Enabling Transparent Cyber Threat Intelligence Combining Large Language Models and Domain Ontologies

Luca, Cotti[1,*], Anisa, Rula[1], Devis, Bianchini[1] and Federico, Cerutti[1,2,3]

[1]Department of Information Engineering, University of Brescia, Italy
[2]School of Computer Science and Informatics, Cardiff University, United Kingdom
[3]Department of Electronics and Computer Science, University of Southampton, United Kingdom

Abstract

Effective Cyber Threat Intelligence (CTI) relies upon accurately structured and semantically enriched information extracted from cybersecurity system logs. However, current methodologies often struggle to identify and interpret malicious events reliably and transparently, particularly in cases involving unstructured or ambiguous log entries. In this work, we propose a novel methodology that combines ontology-driven structured outputs with Large Language Models (LLMs), to build an Artificial Intelligence (AI) agent that improves the accuracy and explainability of information extraction from cybersecurity logs. Central to our approach is the integration of domain ontologies and SHACL-based constraints to guide the language model's output structure and enforce semantic validity over the resulting graph. Extracted information is organized into an ontology-enriched graph database, enabling future semantic analysis and querying. The design of our methodology is motivated by the analytical requirements associated with honeypot log data, which typically comprises predominantly malicious activity. While our case study illustrates the relevance of this scenario, the experimental evaluation is conducted using publicly available datasets. Results demonstrate that our method achieves higher accuracy in information extraction compared to traditional prompt-only approaches, with a deliberate focus on extraction quality rather than processing speed.

Keywords

Cyber Threat Intelligence, Knowledge Graphs, Large Language Models, AI Agents

1. Introduction

The increasing complexity of cyber threats has led to a greater reliance on intelligence-oriented approaches within cybersecurity [1, 2, 3]. CTI contributes to this development by converting unstructured data into structured forms that support threat detection, analysis, and response (Section 2). Among available data sources, system logs—particularly those collected from honeypot or honeynet deployments—are especially relevant due to their significant concentration of adversarial activity [4]. Unlike routine operational environments, which generate logs dominated by benign entries, honeypots are designed to attract malicious interactions, making the resulting data more relevant for analysis.

Log data remains challenging to process. It is typically unstructured, syntactically inconsistent, and often ambiguous in meaning, which complicates automated interpretation. Traditional techniques based on fixed rules or heuristics tend to be limited in adaptability and are sensitive to variation in threat behavior. More recent methods involving LLMs have shown improved capabilities in extracting information from natural language [5, 6, 7]. However, they remain general in scope and can lack precision when applied to domain-specific content.

This work outlines a methodology (Section 3), named "OntoLogX", aimed at facilitating more transparent extraction of CTI from logs. The approach incorporates prompt-driven interaction with LLMs, guided by a domain-specific ontology designed to represent the structural and contextual characteristics

XAI-KRKG@ECAI25: First International ECAI Workshop on eXplainable AI, Knowledge Representation and Knowledge Graphs, October 25–30, 2025, Bologna, Italy
*Corresponding author.
luca.cotti@unibs.it (L. Cotti)
0009-0004-6351-556X (L. Cotti); 0000-0002-8046-7502 (A. Rula); 0000-0002-7709-3706 (D. Bianchini); 0000-0003-0755-0358 (F. Cerutti)

of cybersecurity-relevant data. The extracted information is structured into a Knowledge Graph (KG), enabling semantic querying, traceability, and integration with CTI workflows. The use of natural language interfaces may contribute to greater transparency in processing, as the logic of data extraction can be expressed and modified through prompts. The ontology also supports the organization of extracted information into knowledge graphs and enables the enforcement of quality constraints through Shapes Constraint Language (SHACL)-based validation.

We demonstrate that the proposed approach significantly improves information extraction accuracy over a traditional, prompt-only baseline method (Section 4). While the methodological design of this research draws on the characteristics of honeypot data, the experimental evaluation is conducted using publicly available log datasets to ensure reproducibility and comparability. Extracted intelligence is stored in an ontology-enriched graph database, enabling future semantic querying and downstream CTI tasks.

We conclude the paper by outlining directions for future work (Section 5). While the present methodology demonstrates accuracy in information extraction from cybersecurity logs, further development is required to ensure operational relevance for CTI analysis.

2. Background

2.1. The Need for Cyber Threat Intelligence

The escalation in complexity, sophistication, and frequency of cyber threats has increasingly exposed the limitations of traditional reactive cybersecurity strategies, necessitating a shift towards proactive and anticipatory methods [1, 2]. CTI addresses this need by systematically collecting, transforming, and analyzing vast amounts of threat-related data, transforming it into actionable knowledge capable of supporting informed decision-making processes within cybersecurity operations [3]. Effective CTI facilitates not only threat detection and mitigation but also proactive threat hunting and risk management, thus significantly enhancing organizational cyber resilience [8].

A critical source of CTI derives from honeypots or honeynets, i.e. deliberately vulnerable systems explicitly designed to attract and capture malicious cyber activity. Logs produced by these systems inherently record interactions from malicious actors, offering a unique dataset rich with insights into attacker behavior, tactics, techniques, and emerging threat trends [9]. Unlike traditional security logs, which are often overloaded with benign or irrelevant entries, honeypot-generated logs predominantly represent genuine threats, making them particularly valuable for focused, accurate intelligence extraction and threat analysis [4].

However, despite their value, honeypot logs pose significant analytical challenges due to their inherently heterogeneous, unstructured, and semantically ambiguous nature [10]. Traditional heuristic-based or pattern-matching approaches struggle to reliably extract comprehensive and accurate information from these logs, particularly when dealing with sophisticated attacks exhibiting novel or evolving characteristics. Consequently, there is an explicit requirement for more advanced methods capable of interpreting and structuring these logs accurately and semantically, ensuring the reliability and efficacy of derived CTI.

This challenge underscores the critical importance of innovative methodologies, such as LLMs combined with semantic technologies [11]. The integration of ontology-enriched representations into CTI workflows significantly enhances semantic clarity and precision, enabling more accurate interpretation, classification, and contextualization of threats derived from honeypot logs. Leveraging such structured semantic frameworks ultimately improves the overall quality and actionable value of threat intelligence, substantially reinforcing cybersecurity preparedness and response [12, 13].

2.2. Fundamental Concepts of Ontologies and Knowledge Graphs

Ontologies are commonly defined as formal, explicit specifications of shared conceptualizations encompassing classes, relationships, and constraints within a given domain [14]. In cybersecurity, ontology-

based frameworks are increasingly adopted to organize and standardize threat-related knowledge, enabling semantic interoperability, automated reasoning, and improved information integration across heterogeneous sources [12].

Many widely adopted cybersecurity ontologies are available in the literature, each addressing specific needs within CTI analysis. Unified Cyber Ontology (UCO) [12] provides comprehensive concepts and relationships for broad cybersecurity knowledge integration; Structured Threat Information Expression (STIX) [13] defines a widely adopted standard for CTI exchange; CRATELO [15] specifically targets the representation of cyber incidents and forensic data; Malware Information Sharing Platform (MISP) [16] facilitates structured malware and threat indicator sharing. Particularly relevant to our use case is the SEPSES ontology [17], which provides a rich, structured vocabulary for the semantic integration of already-parsed logs, but is not intended to guide information extraction from free-text messages or support interaction with language models.

The use case of real-time extraction from log messages using language models, however, benefits from an ontology that is minimal, interpretable, and structurally constrained for validation. Ontologies that rely on preprocessed observables, normalized fields, or syntactically consistent event abstractions, do not adapt easily to this task. To address this, we introduce a lightweight ontology designed for structured output parsing and SHACL-conformant triple generation. While it does not aim for the expressive depth of standard cybersecurity ontologies, it serves as an intermediate semantic layer suitable for direct anchoring in LLM workflows and subsequent alignment with established models.

A KG can be defined as a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities [18]. In practice, KGs are typically grounded in an ontology that ensures semantic consistency and facilitates reasoning, validation, and integration of heterogeneous data sources.

The quality and ontology-compliance of KGs can be verified through the use of SHACL [19] constraints. SHACL provides a declarative framework to define and validate structural and semantic constraints. When integrated with ontology-based representations, SHACL enables automated quality control over knowledge, verifying whether generated KGs respect required types, property cardinalities, and relationship patterns. This validation process helps detect inconsistencies, enforce domain rules, and ensure that the resulting KGs remain semantically coherent and human-auditable, even when derived from noisy or inferred inputs. In the context of log analysis, SHACL plays a vital role in guaranteeing that automatically generated semantic representations conform to expected schema designs, thereby enhancing both the robustness and explainability of downstream CTI tasks [20, 21].

2.3. Large Language Models, Retrieval Augmented Generation, and AI Agents

LLMs are transformer-based models trained on extensive textual corpora to perform a broad range of natural language understanding and generation tasks [22, 23, 24]. These models learn probabilistic representations of language, enabling them to complete, summarize, translate, and interpret text across multiple domains. Their performance has been demonstrated across benchmarks in zero-shot and few-shot learning [7]. However, despite their versatility, LLMs do not inherently guarantee factual consistency, structural coherence, or domain-specific accuracy. Outputs may reflect biases in the training data or lack sufficient grounding in external knowledge, particularly when applied to specialized domains such as cybersecurity, where terminology and contextual interpretation are often critical.

One strategy to address these limitations is Retrieval Augmented Generation (RAG), which combines language generation with information retrieval to enhance factual grounding [6]. In the RAG framework, a retriever component selects documents relevant to a given query from a pre-indexed knowledge base. The retrieved content is then passed to a language model, which conditions its output on this external evidence. This method allows for more contextually informed and domain-relevant responses, especially in scenarios where training data alone may be insufficient or outdated [5, 25]. In the context of cybersecurity, it offers the potential to improve information extraction from unstructured sources such as system logs, where background knowledge may be necessary to interpret ambiguous or incomplete

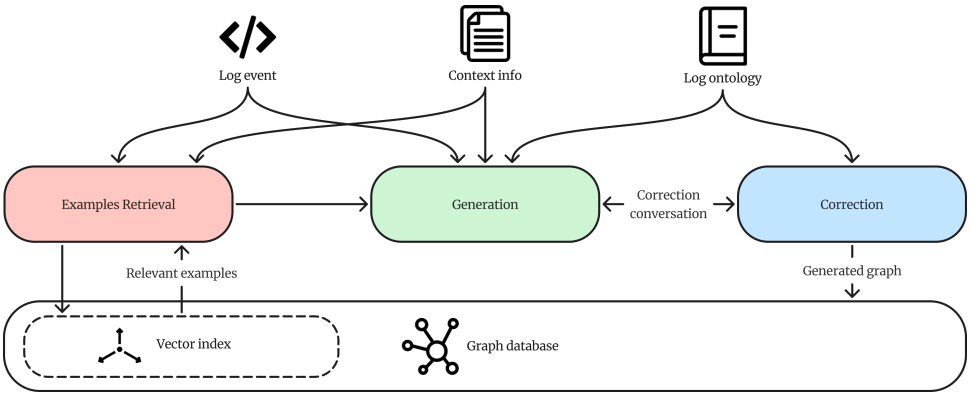


Figure 1: High-level procedure for generating a log event KG.

entries.

While the integration of LLMs with retrieval mechanisms improves output relevance, challenges remain with respect to consistency, interpretability, and quality control. These concerns are particularly salient in applications involving automated extraction of threat indicators or behavioral patterns from log data, where the reliability of each extracted item has downstream implications for analysis and response. In such cases, the transparency of the processing pipeline—understood here as the ability to inspect and guide the model's decision process through natural language prompts and accessible intermediate outputs—becomes an important consideration. Transparent configurations facilitate manual oversight and error correction, and may also support reproducibility in operational settings.

AI Agents can be defined as autonomous software entities, typically based on LLMs, engineered for goal-directed task execution within bounded digital environments [26, 27, 28]. These agents can perceive structured or unstructured inputs, reason over contextual information, initiate actions toward achieving specific objectives, often acting as surrogates for human users or subsystems [27].

Existing research has demonstrated that combining semantic constraints leads to substantially improved accuracy and reliability of extracted information [29]. Hence, the explicit integration of ontology-driven semantic knowledge and SHACL-based constraints within agentic LLM pipelines constitutes a promising strategy for developing robust and semantically precise cybersecurity intelligence extraction systems.

3. Methodology

OntoLogX operates as an AI Agent with minimal user interaction, constructing a KG representing a log event and, optionally, its contextual information. It integrates an LLM with an ontology for log events to leverage symbolic representations of the cybersecurity domain, thereby enabling automatic feedback behaviors that ensure the generation of a valid KG. OntoLogX is designed for online operation, parsing and processing log events incrementally and sequentially, one by one, in a setting that reflects realistic cybersecurity use cases requiring near real-time event analysis.

Figure 1 depicts the overall workflow. Given a log event and any optional context information (e.g., the device or application the log originates from), several relevant log event KGs are retrieved from the graph database. Then, the input log event, context information, and the log ontology are used to instruct an LLM to generate a new KG, using the retrieved KGs as few-shot examples. The generated KG is validated: if it does not conform to the expected format or is not ontology-compliant, the LLM is prompted again in the same conversation to generate specific corrections. If a valid KG is eventually produced, it is saved in the graph database where it may be retrieved during future log processing operations. It is worth noting that KGs are stored independently from each other, as the semantic connection of different KGs is not the focus of this work.

Table 1
Comparison of the suitability of selected cybersecurity ontologies for log-based semantic extraction.

Aspect	OntoLogX ontology	SEPSES ontology	UCO	STIX
Purpose	LLM-based log parsing	Log integration	Unified CTI modeling	Threat intel exchange
Input Type	Free-text logs	Parsed logs	Structured observables	Structured indicators
Complexity	Minimal, flat schema	Medium-high	High, modular	High, domain-specific
SHACL Use	Core for validation	RDF quality checks	Optional	No
Ontology Role	Preprocessing layer	Integration layer	Reasoning layer	Exchange format
Mapping Potential	Core CTI entities	CVE, CWE, CAPEC, CPE	STIX, CVE, CAPEC	Often paired with UCO

3.1. Log Ontology

To formalize the knowledge to be inferred from log events and enable semantic interoperability of the generated KGs, we developed a custom log ontology, illustrated in Figure 2. Starting from the recommended log event details by the Australian Signals Directorate's Australian Cyber Security Centre[1], we adopted a bottom-up, data-driven approach to design a lightweight, flat ontology grounded in the direct analysis of representative log datasets, including but not limited to honeynet event streams. While it does not aim for full expressivity, it provides a stable intermediate representation suitable for post-processing alignment with richer ontologies such as STIX or SEPSES. For example, a log entry such as:

```
2022-01-21 03:49:44 jhall/192.168.230.165:46011 VERIFY OK: CN=OpenVPN CA
```

implicitly contains information such as a timestamp, user identity, network address, and certificate details—all encoded without structural consistency or explicit separation. Ontologies such as UCO or STIX typically assume the availability of structured metadata or pre-parsed fields, making them difficult to apply directly to such unstructured inputs.

Table 1 summarizes the key differences between our ontology and widely adopted cybersecurity schemas in terms of design goals, input assumptions, and applicability to log-based semantic extraction.

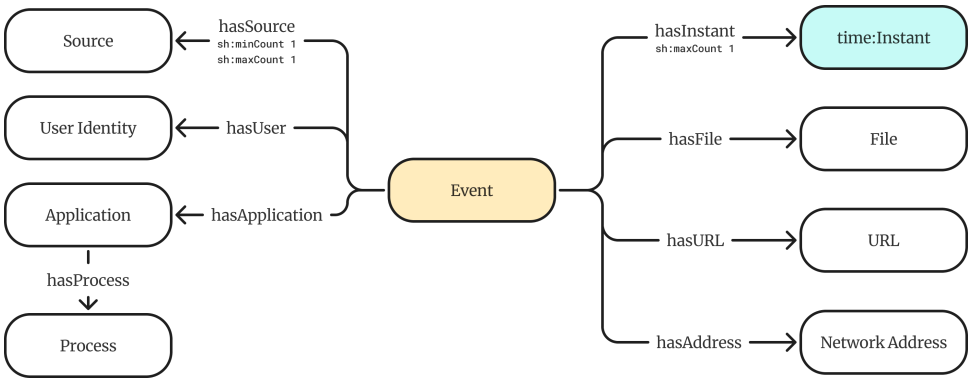


Figure 2: Representation of the classes and object properties of the OntoLogX ontology, with SHACL constraints that apply to them. Data properties (along with their SHACL constraints) are not reported for conciseness.

The primary design objective was to create a self-contained, semantically expressive model that captures the core entities and relationships found in raw log entries without being too specific for our specific use case, thus making the ontology scalable to logs from different datasets and sources. The central concept is the class `Event`, which serves as the semantic anchor for each log entry. Associated classes are also modeled: `User Identity`, the credential or identity of a user involved in the event; `Application` and `Process`, used to represent execution context; *File*, the file resource being accessed

[1] https://www.cyber.gov.au/sites/default/files/2024-08/best-practices-for-event-logging-and-threat-detection.pdf (on August 6, 2025).

or modified; `Network Address`, such as IP addresses or ports mentioned in the log; `Source`, the logical or physical origin of the log entry; `URL`: representing any remote reference in the log content. Temporal dimensions are modeled using the standard W3C Time ontology[2]. In addition to object properties (e.g., `hasUser`, `hasProcess`), the ontology includes data properties such as `eventMessage`, capturing the log's raw content, and `logLevel`, denoting its severity.

To enforce schema compliance and semantic consistency, a companion SHACL specification defines constraints on key classes and properties. These constraints are critical when constructing graphs via a LLM-based extraction pipeline, where enforcing schema compliance helps mitigate issues such as missing fields or incorrect relationships. By enforcing constraints on cardinality, type consistency, and presence of essential properties, we ensure that generated KGs not only follow the OntoLogX vocabulary but are also semantically valid and queryable.

3.2. Examples Retrieval

To guide the generation process, the raw log event, along with optional context (e.g., the device or application that generated the log), is first queried against a vector store. This store indexes both previously generated KGs and manually crafted examples aligned with the log ontology. The goal of this retrieval step is to identify semantically similar examples that can serve as few-shot prompts, thereby enhancing the quality and structure of the output.

The search leverages Maximal Marginal Relevance (MMR) [30], a ranking strategy designed to balance *relevance* and *diversity*. Rather than simply returning the top-k most similar entries, MMR penalizes redundancy by selecting items that are both relevant to the query and dissimilar from one another. This reduces the likelihood of retrieving near-duplicate examples and encourages exposure to a broader range of graph structures and content patterns. Formally, given a query q, a candidate set of documents D, and a set of already selected items $S \subset D$, MMR iteratively selects the next item $d^* \in D \setminus S$ that maximizes the following objective:

$$\text{MMR}(d) = \lambda \cdot \text{Sim}(d, q) - (1 - \lambda) \cdot \max_{s \in S} \text{Sim}(d, s)$$

where $\text{Sim}(d, q)$ measures the similarity between document d and query q, $\text{Sim}(d, s)$ measures the similarity between d and already selected document s, and $\lambda \in [0, 1]$ controls the trade-off between relevance and diversity. This retrieval strategy helps the LLM better internalize the expected structure and semantics of the output graph. Additionally, it enables the model to reuse domain knowledge and contextual cues from previously processed log events, improving both consistency and completeness in KG construction.

3.3. Generation

The generation component of OntoLogX is responsible for constructing a KG from raw log events in alignment with the log ontology described in Section 3.1. This step involves invoking an LLM, which takes as input the raw log entry, any optional contextual information, and a set of few-shot examples retrieved through semantic search. The use of a LLM enables the generation of high-quality KGs even in low-resource scenarios, without requiring task-specific supervised training. Furthermore, the LLM can leverage its internal knowledge to infer implicit information, resolve ambiguities, and normalize terminology—capabilities that are often difficult to replicate with rule-based or statistical parsers.

We hypothesize that LLMs inherently encode significant amounts of latent cybersecurity knowledge, due to their pretraining on diverse and large-scale datasets. This domain familiarity improves their ability to recognize entities, activities, and relationships common in log data, even when such elements are expressed in non-standard, abbreviated, or noisy formats. Consequently, an LLM-driven approach offers significantly greater adaptability to heterogeneous or previously unseen log types compared to conventional log parsers, which typically depend on learned templates to extract parameters from structured inputs.

[2] https://www.w3.org/TR/owl-time/ (on August 6, 2025).

Table 2
Prompt for log event KG generation, used in conjunction with structured output.

```
# Overview
You are a top-tier cybersecurity expert specialized in extracting structured information from
unstructured data to construct a knowledge graph according to a predefined ontology. You will
be provided with a log event, optionally accompanied by contextual information.
Your goal is to maximize information extraction from the event while maintaining absolute
accuracy. Leverage both the contextual information and your knowledge of computer systems and
cybersecurity to infer additional insights where possible. The objective is to achieve
completeness in the knowledge graph while remaining strictly ontology-compliant.
# Rules
You MUST adhere to the following constraints at all times:
- The graph must contain exactly one "Event" node, with a property "eventMessage" that holds
the original event text.
- Do not introduce any new node types, relationship types, or property types. Only use the
available types.
- Respect the appropriate casing for all types.
- Use the appropriate node prefix for properties, e.g. "userUID" instead of "uid".
- If not specified in the log event, try to infer the severity of the event based on the
message. If you cannot determine the severity, default to "INFO".
- The graph must be connected: there should be no isolated nodes.
# Strict Compliance
Adhere to these rules strictly. Any deviation will result in termination.
```

Table 3
Format of structured output that the LLM must conform to.

Entity	Description	Elements	
		Name	Description
Graph	An event knowledge graph composed of nodes and relationships.	Nodes	List of nodes in the graph.
		Relationships	List of relationships in the graph.
Node	A node in the event graph. Each node type has a specific set of allowed properties.	ID	Unique identifier for the node.
		Type	Ontology class name.
		Properties	List of properties of the node.
Property	A property of a node in the event knowledge graph.	Type	Ontology data property name.
		Value	Extracted value of the property.
Relationship	A relationship between two nodes in the event graph. Each relationship type has a predefined source and target node type.	Source ID	Unique identifier of source node.
		Target ID	Unique identifier of target node.
		Type	Ontology object property name.

The prompt used to instruct the LLM (shown in Table 2) is intentionally designed to be model-agnostic, maximizing compatibility with a broad range of language models. It comprises: (i) a clear role and task definition, (ii) detailed instructions grounded in the OntoLogX ontology, and (iii) a set of constraints and clarifications based on common failure patterns observed in early experiments, such as malformed URIs, incorrect predicate usage, and mismatched types.

To reinforce structural consistency and reduce formatting variability, the LLM is equipped with a structured output tool, whose target schema is shown in Table 3. This tool defines the required fields and their expected types, effectively serving as a strong constraint during generation. It not only improves the reliability of downstream parsing and validation but also helps encode explicit knowledge about the ontology, ensuring that the produced KGs conform to the intended semantic structure.

Finally, the LLM is provided with few-shot examples—using the KGs retrieved from the database in the previous step—further reinforcing the desired output structure and ontology-aligned vocabulary, guiding the model toward generating consistent and valid outputs.

Table 4
Lines added to the prompt in Table 2 to compose the baseline prompt for KG generation.

```
# Output Format
The output graph must be in the following JSON format: {{json output format}}
Each node type has a specific set of allowed properties. The allowed properties for each node
type are: {{properties schema}}
Each relationship type has a predefined source and target node type. The allowed
relationships, formatted as (source type, relationship type, target type), are: {{triples}}
# Strict Compliance
Adhere to these rules strictly. Any deviation will result in termination.
```

3.4. Correction

Given the inherent variability of LLMs, the output of the generation step may fail to conform to the expected ontology structure, or it may be incomplete or semantically invalid. To address this, OntoLogX incorporates a dedicated *correction* phase that identifies such issues and engages in a feedback-driven interaction with the LLM to request targeted revisions.

This correction process is coordinated through the LLM's structured output tool, which performs automatic validation using two key criteria. First, it assesses the syntactic validity of the output, verifying that the graph is composed of well-formed nodes and relationships. Second, it checks for ontology compliance, ensuring that the graph adheres to the structural and semantic constraints specified by the log ontology—this includes correct class and property usage, proper data typing, and satisfaction of required schema constraints.

When one or more issues are detected, a tailored correction prompt is constructed and passed back to the LLM. This iterative interaction continues for multiple rounds until a fully valid and ontology-compliant KG is produced. If a graph without errors KG is not achieved within these rounds, an empty graph is instead produced, thus penalizing evaluation metrics.

Once a syntactically valid and ontology-compliant KG is produced—either on the first attempt or after one or more correction cycles—it is stored in the graph database for long-term persistence and downstream consumption. Each KG is also annotated with its input log event and contextual information, to enable traceability and further querying. This source information is also used to compute a vector embedding of each KG, allowing for efficient semantic search and retrieval of related events in future queries.

4. Experimental Results

This section compares the experimental results obtained using OntoLogX and a baseline configuration designed for comparative evaluation. The baseline consists of a single generation step based on a traditional prompt-only approach, thus excluding the examples retrieval and correction features of OntoLogX. The baseline prompt, which is obtained by adding the lines shown in Table 4 to the prompt in Section 3.2, instructs the LLM in producing a JSON-formatted output conforming to the schema in Table 3 with plain prompting rather than with structured output.

Both methodologies were evaluated across a total of five different LLMs, which are reported in Table 5. Notably, the proprietary models chosen are significantly more expensive than those with more permissive licenses, as are medium-sized models compared to smaller ones.

All models were accessed via AWS Bedrock[5], except for `Qwen 2.5 Coder`, which was run locally using vLLM [31] on a system equipped with four NVIDIA T4 GPUs. A temperature of 0.7 was used over all runs to promote creative reasoning, which we hypothesize is beneficial for inferring implicit

[3] https://mistral.ai/static/licenses/MNPL-0.1.md (on August 6, 2025).
[4] https://www.llama.com/license/ (on August 6, 2025).
[5] https://aws.amazon.com/it/bedrock/ (on August 6, 2025).

Table 5
Comparison of LLMs used in experiments. The number of parameters indicates the size of the model.

Model	# of parameters	License
Claude 3.5 Haiku	Unknown, small size	Proprietary
Claude 3.5 Sonnet	Unknown, medium size	Proprietary
Llama 3.3	70 billions	Custom[3], permissive
Mistral Large 24.02	123 billions	Custom[4], permissive
Qwen 2.5 Coder	14 billions	Apache 2.0

information from raw logs. To mitigate the variability introduced by this setting, each experiment was repeated ten times.

4.1. Implementation

OntoLogX is implemented in Python, using the LangChain framework[6] to manage the LLM and parsing pipeline. The resulting ontology-compliant KGs, along with the underlying log ontology, are stored in a Neo4j graph database[7], extended with a vector index to support efficient storage and semantic retrieval based on node property embeddings. To facilitate reproducibility and the structured sharing of results, the database is organized with a graph schema that follows the MLSchema ontology [32].

The structured output format is specified through the Pydantic[8] library, and the LLMs generate the output through function calling.

In the version evaluated in this study, only manually annotated examples are used for few-shot prompting; graphs previously generated by the system are excluded from the retrieval step. The embedding model used is `gte-multilingual-base`. Moreover, the correction step is limited to a maximum of three refinement prompts per log. If a valid graph is not obtained within this limit, the output is considered to be an empty graph.

4.1.1. Dataset

The test dataset consists of log events sampled from the AIT-LDS dataset [33]. A total of 70 log entries were randomly selected to ensure broad diversity, both in log types (e.g., Apache, VPN, audit) and semantic content. To promote this diversity, the first 100 log events were extracted from each file in the *RussellMitchell* testbed. From this pool, 70 events were iteratively selected using an embedding-based dissimilarity criterion.

Specifically, the embedding of each candidate log entry was computed using the `nomic-embed-text-v1.5` model. For each newly drawn candidate, the cosine distance was calculated with respect to the embeddings of all previously selected entries. If the minimum distance was below 0.7—indicating excessive semantic similarity—the candidate was retained; otherwise, it was discarded and another was drawn. Each selected log entry was manually annotated with a corresponding ground-truth KG, and the dataset was partitioned into three subsets: the first 10 events were reserved for few-shot prompting, the next 10 for validation, and the remaining 50 for testing.

4.1.2. Metrics

To evaluate the quality of KGs generated by LLMs, we combine the precision, recall and F1 score metrics from the ontologies field with G-Eval LLMs. In addition to these metrics, we evaluate the percentage of generated graphs that violate SHACL rules.

[6] https://www.langchain.com/ (on August 6, 2025).
[7] https://neo4j.com/product/neo4j-graph-database/ (on August 6, 2025).
[8] https://pydantic.dev/ (on August 6, 2025).

Table 6
Model performance comparison between Baseline and OntoLogX. Given a row and a metric, bold values indicate the better mean performance.

Model	Precision				Recall				F1 Score				G-Eval Score			
	Baseline		OntoLogX		Baseline		OntoLogX		Baseline		OntoLogX		Baseline		OntoLogX	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Claude 3.5 Haiku	0.778	0.006	**0.857**	0.007	0.453	0.004	**0.820**	0.007	0.573	0.004	**0.838**	0.006	0.714	0.008	**0.750**	0.007
Claude 3.5 Sonnet	0.753	0.006	**0.859**	0.005	0.514	0.003	**0.858**	0.007	0.611	0.002	**0.859**	0.005	0.728	0.012	**0.767**	0.007
Llama 3.3	0.810	0.004	**0.904**	0.007	0.461	0.007	**0.814**	0.007	0.588	0.006	**0.856**	0.005	0.660	0.016	**0.720**	0.010
Mistral Large	0.803	0.011	**0.811**	0.019	**0.406**	0.012	0.286	0.015	**0.539**	0.011	0.423	0.015	**0.637**	0.025	0.418	0.030
Qwen 2.5 Coder	0.420	0.450	**0.870**	0.015	0.005	0.006	**0.759**	0.015	0.009	0.011	**0.811**	0.011	0.008	0.009	**0.723**	0.014

1. *Precision*: ratio of generated triples that are correct, i.e., have a matching triple in the ground truth, over the total number of generated triples. High precision values indicate that the model tends to produce accurate triples with relatively few incorrect or spurious facts.

2. *Recall*: ratio of correct triples that were generated, i.e., those that match a ground-truth triple, over the total number of ground-truth triples. High recall values indicate that the model successfully captures most of the relevant information present in the ground truth.

3. *F1 Score*: harmonic mean between precision and recall.

4. *G-Eval Score*: a LLM-as-a-judge framework that uses chain-of-thought reasoning and a form-filling paradigm to evaluate natural language generation outputs [34]. It operates by prompting the LLM with a set of evaluation criteria—either user-defined or automatically derived—to produce scores reflecting various aspects of output quality. In our setting, it uses the `Llama 3.3` model to assess the extent to which the generated KG retains the original log event's meaning. The LLM is prompted to produce natural language summaries of both the raw log and the KG, and then to score their semantic overlap on a scale from 0 to 1. Additional information in the KG lowers the score only if it is judged irrelevant. Higher G-Eval scores indicate more faithful and relevant extractions.

4.1.3. Results

Table 6 reports the mean and standard deviation over all runs of each model under the baseline and OntoLogX configurations. The mean values for the F1 and G-Eval scores, along with the percentage of SHACL violations, are also reported in Figure 3. Overall, the execution time—encompassing the examples retrieval, generation, and corrections but excluding database storage time—goes from 1.4 seconds with `Llama 3.3` to 9.8 seconds with `Mistral Large`.

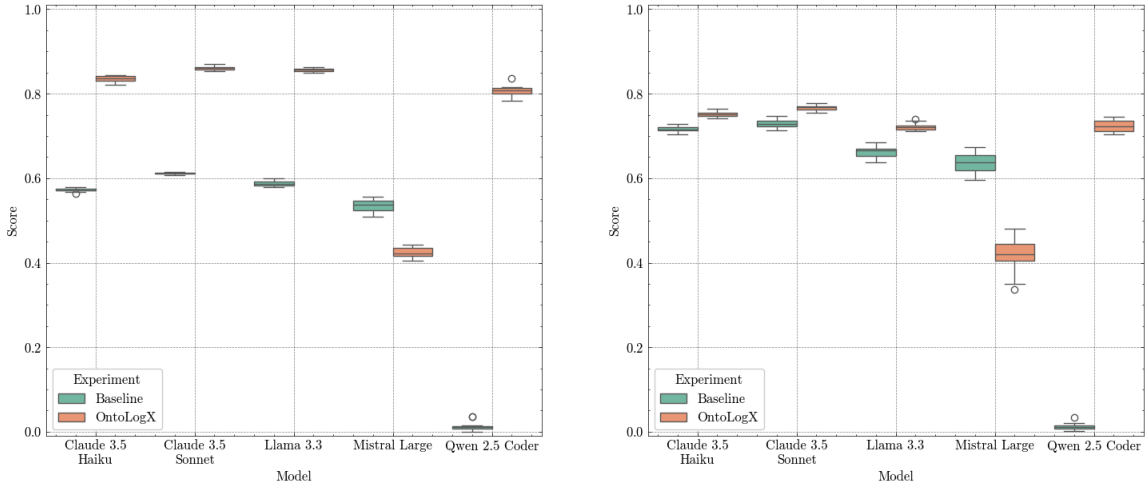
Across nearly all models and metrics, the OntoLogX-enhanced setup consistently outperforms the baseline configuration. In particular:

- All models except `Mistral Large` achieve higher precision under the OntoLogX setup. The most significant improvement is observed with `Qwen 2.5 Coder`, for which the baseline configuration yields highly variable results.

- OntoLogX substantially improves recall for all models except `Mistral Large`, which shows a decrease. `Claude 3.5 Sonnet` and `Qwen 2.5 Coder` exhibit the largest gains.

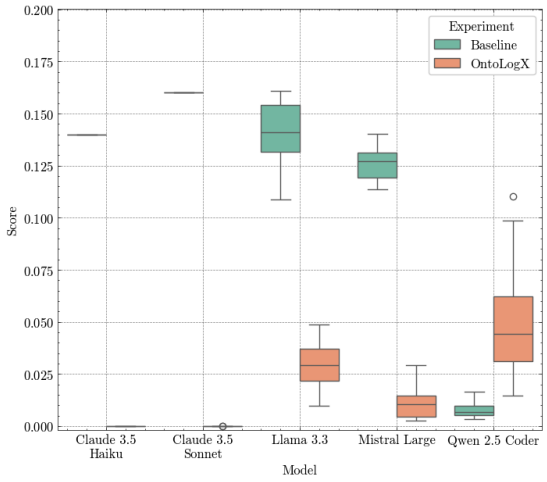
- OntoLogX improves G-Eval scores across all models. Again, `Qwen 2.5 Coder` benefits the most, suggesting enhanced semantic alignment and structural quality in its outputs.

- OntoLogX also significantly increases SHACL compliance, mostly due to the use of structured outputs that conform to the SHACL rules. Interestingly, the results are not guaranteed to be fully



(a) F1 scores

(b) G-Eval scores



(c) Percentage of graphs that violate SHACL constraints

Figure 3: Comparison of selected metrics between baseline and OntoLogX.

SHACL-compliant: this can be attributed to the limited expressiveness offered by the adopted structured output approach compared to the SHACL constraints language.

Among all evaluated models, `Claude 3.5 Sonnet` stands out as the top performer in the OntoLogX configuration, achieving the highest F1 and G-Eval scores, alongside strong results in the other metrics. Nevertheless, `Llama 3.3` emerges as a robust open-weights alternative with competitive performance. Conversely, `Mistral Large` is the only model that performs better under the baseline configuration for recall, F1, and G-Eval—though the absolute values remain modest. This outcome suggests a lower compatibility with the OntoLogX pipeline or a higher sensitivity to structured prompting and tool-assisted outputs. It also underscores the limitations of adopting a uniform prompting strategy across heterogeneous models and the importance of selecting models that support advanced tooling features.

In summary, the OntoLogX methodology delivers substantial gains in accuracy, completeness, and semantic faithfulness across a range of LLMs, demonstrating its effectiveness in guiding model outputs toward high-quality, ontology-compliant KG construction. Small and inexpensive models are also viable, especially code-specific ones.

5. Conclusions

This paper introduces OntoLogX, a novel methodology for extracting semantically enriched CTI from cybersecurity logs using a combination of ontology-driven structured output and LLMs. The approach addresses the challenges of interpreting unstructured and ambiguous log entries by integrating ontological knowledge to guide the language model's reasoning and validate extracted data. The extracted information is then organized into an ontology-enriched graph database, facilitating future semantic analysis and querying.

The experimental evaluation, conducted using logs of various types extracted from a publicly available dataset, demonstrates the effectiveness of OntoLogX, with an intentional focus on extraction quality over processing speed. The results indicate that the OntoLogX-enhanced setup consistently outperforms a prompt-only baseline configuration across precision, recall, and G-Eval metrics for nearly all models tested. Among the tested models, `Claude 3.5 Sonnet` achieved the best overall performance. In addition, code-focused models such as `Qwen 2.5 Coder` showed promising results, highlighting the potential of task-specific architectures.

Looking forward, several directions will guide future work. A key area is the development of interactive interfaces that allow analysts to engage with the ontology-enriched knowledge graph directly. These interfaces will support querying, validating, and refining extracted intelligence, facilitating human-in-the-loop workflows that combine automation with expert control. In parallel, we aim to implement adaptive feedback mechanisms that allow the system to learn from analyst input and respond to evolving adversarial behaviors. This will improve the resilience and adaptability of the extraction pipeline over time.

Another development involves the extension and alignment of the OntoLogX ontology. This includes expanding and integrating logs from other domains such as endpoint detection, cloud infrastructure, and application layers, and formalizing mappings to widely adopted CTI standards like SEPSES, UCO, and MISP. Such efforts are critical to ensure interoperability with existing threat intelligence platforms and support seamless integration into operational pipelines.

Finally, we plan to investigate further the use of task-specialized LLMs, particularly those optimized for reasoning or code generation, such as `Qwen 3` and `DeepSeek Coder`. When properly guided, these models may offer superior performance in graph generation and correction.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU, specifically by the project NEACD: Neurosymbolic Enhanced Active Cyber Defence (CUP J33C22002810001). This project was also partially funded by the Italian Ministry of University as part of the PRIN: PROGETTI DI RICERCA DI RILEVANTE INTERESSE NAZIONALE – Bando 2022, Prot. 2022EP2L7H

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4o for the following purposed: Grammar and spelling check, Paraphrase and reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] E. M. Hutchins, M. J. Cloppert, R. M. Amin, et al., Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, Leading Issues in Information Warfare & Security Research 1 (2011) 80.

[2] P. McDaniel, B. Rivera, A. Swami, Towards proactive cybersecurity: Methodologies and tools, IEEE Security & Privacy 14 (2016) 12–14.
[3] W. Tounsi, H. Rais, A survey on technical threat intelligence in the age of sophisticated cyber attacks, Computers & Security 72 (2018) 212–233.
[4] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, J. Schönfelder, A survey on honeypot software and data analysis, arXiv preprint arXiv:1608.06249 (2016).
[5] G. Izacard, E. Grave, Leveraging passage retrieval with generative models for open domain question answering, arXiv preprint arXiv:2007.01282 (2020).
[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.
[7] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al., Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, arXiv preprint arXiv:2206.04615 (2022).
[8] C. Wagner, A. Dulaunoy, G. Wagener, A. Iklody, Cyber Threat Intelligence, Springer, 2019.
[9] L. Spitzner, Honeypots: Tracking Hackers, Addison-Wesley Professional, 2003.
[10] D. Fraunholz, S. D. Anton, C. Lipps, F. Pohl, M. Zimmermann, H. D. Schotten, Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot, IEEE Access 6 (2018) 37163–37174.
[11] C. Zhao, G. Agrawal, T. Kumarage, Z. Tan, Y. Deng, Y.-C. Chen, H. Liu, Ontology-aware rag for improved question-answering in cybersecurity education, arXiv preprint arXiv:2412.14191 (2024).
[12] Z. Syed, A. Padia, T. Finin, L. Mathews, A. Joshi, Uco: A unified cybersecurity ontology, in: AAAI Workshop: Artificial Intelligence for Cyber Security, 2016, pp. 14–21.
[13] S. Barnum, Standardizing cyber threat intelligence information with the structured threat information expression (stix), The MITRE Corporation (2012).
[14] R. Studer, V. R. Benjamins, D. Fensel, Knowledge engineering: Principles and methods, in: Data & knowledge engineering, volume 25, Elsevier, 1998, pp. 161–197.
[15] A. Oltramari, L. F. Cranor, R. J. Walls, P. McDaniel, Building an ontology of cyber security, in: Proceedings of the Ninth Conference on Semantic Technology for Intelligence, Defense, and Security (STIDS), CEUR-WS.org, 2014, pp. 54–61.
[16] C. Wagner, A. Dulaunoy, G. Wagener, A. Iklody, Misp: The malware information sharing platform, in: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, ACM, 2016, pp. 49–56.
[17] E. Kiesling, A. Ekelhart, K. Kurniawan, F. J. Ekaputra, The SEPSES knowledge graph: An integrated resource for cybersecurity, in: The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II, volume 11779 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 198–214. URL: https://doi.org/10.1007/978-3-030-30796-7_13. doi:10.1007/978-3-030-30796-7_13.
[18] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (2021) 1–37.
[19] H. Knublauch, D. Kontokostas, Shapes constraint language (shacl), W3C Recommendation 20 (2017).
[20] P. Pareti, G. Konstantinidis, T. J. Norman, et al., Knowledge graph quality management with shacl, Journal of Web Semantics 74 (2022) 100714.
[21] K. Rabbani, M. Lissandrini, K. Hose, SHACTOR: improving the quality of large-scale knowledge graphs with validating shapes, in: S. Das, I. Pandis, K. S. Candan, S. Amer-Yahia (Eds.), Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023, ACM, 2023, pp. 151–154. URL: https://doi.org/10.1145/3555041.3589723. doi:10.1145/3555041.3589723.
[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
[23] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers

for language understanding, in: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.
[24] T. B. Brown, B. Mann, N. Ryder, et al., Language models are few-shot learners, Advances in Neural Information Processing Systems 33 (2020) 1877–1901.
[25] K. Guu, K. Lee, Z. Tung, P. Pasupat, M. Chang, Retrieval augmented language model pre-training, in: International conference on machine learning, PMLR, 2020, pp. 3929–3938.
[26] D. B. Acharya, K. Kuppan, B. Divya, Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey, IEEE Access (2025).
[27] R. Sapkota, K. I. Roumeliotis, M. Karkee, AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges, 2025. doi:10.48550/arXiv.2505.10468. arXiv:2505.10468.
[28] F. Sado, C. K. Loo, W. S. Liew, M. Kerzel, S. Wermter, Explainable goal-driven agents and robots-a comprehensive review, ACM Computing Surveys 55 (2023) 1–41.
[29] Y. Sun, X. Ren, et al., Ontology-guided prompt engineering for large language models, arXiv preprint arXiv:2307.07018 (2023).
[30] J. Carbonell, J. Goldstein, The use of mmr, diversity-based reranking for reordering documents and producing summaries, in: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998, pp. 335–336.
[31] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, I. Stoica, Efficient memory management for large language model serving with pagedattention, 2023. URL: https://arxiv.org/abs/2309.06180. arXiv:2309.06180.
[32] G. C. Publio, D. Esteves, A. Lawrynowicz, P. Panov, L. N. Soldatova, T. Soru, J. Vanschoren, H. Zafar, Ml-schema: Exposing the semantics of machine learning with schemas and ontologies, CoRR abs/1807.05351 (2018). URL: http://arxiv.org/abs/1807.05351. arXiv:1807.05351.
[33] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, A. Rauber, AIT Log Data Set V2.0, 2022. doi:10.5281/zenodo.5789064.
[34] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, C. Zhu, G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. URL: https://arxiv.org/abs/2303.16634. arXiv:2303.16634.