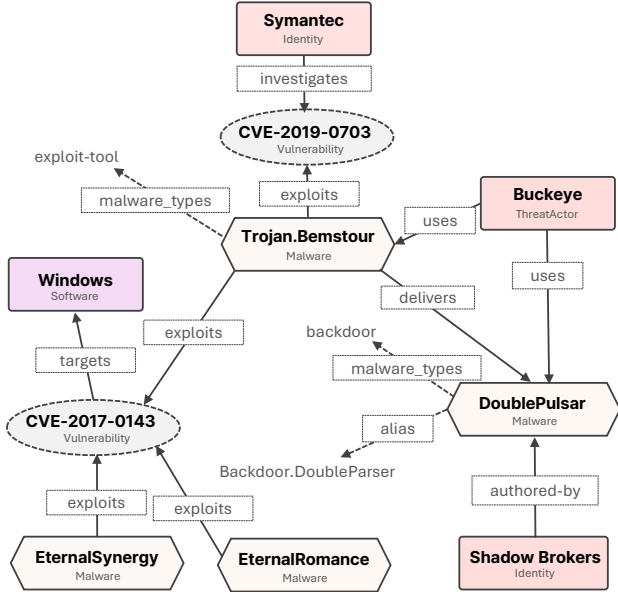


Fig. 3. Example knowledge graph constructed by ANCHOR from a Buckeye APT campaign report.

to prevent unbounded execution. If the violation persists after the maximum attempts, the system retains the entity with its assigned class and flags it with a validation warning rather than silently discarding it. This approach minimizes unverified type assignments while preserving the extracted intelligence.

Beyond constraint checking, these SHACL shapes facilitate facet discovery for ontologies that organize auxiliary properties into compound structures (e.g., UCO). The system performs this through a multi-strategy lookup combining property domain inspection, inheritance traversal, and SHACL shape resolution. Finally, ANCHOR serializes the validated type assignments as an ontology-aligned knowledge graph in JSON format.

To provide an intuitive understanding of how ANCHOR reconstructs semantic connectivity from an unstructured CTI report, as shown in Fig. 3, we visualize an example knowledge graph constructed from a Buckeye APT campaign report. The resulting graph contains 10 typed entities of five classes (ThreatActor, Identity, Malware, Software, and Vulnerability), connected by typed predicates (e.g., uses, delivers, and exploits) that trace the attack chain from the exploit tool through the vulnerabilities and the backdoor to the Shadow Brokers leak. The two vulnerabilities (CVE-2019-0703 and CVE-2017-0143) are typed against the Vulnerability class, demonstrating accurate resolution of numerical CVE refer-

TABLE II
STATISTICS OF TARGET ONTOLOGY SCHEMAS

Schema	Classes	Relations	Properties
UCO	419	177	578
STIX 2.1	109	92	311
MALOnt	75	10	12

ences. Datatype properties such as `malware_types` (“backdoor” and “exploit-tool”) and `alias` (“Backdoor.DoubleParser”) attach literal attributes to the entities. These typed entities, properties, and predicates jointly reconstruct the semantic connectivity discussed in Section III-A, including the causal chain, the temporal sequence, and the qualitative reasoning that indicator-oriented formats omit.

V. EVALUATION

For reproducibility, we use a benchmark provided by CTINexus [16], consisting of 149 CTI reports with manually annotated entities, triplets, and ontology type labels. The original benchmark targets only the small-scale MALOnt schema, so we additionally reconstruct ground-truth type labels for two larger schemas, UCO and STIX 2.1, as summarized in Table II. Three cybersecurity researchers established the ground truth through a human-in-the-loop process. An ensemble of three LLMs (GPT-5.4, Claude-Sonnet-4-6, and Gemini-3.1-flash) produced the initial candidates, and the researchers manually cross-examined and resolved conflicting assignments by consensus.

For our baselines, we compare the F1 scores of ANCHOR against three systems: TTPDrill [11], CTINexus [16], and LLM4CTI [17]. We exclude CTIKG [15], as it performs only knowledge extraction without ontology typing. Since CTINexus originally types only entities, we extend it to predicate typing by reusing its prompt-based entity ontology typing method on relation predicates. We deploy ANCHOR on a workstation equipped with an NVIDIA GB10 board and 128 GB of memory, where we serve Qwen3.5-35B locally via vLLM [26]. For a fair comparison, all baselines also use Qwen3.5-35B as their underlying model. In the comparison against enterprise LLMs (Section V-D), we additionally use GPT-5.4-mini and Claude Haiku-4.5.

A. Ontology Typing Performance

We examine the ontology typing performance on three schemas with different scales: UCO (large, 419 classes), STIX (medium, 109 classes), and MALOnt (small, 75 classes). Ontology typing measures whether the extracted entities and relations can be aligned to formal ontology classes and properties. Since the UCO schema is deeply nested with multi-level class hierarchies, we adopt a hierarchical F1 score that assigns full credit (1.0) for an exact match and partial credit (0.6 for a one-step parent or child mismatch, 0.3 for a two-step mismatch) to capture semantic proximity. We evaluate two complementary tasks: entity ontology typing, where each extracted entity is mapped to an ontology *class* uniform resource identifier (URI),

TABLE III
ENTITY ONTOLOGY TYPING PERFORMANCE ON THREE SCHEMAS

System	UCO	STIX	MALOnt	Average
TTPDrill [11]	0.0652	0.1042	0.2305	0.1333
CTINexus [16]	0.4439	0.5698	0.6370	0.5502
LLM4CTI [17]	0.4521	0.7886	0.6891	0.6432
ANCHOR	0.7347	0.8724	0.6942	0.7371

TABLE IV
PREDICATE ONTOLOGY TYPING PERFORMANCE ON THREE SCHEMAS

System	UCO	STIX	MALOnt	Average
TTPDrill [11]	0.0033	0.0000	0.0000	0.0011
CTINexus [16]	0.1282	0.4289	0.4528	0.3366
LLM4CTI [17]	0.4000	0.5355	0.5647	0.5001
ANCHOR	0.5180	0.5860	0.5412	0.5484

and predicate ontology typing, where each extracted relation predicate is mapped to an ontology *property* URI.

Entity ontology typing. For each extracted entity, the system selects the best-matching class URI from the target ontology. As shown in Table III, ANCHOR achieves the highest performance on every schema, with an average F1 of 0.7371. Specifically, ANCHOR outperforms the second-best baseline (LLM4CTI) by 62.5% (from 0.4521 to 0.7347) on the UCO schema. The margin shrinks on smaller schemas: ANCHOR outperforms LLM4CTI by 10.6% (from 0.7886 to 0.8724) on STIX and by 0.7% (from 0.6891 to 0.6942) on MALOnt. CTINexus, which includes the entire schema in the LLM prompt, suffers a 30.3% performance drop when scaling from MALOnt (0.6370) to UCO (0.4439). TTPDrill never exceeds 0.2305 on any schema because it relies on a simple rule-based pipeline.

We observe that this performance gap widens as the schema size grows, highlighting the contrast between prompt-based schema inclusion and the dynamic hybrid ontology discovery provided by ANCHOR. As the number of candidate classes increases from 75 in MALOnt to 419 in UCO, prompt-based baselines exhaust the LLM context window and degrade due to the “lost in the middle” phenomenon [25]. In contrast, ANCHOR retrieves only the relevant ontology fragments and remains stable on the larger schema. The small performance difference on MALOnt indicates that hybrid ontology discovery offers limited benefit when the schema fits easily within a single prompt. However, its advantage becomes highly pronounced on large schemas such as UCO, where prompt-based inclusion fails.

Predicate ontology typing. The system maps each extracted relation predicate to a formal property URI (either ObjectProperty or DatatypeProperty). As shown in Table IV, ANCHOR achieves the highest average F1 of 0.5484, outperforming the baselines on the UCO and STIX schemas. Specifically, ANCHOR outperforms LLM4CTI by 29.5% (from 0.4000 to 0.5180) on UCO and by 9.4% (from 0.5355 to 0.5860) on STIX. On MALOnt, ANCHOR scores 0.5412, which is 4.3% below LLM4CTI at 0.5647. TTPDrill records zero

TABLE V
ABLATION STUDY ON HYBRID ONTOLOGY DISCOVERY COMPONENTS

Configuration	Entity Ontology Typing	Predicate Ontology Typing
Search-Only	0.9184	0.3142
Recurse-Only	0.7914	0.6416
Hybrid (ours)	0.9364	0.7843

performance on STIX and MALOnt, since its template-based approach cannot produce property URIs outside the predefined ATT&CK vocabulary.

Predicate ontology typing is more difficult than entity ontology typing for all four systems: ANCHOR drops by 25.6% (from 0.7371 to 0.5484) and LLM4CTI drops by 22.2% (from 0.6432 to 0.5001). The gap is intuitive since a predicate’s correct property URI depends not only on the surface verb but also on the domain and range of its subject and object entities, and on the direction of the edge (e.g., *uses* vs. *used_by*). Single-word embedding similarity alone is therefore insufficient, and the schema-navigation component of hybrid ontology discovery becomes the main contributor to predicate ontology typing, as analyzed in Section V-B. The marginal underperformance of ANCHOR on MALOnt (0.5412 vs. 0.5647 for LLM4CTI) is consistent with our earlier observation that hybrid ontology discovery offers limited advantage on small schemas. The key findings from the ontology typing analysis are summarized as follows: (i) ANCHOR generalizes to large hierarchical schemas where prompt-based schema inclusion baselines collapse, and (ii) predicate typing remains a harder task than entity typing for all four systems, motivating the ablation study on hybrid ontology discovery in Section V-B.

B. Ablation Study

We isolate two design choices within hybrid ontology discovery: (i) the combination of embedding search and recursive schema navigation, and (ii) the embedding-similarity threshold τ that controls when the system switches from search to recursive traversal. For both ablations, we use ground-truth entities and triplets as fixed inputs so that the reported scores reflect only the typing performance of each configuration, while all other parameters (ontology schema, candidate inventory, scoring rule) remain unchanged.

1) *Effect of Hybrid Ontology Discovery:* We evaluate three configurations on both ontology typing tasks: Hybrid (ANCHOR), Search-Only, and Recurse-Only. The Hybrid configuration combines embedding-based search with hierarchical recursive navigation, while the others rely solely on one of these strategies.

As shown in Table V, the Hybrid configuration achieves the highest performance on both tasks, scoring 0.9364 on entity typing and 0.7843 on predicate typing, while the two single-strategy baselines exhibit asymmetric behavior. On entity ontology typing, Search-Only (0.9184) approaches the Hybrid score, whereas Recurse-Only falls to 0.7914. The pattern reverses on predicate ontology typing: Recurse-Only reaches 0.6416, while Search-Only collapses to 0.3142. Notably, the

Hybrid configuration improves performance by 22.2% (from 0.6416 to 0.7843) over Recurse-Only and more than doubles the score of Search-Only.

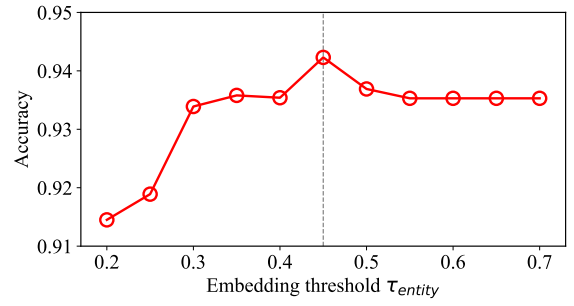
The contrasting tendency of the two tasks reflects a structural difference between entity and predicate ontology typing. Entity ontology typing maps a single named entity to a class, a problem that aligns well with semantic similarity over class names and descriptions, so embedding search alone suffices in most cases. In contrast, predicate ontology typing requires reasoning over the subject-object relation, the property hierarchy, and directional constraints that distinguish subject from object, all of which are structurally represented in the schema and accessed through recursive navigation rather than lexical similarity. We further quantify this contrast in Appendix A: instrumented measurements show that 91.0% of entity-class queries are resolved by embedding search alone, while 72.1% of ObjectProperty queries are routed to recursive navigation. Ultimately, the hybrid configuration effectively combines these strengths: embedding-based search provides a fast lexical entry point for entities, and hierarchical recursive navigation handles the structural reasoning needed for predicates.

2) *Effect of Embedding Threshold*: To evaluate the effect of the embedding thresholds, we analyze τ_{entity} for class search in entity ontology typing and $\tau_{\text{predicate}}$ for property search in predicate ontology typing. Each threshold controls when the system accepts an embedding search result rather than triggering a fallback to recursive navigation. The system accepts a candidate only if its similarity score reaches or exceeds the corresponding threshold. We sweep each threshold over $\{0.20, 0.25, 0.30, \dots, 0.70\}$ while keeping all other parameters fixed. We then measure the performance of the corresponding typing task.

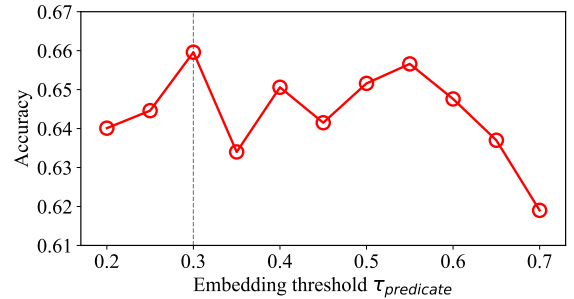
As shown in Fig. 4, the two tasks exhibit distinct optimal thresholds and sensitivity patterns. For entity ontology typing, performance peaks at $\tau_{\text{entity}} = 0.45$ (0.9423) and remains within 0.7% of this peak for $\tau_{\text{entity}} \in [0.30, 0.55]$. This performance converges to a constant 0.9353 at $\tau_{\text{entity}} \geq 0.55$, where every query triggers recursive navigation. For predicate ontology typing, performance peaks at $\tau_{\text{predicate}} = 0.30$ (0.6596) and is more sensitive to the threshold value. This task shows moderate fluctuations in the middle range and experiences a sharper degradation by 6.2% (from 0.6596 to 0.6190) at $\tau_{\text{predicate}} = 0.70$.

This difference in optimal thresholds stems from the distinct embedding similarity distributions of the two search targets. As detailed in Appendix A, entity-class queries demonstrate a high median similarity of 0.837, whereas ObjectProperty queries demonstrate a much lower median of 0.244. Class names and descriptions provide rich lexical signals, enabling the system to tolerate a stricter threshold for entity ontology typing. In contrast, property predicates offer weaker lexical signals, requiring a more permissive threshold to prevent the system from indiscriminately routing every query into recursive traversal.

Consequently, we adopt $\tau_{\text{entity}} = 0.45$ and $\tau_{\text{predicate}} = 0.30$ as the default configurations for all subsequent experiments.



(a)



(b)

Fig. 4. Effect of the embedding thresholds on ontology typing performance: (a) entity ontology typing and (b) predicate ontology typing.

TABLE VI
SCHEMA NON-COMPLIANCE RATE ON UCO SCHEMA

System	Total Items	Violations	Non-compliance (%)
TTPDrill [11]	3,784	2,868	75.8
CTINexus [16]	10,382	4,556	43.9
LLM4CTI [17]	3,881	1,950	50.2
ANCHOR	3,665	191	5.2

Note that optimal values may vary depending on the target ontology and downstream task. The key findings from this ablation study are summarized as follows: (i) embedding-based search and hierarchical recursive navigation are fundamentally complementary, where embedding search drives entity ontology typing and recursive navigation drives predicate ontology typing, and (ii) the asymmetric lexical richness between classes and properties dictates distinct similarity thresholds for optimal performance.

C. Schema Compliance

To investigate whether the constructed knowledge graph follows the formal structural rules of the target ontology, we measure the non-compliance rate over the full set of extracted entities and triplets on UCO. This measurement is essential because downstream RDF/OWL reasoning systems reject malformed inputs at load time. An item fails if it violates at least one of two checks: (i) URI and namespace conformance, and (ii) SHACL constraints declared by the schema. These constraints cover required-attribute counts, datatype rules, malformed URIs, and missing required properties.

As shown in Table VI, ANCHOR achieves the lowest non-compliance rate of 5.2%. Specifically, ANCHOR improves performance over the second-best baseline (CTINexus) by 88.2% (from 43.9% to 5.2%). CTINexus and LLM4CTI emit class-like labels that satisfy URI conformance for common entity types. However, they skip schema-level checks, which causes missing required-attribute violations and datatype violations to dominate their failure counts. TTPDrill produces rule-based templates with identifiers that diverge from the canonical UCO namespaces. This divergence inflates URI conformance violations in addition to the required-attribute failures observed in the LLM-based baselines.

This performance gap arises from the closed-loop SHACL validation described in Section IV-D. The baseline systems [11], [16], [17] do not perform this validation. ANCHOR re-checks each predicted type against the required-attribute counts and datatype rules declared by the target schema before commitment. The system re-invokes the discovery tool when it detects a violation. We limit this self-correction to three retries to prevent unbounded execution. The system records any remaining violation as a validation warning rather than silently dropping it. Most violations in the LLM-based baselines are missing-attribute failures, such as a malware entity lacking the required hash algorithm attribute. The closed-loop retry addresses these failures by routing a targeted property search back through the discovery tool.

The 5.2% of non-compliant items in ANCHOR represent violations that remain after the three-retry budget. These items typically correspond to exceptional cases where no candidate property satisfies the missing required attribute. ANCHOR retains these items with their assigned class and a validation marker to preserve the extracted intelligence for downstream review. A graph with 5.2% marked items can be loaded into an RDF triplestore with minimal cleanup. In contrast, the high violation rates of the baselines require pre-validation of every downstream query or manual cleanup of 40% to 75% of the items before reasoning is possible.

D. Local LLM Feasibility

A primary design goal of ANCHOR is the integration of external CTI with sensitive internal incident data without exposing the data to enterprise LLM APIs. To demonstrate this capability, we evaluate whether ANCHOR preserves competitive typing performance with locally hosted open-source backbones. We compare four LLM backbones under two conditions: with ANCHOR and without ANCHOR (i.e., naive prompt-based ontology typing). The local backbones are Qwen3.5-35B and Gemma-4-26B, served through vLLM on the local workstation. The enterprise backbones are GPT-5.4-mini and Claude Haiku-4.5, accessed through their official APIs. As in Section V-B, we use ground-truth entities and triplets as fixed inputs so that the measured performance reflects only the typing stage.

As shown in Table VII, ANCHOR improves typing performance for every backbone. Specifically, for Qwen3.5-35B, Gemma-4-26B, GPT-5.4-mini, and Claude Haiku-4.5,

TABLE VII
EFFECT OF ANCHOR ON LOCAL AND ENTERPRISE LLM BACKBONES

Backbone	Discovery	Entity	Predicate
Qwen3.5-35B (local)	w/o ANCHOR	0.4284	0.1491
	w/ ANCHOR	0.9364	0.7843
Gemma-4-26B (local)	w/o ANCHOR	0.3905	0.2849
	w/ ANCHOR	0.9198	0.6657
GPT-5.4-mini (enterprise)	w/o ANCHOR	0.4540	0.1581
	w/ ANCHOR	0.9410	0.8012
Claude Haiku-4.5 (enterprise)	w/o ANCHOR	0.5412	0.4318
	w/ ANCHOR	0.9439	0.8021

ANCHOR improves entity ontology typing by 118.6% (from 0.4284 to 0.9364), 135.5% (from 0.3905 to 0.9198), 107.3% (from 0.4540 to 0.9410), and 74.4% (from 0.5412 to 0.9439), respectively. Predicate ontology typing exhibits the same pattern for all four backbones. The largest absolute improvements appear on local backbones, where the naive prompt-based scores are the lowest (Table VII). The strongest enterprise configuration (Claude Haiku-4.5 with ANCHOR) reaches 0.9439 on entity ontology typing and 0.8021 on predicate ontology typing. Meanwhile, the strongest local configuration (Qwen3.5-35B with ANCHOR) reaches 0.9364 and 0.7843. This local configuration retains 99.2% ($= \frac{0.9364}{0.9439}$) and 97.8% ($= \frac{0.7843}{0.8021}$) of the performance of the best enterprise model, respectively.

We observe that Qwen3.5-35B with ANCHOR (entity typing of 0.9364 and predicate typing of 0.7843) outperforms Claude Haiku-4.5 without ANCHOR (entity typing of 0.5412 and predicate typing of 0.4318) by 1.73 \times and 1.82 \times , respectively. This result indicates that the integration of ANCHOR into a local backbone delivers a larger gain than upgrading the backbone from local to enterprise. The four backbones converge to a narrow performance band with ANCHOR. This convergence occurs because the system offloads the heaviest part of the reasoning, such as the navigation of large class and property hierarchies under formal constraints, from the LLM to the schema itself. Consequently, the LLM is left with a small bounded selection problem. This finding demonstrates that for schema-grounded reasoning tasks such as ontology typing, the design of the discovery pipeline matters more than the raw capability of the underlying LLM. The key findings of the feasibility analysis are summarized as follows: (i) ANCHOR offloads structural reasoning from the LLM to the schema, which reduces the performance gap between local and enterprise backbones, and (ii) local deployment with ANCHOR retains 99.2% (entity) and 97.8% (predicate) of the best enterprise LLM performance. This level of performance supports privacy-preserving CTI knowledge graph construction without a loss of typing fidelity.

VI. RELATED WORK

The automation of structured knowledge extraction from CTI reports has been an active research topic. We organize

existing studies into three categories: rule-based pipelines, classification-based approaches, and LLM-based methods.

Rule-based pipelines. Husari *et al.* [11] proposed TTPDrill, which mapped threat actions to MITRE ATT&CK patterns through natural language processing (NLP) over predefined templates. Satvat *et al.* [12] introduced EXTRACTOR, which built attack graphs through dependency parsing and heuristic rules. While these systems demonstrated automated CTI extraction, they rely on hand-crafted rules that require manual updates when new attack patterns appear.

Classification-based approaches. Later studies adopted machine learning models for CTI extraction. Alam *et al.* [13] presented LADDER, a BERT-based entity classifier for attack pattern recognition. Peng *et al.* [14] proposed a retrieval-augmented named entity recognition (NER) pipeline with adaptive instructions. Mouiche and Saad [27] combined SecureBERT [28] with a BiLSTM joint extractor for entity and relation extraction. Piplai *et al.* [29] applied NER and fused the results into the UCO knowledge graph. These methods improved upon rule-based systems, but they remain bound to fixed type inventories defined during training. This limitation reduces their applicability when ontology schemas change.

LLM-based methods. Recent work has explored LLM-based pipelines for CTI knowledge extraction. Huang *et al.* [15] proposed CTIKG, which used GPT-4 with in-context learning (ICL) to extract open-ended knowledge graphs from CTI reports while deliberately avoiding fixed ontology schemas. Cheng *et al.* [16] proposed CTINexus, which adopted a similar ICL approach but aligned extracted triplets to the MALOnt ontology through in-context schema inclusion. However, this design does not scale to large ontologies such as UCO. LLM4CTI [17] introduced a chunk-wise dual-context framework with GNN-based link prediction, but it relies on a fixed, custom-defined schema and does not support formal ontology alignment. Other LLM-based systems for CTI knowledge graph construction (TRACE [30], CTI-Thinker [31], AttackKG+ [32], LLM-TIKG [33]) and general-purpose sequence-to-sequence extractors (KnowGL [34], REBEL [35]) remain bound to fixed schemas or require task-specific fine-tuning. Furthermore, none of these systems addresses cybersecurity-specific ontology constraints. None of the above systems validates type assignments against a formal schema. This omission causes hallucinated mappings to enter the downstream knowledge graph without verification.

In summary, prior systems either bind to a fixed ontology during pipeline design or rely on in-context schema inclusion. Furthermore, none of these systems verifies ontology typing against formal schema constraints. Therefore, ANCHOR has addressed both limitations through schema-agnostic hybrid ontology discovery with SHACL-based validation.

VII. DISCUSSION

In this section, we discuss the operational trade-offs of local deployment, the structural boundaries of static ontologies, and the rationale behind our ontology typing mechanism.

Practical deployment in security operations. The local-deployment configuration (Section V-D) directly addresses privacy regulations (e.g., GDPR, HIPAA) that restrict sensitive data sharing. Our local backbones achieve nearly the same typing quality as enterprise LLMs, making this privacy-preserving setup a practical default rather than a performance compromise. The primary trade-off is increased processing latency due to multiple LLM calls per entity. However, in threat intelligence operations where reports are processed asynchronously, this latency is a reasonable cost to maintain complete control over sensitive data.

Ontology schema augmentation. Schema-aligned extraction is inherently bounded by the target schema’s expressiveness. For example, when encountering novel entities like newly registered CVEs, the schema often lacks a suitable class. In these cases, ANCHOR conservatively defaults to *owl:Thing* rather than assigning an imprecise or fabricated type. This fallback mechanism highlights a fundamental limitation of mapping dynamic text to a static ontology: a system cannot assign a concept that the schema does not define. Future work will explore ontology schema augmentation methods, dynamically generating new classes for concepts that repeatedly trigger this fallback to relieve static schema constraints.

Confidence-based typing. ANCHOR rejects mappings below specific confidence thresholds (τ_{entity} and $\tau_{\text{predicate}}$), defaulting unverified items to *owl:Thing* or generic properties. This prevents low-confidence guesses from corrupting the knowledge graph, which is essential to avoid misleading threat attribution in security operations. While we establish default thresholds ($\tau_{\text{entity}} = 0.45$, $\tau_{\text{predicate}} = 0.30$) based on Section V-B, these are empirical baselines. Operators should recalibrate these values depending on the target ontology’s complexity and the specific downstream analytical objective.

VIII. ETHICAL CONSIDERATIONS

ANCHOR is designed as a defensive analysis tool to structure and integrate publicly available cyber threat intelligence. All data used in this study originate from open-source threat reports that are freely accessible to the public. The system does not discover or exploit new vulnerabilities. Its primary purpose is to assist security analysts in organizing threat knowledge rather than enabling offensive operations.

IX. CONCLUSION

We have presented ANCHOR, a schema-agnostic CTI knowledge graph construction framework. ANCHOR has addressed the limitations of existing LLM-based extraction pipelines, such as schema dependency, scalability issues on large ontologies, and the privacy risks of enterprise LLMs. ANCHOR introduces a *hybrid ontology discovery* mechanism that combines embedding-based search with LLM-guided hierarchical recursive navigation. Furthermore, a SHACL-based closed-loop validation enforces schema-compliant ontology typing, reducing hallucinated mappings in the output graph. Experimental results on UCO, STIX, and MALOnt demonstrated that ANCHOR improves UCO entity and predicate ontology

typing by 62.5% and 29.5%, while decreasing the schema non-compliance rate from 43.9% to 5.2%. When paired with a locally deployed open-source LLM, ANCHOR maintains 99.2% (entity) and 97.8% (predicate) of the typing performance of the best enterprise LLM, supporting privacy-preserving CTI analysis without a loss of typing fidelity. In future work, we will explore cross-ontology translation, where the schema-agnostic reasoning of ANCHOR maps threat knowledge between disparate schemas to facilitate seamless inter-organization CTI sharing.

REFERENCES

- [1] N. Sun, M. Ding, J. Jiang, W. Xu, X. Mo, Y. Tai, and J. Zhang, "Cyber threat intelligence mining for proactive cybersecurity defense: A survey and new perspectives," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1748–1774, 2023.
- [2] OASIS Cyber Threat Intelligence Technical Committee, "STIX version 2.1: OASIS standard," <https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html>, 2021, accessed: 2026-03-31.
- [3] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on workshop on information sharing and collaborative security*, 2016, pp. 49–56.
- [4] P. Gao, X. Liu, E. Choi, B. Soman, C. Mishra, K. Farris, and D. Song, "A system for automated open-source threat intelligence gathering and management," in *Proceedings of the 2021 International conference on management of data*, 2021, pp. 2716–2720.
- [5] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "Attackg: Constructing technique knowledge graph from cyber threat intelligence reports," in *European symposium on research in computer security*. Springer, 2022, pp. 589–609.
- [6] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 755–766.
- [7] Z. Syed, A. Padia, M. L. Mathews, T. Finin, A. Joshi et al., "Uco: A unified cybersecurity ontology," in *Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security*, 2016, pp. 195–202.
- [8] M. Iannacone, S. Bohn, G. Nakamura, J. Gerth, K. Huffer, R. Bridges, E. Ferragut, and J. Goodall, "Developing an ontology for cyber security knowledge graphs," in *Proceedings of the 10th annual cyber and information security research conference*, 2015, pp. 1–4.
- [9] N. Rastogi, S. Dutta, M. J. Zaki, A. Gittens, and C. Aggarwal, "Malont: An ontology for malware threat intelligence," in *International workshop on deployable machine learning for security defense*. Springer, 2020, pp. 28–44.
- [10] M. Adach, K. Hänninen, and K. Lundqvist, "Security ontologies: A systematic literature review," in *International Conference on Enterprise Design, Operations, and Computing*. Springer, 2022, pp. 36–53.
- [11] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd annual computer security applications conference*, 2017, pp. 103–115.
- [12] K. Satvat, R. Gjomemo, and V. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," *arXiv preprint arXiv:2104.08618*, 2021.
- [13] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking beyond iocs: Automatically extracting attack patterns from external cti," in *Proceedings of the 26th international symposium on research in attacks, intrusions and defenses*, 2023, pp. 92–108.
- [14] J. Peng, H. Sun, X. Tian, C. Huang, Z. Li, and R. Yan, "From retrieval to reasoning: A framework for cyber threat intelligence ner with explicit and adaptive instructions," *arXiv preprint arXiv:2512.19414*, 2025.
- [15] L. Huang and X. Xiao, "Ctikg: Llm-powered knowledge graph construction from cyber threat intelligence," in *First Conference on Language Modeling*, 2024.
- [16] Y. Cheng, O. Bajaber, S. A. Tsegai, D. Song, and P. Gao, "Ctinexus: Automatic cyber threat intelligence knowledge graph construction using large language models," in *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2025, pp. 923–938.
- [17] L. Huang, M. Zhang, Z. Chen, S. Ma, Z. Liu, Y. Ye, and X. Xiao, "Llm4cti: Uncovering security entities and their interactions from unstructured cyber threat intelligence."
- [18] M. Büchel, T. Paladini, S. Longari, M. Carminati, S. Zanero, H. Binyamini, G. Engelberg, D. Klein, G. Guizzardi, M. Caselli et al., "Sok: Automated ttp extraction from cti reports—are we there yet?" in *34th USENIX security symposium (USENIX Security 25)*, 2025, pp. 4621–4641.
- [19] D. Preuveneers and W. Joosen, "An ontology-based cybersecurity framework for ai-enabled systems and applications," *Future internet*, vol. 16, no. 3, p. 69, 2024.
- [20] T. D. Wagner, K. Mahub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: Survey and research directions," *Computers & Security*, vol. 87, p. 101589, 2019.
- [21] OASIS Cyber Threat Intelligence Technical Committee, "TAXII version 2.1: OASIS standard," <https://docs.oasis-open.org/cti/taxii/v2.1/taxii-v2.1.html>, 2021, accessed: 2026-03-31.
- [22] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [23] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat et al., "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [24] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model context protocol (mcp): Landscape, security threats, and future research directions," *ACM Transactions on Software Engineering and Methodology*, 2025.
- [25] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the association for computational linguistics*, vol. 12, pp. 157–173, 2024.
- [26] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.
- [27] I. Mouiche and S. Saad, "Entity and relation extractions for threat intelligence knowledge graphs," *Computers & Security*, vol. 148, p. 104120, 2025.
- [28] E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "Securebert: A domain-specific language model for cybersecurity," in *international conference on security and privacy in communication systems*. Springer, 2022, pp. 39–56.
- [29] A. Piplai, S. Mittal, M. Abdelsalam, M. Gupta, A. Joshi, and T. Finin, "Knowledge enrichment by fusing representations for malware threat intelligence and behavior," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2020, pp. 1–6.
- [30] Z. Xu, Z. Ning, T. Hu, J. Zhuge, Y. Wang, J. Cao, and M. Xu, "Trace: Timely retrieval and alignment for cybersecurity knowledge graph construction and expansion," *arXiv preprint arXiv:2602.11211*, 2026.
- [31] X. Yang, R. Zhong, Y. Chen, G. Peng, D. Yao, C. Chen, C. Wang, D. Zhang, Y. Zhou, and Z. Yang, "Cti-thinker: an llm-driven system for cti knowledge graph construction and attack reasoning," *Cybersecurity*, vol. 9, no. 1, p. 106, 2026.
- [32] Y. Zhang, T. Du, Y. Ma, X. Wang, Y. Xie, G. Yang, Y. Lu, and E.-C. Chang, "Attackg+: Boosting attack graph construction with large language models," *Computers & Security*, vol. 150, p. 104220, 2025.
- [33] Y. Hu, F. Zou, J. Han, X. Sun, and Y. Wang, "Llm-tikg: Threat intelligence knowledge graph construction utilizing large language model," *Computers & Security*, vol. 145, p. 103999, 2024.
- [34] G. Rossiello, M. F. M. Chowdhury, N. Mihindukulasooriya, O. Cornec, and A. M. Gliozzo, "Knowgl: Knowledge generation and linking from text," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, 2023, pp. 16476–16478.
- [35] P.-L. H. Cabot and R. Navigli, "Rebel: Relation extraction by end-to-end language generation," in *Findings of the association for computational linguistics: EMNLP 2021*, 2021, pp. 2370–2381.

APPENDIX A

DETAILS OF HYBRID ONTOLOGY DISCOVERY

The ablation study in Section V-B shows that entity ontology typing and predicate ontology typing draw on different

components of hybrid ontology discovery. We analyze this difference by recording, for each typing query, whether the final answer comes from the embedding-based search step or from the hierarchical recursive navigation step. We also measure the top-1 search similarity to indicate how reliably embedding similarity resolves each phase.

TABLE VIII
TYPING QUERIES RESOLVED BY SEARCH VS. RECURSIVE NAVIGATION

Phase	Search	Recurse
Entity (class)	91.0%	9.0%
DatatypeProperty	54.4%	45.6%
ObjectProperty	27.9%	72.1%
Total	62.3%	37.7%

Resolution by phase. Table VIII shows that the two phases of hybrid ontology discovery are exercised at very different rates depending on the typing target. The search-resolution rates for entity class, DatatypeProperty, and ObjectProperty queries are 91.0%, 54.4%, and 27.9%, respectively, while the remaining 9.0%, 45.6%, and 72.1% are answered by recursive navigation. Entity typing and ObjectProperty typing sit at opposite ends of this routing: the search-resolution rate drops by 69.3% (from 91.0% to 27.9%) between the two phases. DatatypeProperty typing sits between the two extremes at 54.4% search and 45.6% recurse, because literal-valued properties carry stronger lexical signals than relations but weaker structural cues than entity classes. Aggregated over the three phases, 62.3% of typing queries are answered by search and 37.7% by recursive navigation, confirming that neither component dominates the workload.

TABLE IX
TOP-1 EMBEDDING SIMILARITY AND THRESHOLD CLEARANCE RATES

Phase	Median	Mean	Above τ
Entity (class)	0.837	0.856	92.9%
DatatypeProperty	0.507	0.512	60.0%
ObjectProperty	0.244	0.300	13.2%
All	0.624	0.702	57.9%

Embedding similarity distribution. As shown in Table IX, the top-1 similarity statistics align with the routing pattern in Table VIII. The median top-1 similarities for entity-class, DatatypeProperty, and ObjectProperty queries are 0.837, 0.507, and 0.244, respectively, and the means follow the same ordering at 0.856, 0.512, and 0.300. The threshold-clearance rates differ even more sharply: 92.9% of entity-class queries exceed $\tau_{\text{entity}} = 0.45$, whereas only 13.2% of ObjectProperty queries exceed $\tau_{\text{predicate}} = 0.30$. ObjectProperty typing depends on the domain-range pair of subject and object entities rather than on the surface verb, so embedding similarity over surface forms is a weak signal, which is consistent with the drop in median top-1 similarity from 0.837 for entity-class queries to 0.244 for ObjectProperty queries. DatatypeProperty queries

clear the predicate threshold at 60.0%, sitting between the two extremes and matching the intermediate 54.4% search-resolution rate in Table VIII.

Explanation of the ablation study. Together, Tables VIII and IX explain why the ablation study in Section V-B affects entity typing and predicate typing so differently. Search-Only performs well on entity ontology typing because 92.9% of entity-class queries already clear τ_{entity} by embedding similarity, but it loses most of predicate ontology typing because only 13.2% of ObjectProperty queries clear $\tau_{\text{predicate}}$ and the remaining 86.8% have no recursive navigation route available. Recurse-Only partially recovers the ObjectProperty case because recursive navigation can travel the domain-range structure, but it underperforms on entity ontology typing because it discards the median similarity of 0.837 lexical signal in favor of a longer LLM-driven navigation. The full hybrid configuration (ANCHOR) combines the two complementary signals: search resolves the lexically dominated portion (62.3% of all queries), and recursive navigation handles the schema-driven remainder (37.7%).