Scientific Research Publishing

Journal of Information Security, 2025, 16(4), 544-567
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

STIXAgent—A Multi-Agent Framework for Standardized Management of Cyber Threat Intelligence (CTI) Reports

Vijay K. Madisetti

Georgia Institute of Technology, Atlanta, GA, USA
Email: vkm@gatech.edu

How to cite this paper: Madisetti, V.K. (2025) STIXAgent—A Multi-Agent Framework for Standardized Management of Cyber Threat Intelligence (CTI) Reports. *Journal of Information Security*, **16**, 544-567.
https://doi.org/10.4236/jis.2025.164028

Received: September 15, 2025
Accepted: October 25, 2025
Published: October 28, 2025

Abstract

As the volume of Cyber Threat Intelligence (CTI) reports from multiple sources continues to rise, automated tools are essential for consistent and accurate management of heterogeneous data. Existing methods, such as STIXnet and aCTIon, require substantial human intervention, limiting their scalability. This paper introduces STIXAgent, a multi-agent framework that automates the conversion of unstructured CTI reports into STIX-compliant JSON structures. Our approach leverages LangGraph to orchestrate modular task execution, incorporating Large Language Models (LLMs) for information extraction, structured validation, and error handling. This paper utilises 5 LLMs (GPT4o, Gemini, Llama3 (8B), DeepseekR1-distilled Qwen32B, and LLama 70B) for multiagentic LLM evaluation, introducing in the process a novel Bayesian statistical evaluation framework to assess model performance, offering probabilistic insights into content accuracy and structural consistency. We benchmark STIXAgent across 10, 40, 100, and 400 reports from Microsoft and Talos, respectively. The results demonstrate that STIXAgent not only enhances automation but also improves reliability through structured evaluation, setting a new standard for scalable CTI processing.

Keywords

Cyber Threat Intelligence, Large Language Models (LLMs), Natural Language Processing (NLP), Structured Threat Information Expression (STIX)

1. Introduction

Cyber Threat Intelligence (CTI) serves a pivotal role in furnishing security professionals with vital information necessary for protection against cyber threats and

the formulation of response strategies. The Structured Threat Information Expression (STIX) standard, represented graphically and stored in JSON format, effectively captures critical threat information and their interrelationships. This structured approach facilitates the use of CTI by automated tools, enhancing the efficiency of search and analytical processes. Despite the prevalence of structured data sources in CTI, such as IP blocklists and malware signatures conveyed through STIX graphs, among other formats, a substantial portion of CTI exists in unstructured forms, including textual reports and articles. This unstructured information holds significant value for security professionals, as it provides comprehensive insights into threats, comprising details about adversaries, targets, methodologies, employed tools, malware, and patterns of attack. Consequently, analysts invest a considerable amount of effort in synthesizing these disparate information sources into actionable intelligence.

This paper introduces STIXAgent, a multi-agent framework that automates the transformation of unstructured CTI reports into standardized STIX JSON structures. By leveraging LangGraph, STIXAgent orchestrates modular LLM-driven agents to perform entity extraction, UUID generation, schema validation, and missing information retrieval. A key contribution of this work is a Bayesian statistical evaluation framework, which provides probabilistic insights into the accuracy and consistency of LLM-generated STIX outputs. Through comparative benchmarking across multiple LLMs (GPT-4o, Gemini, Llama, Qwen, etc.), we demonstrate STIXAgent's effectiveness in enhancing automation, improving structural compliance, and reducing reliance on manual intervention [1].

2. Background

2.1. Natural Language Processing

Efforts to turn natural language into some form of structured output are not new. The field of Natural Language Processing (NLP) has explored this area, from speech recognition (converting spoken language into text) to Part-Of-Speech (POS) tagging (determining the grammatical role of words based on their context) and Named Entity Recognition (NER), which identifies specific terms in a text as entities of a particular type [2]. Among these, NER plays a significant role in Information Extraction tasks that involve processing extensive textual data [3], such as in medical applications [4], scientific research [5], and cybersecurity intelligence [1].

Information Extraction also encompasses Relation Extraction, which involves identifying and categorizing the semantic relationships between two or more tokens in a text [6]. This task often complements Entity Extraction, enabling the integration of extracted information into a cohesive structure. By combining these techniques, textual data can be transformed into a knowledge graph, providing an interconnected representation of the information [7].

In more recent years, the concept of Large Language Model (LLM) agents has gained prominence with the development of models like GPT-3 in 2020, which

showcased advanced language understanding and generation capabilities. This progress led to the creation of autonomous AI agents such as AutoGPT in 2023, where the model autonomously generated and executed plans based on user-defined goals [8]. Another significant development was the "Generative Agents" framework by Park *et al.* in April 2023, which simulated human-like behaviors in interactive environments [9]. These innovations aimed to enhance AI's ability to perform complex tasks with minimal human intervention.

Today, LLM agents function as intelligent entities that perceive their surroundings, make decisions, use their own tools, and execute actions. By simulating human-like collaboration and communication, LLM agents embody capabilities such as self-action, mutual interaction, and evolution. Their modular framework, which encompasses profile creation, environmental perception, self-learning, and inter-agent communication, offers a structured way to enhance adaptability and scalability in autonomous systems [9].

The promise of LLM agents lies in their ability to synthesize collective intelligence while maintaining the unique expertise of individual agents. This capability is a significant step towards achieving general artificial intelligence, as these agents can reason, learn, and evolve dynamically in real-world scenarios. Applications like multi-agent simulations and problem-solving have demonstrated their potential to optimize decision-making, facilitate teamwork, and improve the robustness of autonomous systems [9]. Our paper leverages the promise that LLM agents provide and extends it to more traditional areas of information extraction and conversion.

2.2. Cyber Threat Intelligence

Cyber Threat Intelligence (CTI) is frequently disseminated in unstructured formats, such as those found on the dark web, industry reports, online listicles, and government notices, among others. Due to the expansive volume of CTI, analysts invest considerable effort in aggregating these diverse information sources to support their research and analysis concerning these threats. This includes examining the intentions of threat actors, the techniques they employ, and the tools and malware they utilize. Such analysis contributes to profiling malicious actors' activities and facilitates the formulation of more effective cyber defense strategies [10]-[12].

CTI encompasses various types of intelligence, tailored to be more or less technical depending on the intended audience. Each piece of intelligence adheres to a defined lifecycle, spanning from planning and direction to dissemination and integration [11]. To ensure the efficient sharing and distribution of collected intelligence, a standardized protocol is essential to minimize misunderstandings among consuming teams. The STIX (Structured Threat Information eXpression) standard was developed to address this need [13]. In STIX, each report or bundle is essentially a knowledge graph, with different entity and relation types. These entities include Threat Actor, Malware, Vulnerability, and Attack Pattern, with different uses and targets.

3. Related Work

3.1. STIXnet

In developing STIXnet, the authors established a framework for the automated extraction of entities and relationships that adhere to the STIX format from Cyber Threat Intelligence (CTI) reports. The techniques of Named Entity Recognition (NER) and Part-Of-Speech (POS) tagging were employed to discern key entities, such as malware, threat actors, and tools, within the textual data. This framework utilized a collection of modular entity extraction components functioning cooperatively to identify both pre-existing and novel entities. The components comprise an Indicators Of Compromise (IOCs) Finder, which employs regular expressions for IOC extraction; a Knowledge Base extraction module, which utilizes efficient string-matching algorithms to recognize entities contained within a precompiled database; an Entity Extraction module, deploying dependency parsing and pattern recognition to identify newly discovered entities; and a TTP Extraction module, which implements machine learning techniques to extract tactics and techniques as outlined in the MITRE ATT&CK framework [13] [14].

In addition to entity extraction, STIXnet introduces two complementary approaches for relation extraction, which are critical for understanding the relationships between identified entities. The rule-based approach leverages the dependency graphs generated during the entity extraction process, using graph theory techniques to determine the shortest paths between entities. By analyzing the verbs within these paths and comparing them with known STIX relationship types, the system can classify relationships with a degree of confidence. However, this approach can be limited in its ability to capture relationships in more complex sentences. To address this, the system also employs a deep learning-based approach using Sentence-BERT (SBERT), a variation of the BERT model optimized for sentence embeddings. This method computes the similarity between sentences and predefined relationship types, allowing the system to capture more nuanced connections between entities [15].

A principal attribute of STIXnet is its interactive Knowledge Base, which expands dynamically as new reports are processed. This adaptive database facilitates the system's incorporation of novel entities, thereby enhancing its extraction accuracy without necessitating comprehensive model retraining. The framework's modularity permits users to tailor the system to their specific requirements, resulting in a solution that is both scalable and adaptable for a variety of Information Extraction tasks. By organizing its pipeline into well-defined modules, STIXnet provides researchers and organizations with the capability to implement diverse configurations of entity and relation extraction techniques, contingent upon the type of CTI data being analyzed. Furthermore, the interactive nature of the Knowledge Base has been demonstrated to be highly advantageous, allowing STIXnet to sustain its performance over time, even as new entities and relationships emerge within CTI reports.

3.2. aCTIon

As part of their work for NEC Corporation, Siracusano *et al.* (2023) proposed aCTion, a structured approach for extracting CTI information. It leveraged Large Language Models (LLMs) such as GPT-3.5 to automate CTI extraction from unstructured data sources and improve the existing methods of converting CTI reports into STIX format [16].

The methodologies employed in this study emphasize the development of a robust framework for information extraction. This encompasses the preprocessing of CTI data to address input size limitations and mitigate the hallucinations frequently associated with Large Language Models (LLMs). The procedure is bifurcated into two distinct phases: initially condensing the input text, followed by the application of extraction and verification prompts for entity classification. Specifically, the entity and relation extraction framework leveraged LLM-generated summaries to efficiently process extensive CTI reports. Concurrently, another specialized pipeline concentrated on extracting attack patterns by employing a synthesis of entity embeddings and LLM reasoning to discern patterns articulated across sentences.

The paper also introduces an extensive benchmark dataset comprising 204 real-world CTI reports converted into STIX bundles. The researchers designed the dataset to address the lack of large, open datasets for benchmarking structured CTI extraction tools. They evaluate the effectiveness of aCTIon by comparing it against ten baseline tools. The system outperformed these baselines in several tasks, achieving notable improvements in F1-scores for entity and attack pattern extraction [16].

Through the integration of large language model capabilities with specialized cybersecurity data, the aCTIon tool enhances the automation process in the extraction of structured cybersecurity threat intelligence. Nonetheless, the system maintains a human-in-the-loop model to review the structured outputs, thereby ensuring their accuracy.

4. Research Motivation

Research Gaps

While previous research has offered some promise in terms of automating CTI extraction, there is still no known LLM multiagent solution to automate the end-to-end conversion of STIX graphs into reports and vice versa. Current tools, such as STIXnet, have offered some promise by partially automating CTI extraction in a modular approach, but this makes it very dependent on the quality and completeness of the knowledge base [15]-[18]. If the knowledge base is not updated frequently, or if there are gaps in the available data, this could result in missing or misclassified entities. Meanwhile, while aCTion introduced an LLM model into its pipeline for information extraction, it still requires extensive manual validation (human in the loop) to avoid introducing errors into the knowledge base, which may slow down the system's long-term scalability. Ultimately, both STIXnet and

aCTion rely extensively on human analysts for verification and synthesis when combining outputs from their modular components.

Unlike STIXnet and aCTIon, STIXAgent employs a multi-agent workflow that automates STIX conversion end-to-end, reducing human dependency. The self-reasoning ability of a multi-agent system can provide a more adaptive and scalable approach, ensuring efficient and accurate extraction and structuring of CTI data [9]. Specifically, each individual agent not only gathers information but also stores relevant details in a shared memory state, allowing them to pass knowledge to other agents in the system. This shared memory enables seamless communication between agents, eliminating the need to reprocess information or duplicate efforts. Additionally, tasks can be broken down into smaller components and assigned to specialized agents best suited to handle them, optimizing efficiency and coordination across the system. Table 1 outlines the entities that are employed in STIXAgent in comparison to other related works.

Table 1. Comparison of the types of entities extracted by STIXAgent.

CTI Models	Attack Pattern	Campaign	Course of Action	Grouping	Identity	Indicator	Infrastructure	Intrusion Set	Location	Malware	Malware Analysis	Note	Observed Data	Opinion	Report	Threat Actor	Tool	Vulnerability	Incident
Weerawardhana *et al.* [19]	✓		✓			✓													
Li *et al.* [20]	✓	✓			✓														
Zhou (2022) *et al.* [21]	✓	✓	✓		✓														
Zhou (2023) *et al.* [22]	✓	✓	✓		✓	✓													
Ranade *et al.* [23]	✓	✓	✓		✓	✓	✓		✓										
Wang *et al.* [24]	✓	✓	✓		✓	✓	✓	✓		✓					✓	✓	✓	✓	
Siracusano *et al.* [16]	✓	✓	✓		✓	✓	✓	✓		✓					✓	✓	✓	✓	
Marchiori *et al.* [15]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
STIXAgent	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

For STIXAgent, it is the prompt template that is crafted to generate a STIX 2.1-compliant schema that adheres to the 19 standard STIX objects defined by OASIS. However, the golden STIX schema, provided in the Appendix, extends this structure by incorporating MITRE ATT&CK-based techniques and procedures, as this allows immediate applicability to this widely used ATP-based framework. The advantage of this approach is that it allows mapping to procedures and techniques that may be used to detect these threats. This hybrid approach enhances extensibility and aligns with modern Cyber Threat Intelligence (CTI) practices, as many cybersecurity platforms-such as MISP, OpenCTI, and Threat-Connect-integrate ATT&CK with STIX to provide deeper contextual analysis of threats.

5. Proposed Approach

In response to the previously identified concerns, the current authors introduce STIXAgent, an innovative framework leveraging LangGraph—a multi-agent system designed to systematically orchestrate tasks across specialized agents—and integrating multiple Large Language Models (LLMs). Specifically, GPT-4o and Gemini represent proprietary models known for stable and robust performance, while Llama 3 (8B), Qwen 32B, and Llama 70B (both DeepSeek-R1 distilled versions) represent open-source alternatives. DeepSeek-R1 distilled models, such as Qwen 32B and Llama 70B, are lighter, optimized versions designed to reduce computational overhead while maintaining competitive performance, thus offering faster response times but potentially at the expense of some model accuracy or consistency. STIXAgent interfaces with these LLMs through the Groq API, selected explicitly to ensure rapid, low-latency access. Utilizing LangGraph's modular approach, STIXAgent efficiently orchestrates sequential operations essential for converting unstructured Cybersecurity Threat Intelligence (CTI) into structured, STIX-compliant JSON outputs, thus addressing the operational needs of cybersecurity professionals effectively.

5.1. System Architecture

STIXAgent's architecture leverages LangGraph and the aforementioned 5 LLMs to structure and validate cybersecurity threat intelligence efficiently. The LangGraph agentic framework was employed due to its capacity to orchestrate sequential operations through the utilization of modular nodes and predefined transitions. In addition, bespoke tools were also specifically written as function calls to address particular facets of the STIX format, such as entity extraction, UUID generation, and STIX validation. The modular nature of LangGraph permits the interchangeability of LLMs within the workflow, thereby offering a framework that accommodates diverse model inputs while maintaining consistency in output.

The workflow begins with an input dataset consisting of cybersecurity reports, threat intelligence feeds, or structured data sources. This input is processed using a prompt template designed to enforce a baseline JSON STIX format, ensuring that the extracted information follows a standardized structure. By applying predefined formatting rules at the outset, the system minimizes inconsistencies and ensures that all extracted threat intelligence elements conform to industry standards.

Once the initial STIX-formatted output is generated, it progresses through a series of LangGraph nodes, each responsible for verifying and refining different aspects of the data. The first step focuses on entity extraction and structuring, where key cybersecurity-related entities-such as vulnerabilities, malware, and attack techniques-are identified and mapped to the appropriate STIX schema. This structured representation ensures that threat intelligence data remains both machine-readable and actionable.

Following entity extraction, the workflow assigns Unique Identifiers (UUIDs)

to each entity, enabling traceability throughout the processing pipeline. These identifiers ensure that extracted elements remain consistent across different stages and allow for efficient referencing in downstream analysis. The system then conducts STIX compliance validation, where the structured intelligence is checked for adherence to the STIX standard. This step ensures that all necessary fields are populated correctly and that the data conforms to established schema requirements.

As the structured output moves further through the pipeline, the system incorporates an error-handling and iterative refinement mechanism. This step identifies potential gaps, missing fields, or inconsistencies in the extracted data. If any errors are detected, the system iterates over previous steps to correct and refine the information, ensuring completeness and accuracy before finalization.

In the final stage, the system consolidates and validates the structured threat intelligence, preparing it for export. The final output consists of a CSV file containing unique entity identifiers and their corresponding classifications, ready for further analysis or integration into cybersecurity workflows. By structuring the intelligence processing pipeline within LangGraph, this system ensures that each stage maintains STIX adherence while tracking transformations across different nodes, minimizing errors and enhancing the reliability of extracted threat intelligence.

5.2. Evaluating STIX JSON Output

For the purpose of evaluation, we will substitute the LLMs tasked with generating JSON outputs with agents while maintaining the integrity of the existing agentic framework and workflow. Each LLM agent, guided by the LangGraph-directed workflow, autonomously produces JSON outputs designed to adhere to uniform standards of STIX compliance. Furthermore, STIXAgent makes reference to a prompt template within the analyze_report node. By standardizing the structure and content mandated from each model, we preserve control over the variables, thus facilitating a more objective assessment of each model's efficacy in generating STIX-compliant outputs. This arrangement enables a controlled evaluation across a spectrum of models, thereby unveiling potential strengths or limitations in each LLM's capacity to encapsulate nuanced cybersecurity threat intelligence data.

To impartially assess the quality of JSON outputs generated by each LLM, we propose a comprehensive evaluation framework based on content accuracy and structural similarity. Content accuracy entails verifying that the values produced by the LLM models align with those found in the reports, whereas structural consistency is assessed based on conformity to a specified JSON schema. Content accuracy can be evaluated by examining the presence and precision of specific Indicators of Compromise (IoCs) and threat descriptors, with the accuracy of each field contributing to a composite score that reflects the model's compliance with CTI requirements [15] [17]. Conversely, structural similarity can be measured through schema-matching algorithms, ensuring that the LLMs do not merely gen-

erate JSON in the correct format but also accurately represent the relationships and attributes outlined in the prompt.

6. Detailed Implementation

6.1. Data Collection

The data acquisition process entailed the extraction of approximately 4000 reports from two esteemed cybersecurity sources: Cisco Talos Intelligence and Microsoft's Security Blog. These sources are recognized for their comprehensive examination of Threat Tactics, Techniques, and Procedures (TTPs), which offer essential insights into emerging cyber threats and defensive strategies. While a larger dataset could provide more extensive coverage, a compromise between scale and feasibility was necessary to ensure efficient model evaluation and interpretability. Four hundred reports per source were chosen as an optimal subset, striking a balance between data diversity and computational feasibility. This sample size ensures sufficient variety in threat intelligence narratives while remaining manageable for thorough model assessment within a reasonable timeframe. The selection of 400 reports provides broad coverage of attack techniques across multiple campaigns and adversaries, ensuring the dataset remains comprehensive without being overwhelmingly large. This approach allows for meaningful comparisons between models while retaining the depth and quality needed for accurate cybersecurity NLP evaluations.

Rather than relying on a single dataset size, model evaluations were conducted across 10, 40, 100, and 400 reports to examine how model performance scales with increasing data. Testing with 10 reports highlights model robustness in low-data conditions, where variance is high and outputs may be unstable. Evaluations on 40 and 100 reports allow us to observe when model performance stabilizes, indicating whether a model generalizes well beyond a small subset. Finally, testing on 400 reports ensures that models maintain consistency across a substantial dataset, revealing whether their performance trends hold when exposed to a diverse and representative sample of real-world threat intelligence data. This multi-scale approach ensures a rigorous and scalable evaluation, capturing how models behave under varying data availability conditions.

6.2. Custom Tool Selection

Several custom tools have been crafted to fulfill specific roles in the workflow, ensuring precision in parsing, validation, and information retrieval (see **Figure 1**). Key tools include:

1) CyberEntityExtractionTool: This tool extracts cybersecurity-specific entities (e.g., IP addresses, URLs, CVE IDs, etc.) from input text using regular expressions. By isolating threat indicators directly from raw text, it reduces manual preprocessing and enables precise entity identification for threat intelligence.

2) UUIDGeneratorTool: To maintain the STIX 2.1 standard, each entity requires a unique identifier. This tool ensures consistent and reliable UUID gener-

ation, which is critical for data integrity across various cybersecurity analyses.

3) STIXFormatValidatorTool: Verifying JSON structure adherence to the STIX standard is crucial. This tool ensures the output aligns with the required format, flagging any structural issues before data distribution.

4) analyze_report: This tool uses a custom prompt to guide the LLM in analyzing a report and structuring the data according to the STIX JSON format. The prompt provides a robust template to ensure each element (indicator, malware, relationship) adheres to STIX guidelines.

5) fetch_missing_info: This tool addresses gaps in extracted data by performing web searches (using the Serp API) to retrieve additional context. If critical fields are missing, it allows the workflow to fill these gaps, enhancing the completeness and utility of the final output.

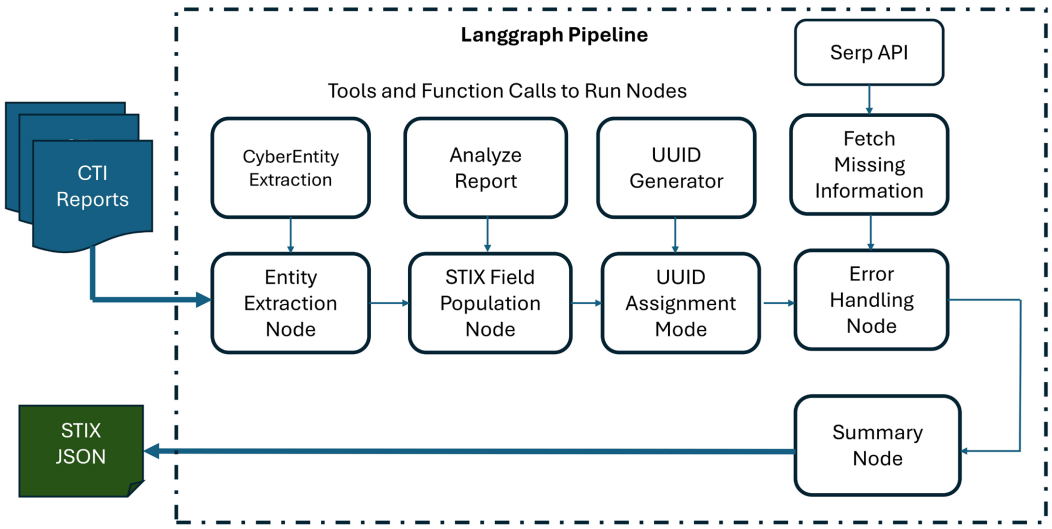

Figure 1. STIXAgent architecture.

6.3. Agentic Workflow

The STIXAgent workflow, comprising interconnected nodes, orchestrates the transformation of raw text into a STIX-compliant JSON report. Below is a breakdown of each workflow step:

1) Entity Extraction: The workflow begins with the CyberEntityExtractionTool, isolating relevant entities (e.g., IPs, URLs, CVEs). Extracted entities form the foundational data to populate the STIX structure.

2) STIX Field Population: Using analyze_report, this node organizes extracted data into the STIX JSON format, guided by a structured prompt template. This step defines threat intelligence data within a standardized schema, making it usable for further analysis or sharing.

3) UUID Assignment: Each extracted entity is assigned a UUID via the UUIDGenerator-Tool to uniquely identify elements in the STIX bundle. This step enhances the traceability of each entity within the threat intelligence structure.

4) Validation: The STIXFormatValidatorTool then validates the JSON output

against STIX 2.1 standards. If the JSON structure is non-compliant, it flags missing or incorrect fields for further review.

5) Error Handling and Data Enrichment: If validation reveals missing fields, the workflow triggers fetch_missing_info to retrieve additional context via web searches. This step allows the workflow to augment missing details, ensuring complete and accurate threat intelligence.

6) Final Summary: The final node provides a summary of actions, verifies successful JSON generation, and identifies whether additional information was fetched. It serves as a checkpoint to confirm the quality and completeness of the STIX output.

7. Evaluation Results

The production of Cyber Threat Intelligence (CTI) artifacts in the Structured Threat Information Expression (STIX) format through LLM agentic frameworks represents a novel methodology, accompanied by distinctive challenges that demand the formulation of a robust evaluation framework to critically assess the efficacy of LLMs. This research develops and implements a comprehensive multi-stage evaluation methodology that includes schema design, error analysis, and Bayesian statistical modeling to benchmark the performance of multiple LLMs, ensuring that we can effectively model uncertainty with posterior distributions.

7.1. Content Accuracy

To assess the accuracy of the content between the report and the semantic representations encapsulated by the STIX objects, a data manipulation methodology was employed. Initially, we isolate particular fields within the STIX output column formatted in JSON from datasets generated by the 5 LLMs (GPT4o, Gemini, Llama3 (8B), DeepseekR1-distilled Qwen32B & LLama 70B) as applied to Talos and Microsoft reports. Each dataset row undergoes iterative processing to verify that the STIX output field comprises valid JSON strings prior to parsing. Rows containing data that are not strings or contain invalid JSON are omitted, and such errors are documented for subsequent analysis. For rows maintaining valid JSON, the array of objects within the STIX output is extracted, and fields such as "name" and "type" are acquired for each object.

Following the initial extraction of the name and type fields from the JSON-formatted STIX output, this step involves verifying whether the extracted name values exist within the corresponding report content for the LLM-generated dataset. To achieve this, a function is used to normalize the text by converting it to lowercase, removing punctuation, and stripping whitespace. This ensures consistency in comparisons by minimizing variations due to formatting differences.

Each name field undergoes normalization in conjunction with the report content against which it is evaluated. Subsequently, tokenized fuzzy matching is employed to assess the similarity between the extracted name and discrete tokens from the normalized report content. The fuzzy matching score is determined uti-

lizing the partial_ratio function from the fuzzywuzzy library. A threshold of 60, which we found sufficient based on our testing, is established to ascertain the presence of the name within the report content, declaring it present if the maximum fuzzy score meets or exceeds this threshold. This approach yields a binary outcome (1 indicating presence, 0 indicating absence), thereby facilitating a systematic evaluation of content congruity.

For each row, diagnostic outputs provide the name, its highest fuzzy match score, and its presence within the report content, accompanied by a content preview for validation purposes. The outcomes, comprising the Row, Type, Name, original content, and the binary existence indication, are compiled into a results list. Any errors encountered during processing, such as absent content, are managed methodically, and the row is recorded with a default "not found" status for the existence check. This method ensures that the extracted name fields from the STIX output correspond with the report data, thereby affirming the semantic consistency of the generated outputs.

7.2. Structural Similarity Tests

The second set of evaluations involved a more complicated approach. First, we developed a custom JSON schema tailored to the nuanced requirements of STIX outputs. The schema was constructed using human annotations of STIX graphs by domain experts, leveraging handcrafted annotations to define field-level constraints, valid object types, and acceptable patterns. By doing so, there is now a means to assess the structural validity of LLM-generated outputs, ensuring alignment with domain expectations and operational use cases.

Using the crafted schema, a comprehensive error detection process was applied to LLM-generated STIX JSON outputs. Errors were categorized into three primary types: Regex Match Errors, Enumeration Errors, and JSON Decode Errors, which provided insight into the specific challenges faced by models in generating schema-compliant outputs.

Regex Match Errors occur when specific fields in the generated STIX JSON do not conform to predefined patterns or regular expressions that define acceptable values. These patterns are crucial for maintaining consistency and machine readability, especially for fields like "id" and "object" references, which must follow STIX conventions.

Enumeration errors occur when a field that is expected to contain a value from a predefined list of acceptable options instead contains an invalid or unrecognized value. This is particularly relevant for fields like kill_chain_phases or phase_name, which rely on controlled vocabularies defined by standards such as the MITRE ATT & CK framework. For example, in the kill_chain_phases, phase_name field, the expected values might include options like reconnaissance, weaponization, or exploitation. An enumeration error arises if the model generates a value like exploring or leaves the field empty.

JSON Decode Errors occur when the generated output is not syntactically valid

JSON. These errors can result from issues such as mismatched brackets, missing commas, or improperly formatted strings. While these errors are less frequent, they indicate a fundamental breakdown in structural adherence. For example, an output might contain unclosed braces ({\"type\": \"indicator\") or invalid syntax such as a missing comma between fields (\"id\": \"indicator\u20131234abcd\", \"type\": \"indicator\").

The Kruskal-Wallis test, serving as a non-parametric alternative to ANOVA, was employed to assess whether structural similarity errors differed significantly across report sizes (10, 40, 100, and 400 reports). Kruskal-Wallis tests were conducted separately for each LLM. Each test included observations of five types of structural errors per report-size group. No statistically significant differences were found for any of the evaluated language models:

- GPT-4o: $H(3) = 3.929$, $p = 0.269$
- Gemini: $H(3)=1.841$, $p = 0.606$
- Llama 3: $H(3) = 5.105$, $p= 0.164$
- Qwen 32B: $H(3) = 3.892$, $p= 0.273$
- Llama 70B: $H(3) = 6.539$, $p= 0.088$

These results indicate that the report size did not significantly impact the structural compliance of the generated STIX JSON outputs.

7.3. Bayesian Analysis of LLM Performance Based on Structural Similarity Scores

In accordance with a specified JSON schema, we calculate a comprehensive score reflecting the degree of conformity of a JSON string to the given schema. This is accomplished through the generation of a validation score, accompanied by a detailed analysis of any discrepancies. The formula used is as follows:

$$\text{Score} = \left(\frac{\text{Matched Fields}}{\text{Total Fields}} \right) \times 100. \quad (1)$$

The process of matching commences with an attempt to parse the JSON string into a Python dictionary. Should the parsing process fail due to invalid JSON syntax, the function subsequently assigns a score of 0 and outputs an error message elucidating the issue. Upon successful parsing, the JSON data undergoes validation against the schema utilizing the Draft7Validator from the jsonschema library, which detects any infringements of the schema's structural or constraint-based parameters. The function proceeds to compute the total number of fields present in the schema, including those that are nested, by employing the count_schema_fields helper function. Thereafter, the function ascertains the number of matched fields by subtracting the number of validation errors from the total number of fields, ensuring a non-negative result. The ultimate score is then derived as the percentage of matched fields in relation to the total number of fields, thus offering a quantitative assessment of schema compliance. In the exceptional event that the schema lacks any fields, the score defaults to 0.

To assess the adherence of generated STIX structures to the expected schema,

we employ Bayesian inference modeling, which enables a probabilistic estimation of posterior distributions for LLM performance across different datasets. Table 3 presents the mean schema compliance scores alongside their corresponding variance estimates, providing a quantitative measure of both the accuracy and consistency of structured outputs. The adoption of Bayesian modeling is attributed to its capacity to produce posterior distributions that afford more comprehensive insights into the spectrum of plausible values for each parameter. This approach is essential in this context, as it transcends mere interest in the average performance of each model to encompass the variability and uncertainty in their scores across diverse datasets and scenarios. By employing Bayesian inference, the analysis adeptly captures the intrinsic uncertainty within the data, thereby providing High-Density Intervals (HDIs) to articulate confidence in the resultant estimates.

A key advantage of this Bayesian approach lies in its ability to assess model consistency, where lower variance values indicate greater stability in schema adherence across different cybersecurity reports. Additionally, the mean compliance scores serve as a benchmark for evaluating the overall reliability of each LLM in generating syntactically and semantically valid STIX structures.

For this, we contrasted the model performance of all 5 LLMs for 10, 40, 100, and 400 reports for Microsoft-based reports, then did the same for Talos reports.

This implementation uses standard uninformative priors, such as normal priors centered at 50 for parameter μ, with a large standard deviation, allowing data to mainly influence outcomes. The likelihood function incorporates observed data, producing posterior distributions for each model's performance (μ) and variability (σ), offering a probabilistic view of performance and consistency while accounting for uncertainty.

7.4. Model Performance on the Talos Dataset

Referencing Table 2, we assess the performance of the different language models (GPT-4o, Gemini, Llama 3, Qwen 32B, and Llama 70B) on the Talos dataset (10, 40, 100, and 400 reports), which highlights distinct patterns of model behavior. Gemini consistently achieved the highest mean performance across all dataset sizes (10, 40, 100, and 400 reports), notably excelling with smaller datasets with 98.360 for 10 reports and further improving to 98.951 at 400 reports, demonstrating exceptional generalizability and consistent adaptation. GPT-4o exhibited relatively lower initial performance (mean = 82.402 at 10 reports), but quickly improved with larger dataset sizes (mean = 96.758 at 40 reports, stabilizing around 96.528 at 400 reports), reflecting good scalability. In contrast, Llama 3 initially struggled, particularly with smaller datasets (mean = 66.894 at 10 reports), accompanied by notable variability (SD = 8.913), highlighting initial challenges in adapting to complex tasks, although performance stabilized at larger sizes (mean = 96.802 at 400 reports). Qwen 32B demonstrated a similar trend, beginning with modest performance (mean = 67.805 at 10 reports) and substantially improving at larger scales (mean = 96.970 at 400 reports). Llama 70B initially struggled sig-

nificantly (mean = 46.296 at 10 reports), indicating pronounced difficulty with limited data, but notably improved its performance at larger dataset sizes, reaching a mean of 96.770 at 400 reports. These findings underscore clear variations in each model's adaptability, with Gemini consistently outperforming others, GPT-4o showing strong scalability, and the remaining models improving notably only as dataset size increased.

Table 2. Talos report bayesian summary results.

Reports	GPT-4o	Gemini	Llama3.2	Qwen-32B	Llama-70B
			Mean/Std Dev		
Reports	82.4/7.4	98.4/1.2	66.9/8.9	67.8/9.1	46.3/9.1
Reports	96.8/0.7	98.3/0.5	96.2/1.1	98.2/0.4	97.3/0.7
Reports	95.4/0.8	98.5/0.3	97.1/0.6	97.9/0.4	97.0/0.4
Reports	96.5/0.3	99.0/0.1	96.8/0.3	97.0/0.2	96.8/0.2

7.5. Model Performance on the Microsoft Dataset

Referencing Table 3, we assess the performance of the evaluated language models (GPT-4o, Gemini, Llama 3, Qwen 32B, and Llama 70B) across the Microsoft dataset at varying report sizes (10, 40, 100, and 400 reports). Gemini consistently exhibited the highest mean performance, peaking at a mean score of 98.335 at 100 reports, showcasing robust stability and generalization capabilities across different scales. GPT-4o maintained strong performance at smaller dataset sizes (mean = 97.120 at 100 reports) but demonstrated a slight decrease in mean accuracy as dataset sizes increased beyond 10 reports. Llama 3 and Llama 70B had comparable intermediate performance scores, with means ranging from approximately 97.102 to 98.187, accompanied by consistent decreases in variability as dataset size grew (e.g., Llama 70B's variability decreased from 1.425 at 10 reports to 0.216 at 400 reports). Qwen 32B showed lower mean performance compared to Gemini and GPT-4o, stabilizing at around 97.389 at 100 reports but exhibiting improved reliability at larger scales (variability reduced from 1.254 at 10 reports to 0.187 at 400 reports). Collectively, these results indicate that Gemini is particularly effective for structural accuracy across dataset sizes, GPT-4o demonstrates consistent reliability, and models like Llama 3, Llama 70B, and Qwen 32B improve notably with increased dataset size, stabilizing in performance with larger data inputs.

Table 3. Microsoft report bayesian summary results.

Reports	GPT-4o	Gemini	Llama3.2	Qwen-32B	Llama-70B
			Mean/Std Dev		
Reports	99.4/0.4	97.6/0.8	97.1/1.3	97.7/1.3	98.1/1.4
Reports	97.8/0.4	98.6/0.4	98.2/0.4	97.3/0.5	98.1/0.5
Reports	97.1/0.4	98.3/0.2	97.1/0.5	97.4/0.4	97.5/0.4
Reports	96.6/0.3	98.7/0.1	97.4/0.2	97.4/0.2	97.5/0.2

8. Comparison with Related Work

Contrary to preceding methodologies, including STIXnet and "Time for aCTIon", which necessitate substantial human supervision to maintain data fidelity, this multi-agent LLM-driven framework offers a scalable and adaptable solution for automating the conversion of CTI reports into STIX-compliant formats.

At the outset, this framework utilizes a multi-agent configuration in which various Large Language Models (LLMs) are employed for the generation and assessment of STIX JSON outputs. The incorporation of customized function calls alongside a prompt template fosters adherence to the designated prompt template. Moreover, the implementation of a JSON schema as a benchmark underscores a workflow that can be readily expanded without requiring extensive alterations.

Secondly, despite the intrinsic risk posed by Large Language Models (LLMs) in generating hallucinations, the STIXAgent incorporates comprehensive validation mechanisms, particularly through validation procedures within the LangGraph framework. These error-handling protocols meticulously assess each output for conformity with the STIX schema and prompt template specifications, thereby enabling the framework to detect and correct discrepancies at an early stage. Although these procedures do not completely eradicate the potential for hallucinations, they offer a robust safeguard that improves the reliability of outputs and reduces the probability of erroneous data being integrated into the final STIX JSON structure.

Furthermore, we have developed a novel multistage evaluation framework that emphasizes both content accuracy and structural similarity, as opposed to exclusively depending on conventional evaluation metrics grounded in machine learning. We propose that this framework offers a more precise evaluation of model idiosyncrasies across various datasets and represents a more effective strategy for future advancements in prompt engineering and modifications to the agentic workflow.

9. Conclusion

This method, which employs a multi-agent LLM configuration, conducts structured correlation analyses and integrates robust validation mechanisms, presenting a scalable and comprehensive solution for automated STIX generation. It advances beyond previous models such as STIXnet by reducing the necessity for manual oversight, systematically assessing model capabilities, and mitigating the risks of hallucinations in an organized manner. A key contribution of this work is the development of a multistage evaluation framework that redirects the emphasis from exact semantic matches-which previously constituted the sole metrics (precision/recall/F1 scores) in past studies-to a framework that considers the variability of STIX outputs as appraised by human annotators, and the generally dynamic nature of CTI reporting. From the results, we note the potential of the GPT-4o model as it demonstrates the highest consistency, while Gemini achieves strong schema adherence with some variability. Future work in this area can utilize this

framework to explore other foundational models, whilst also extending the work to larger datasets and more complex cybersecurity threat intelligence sources. Future work should explore parallelized processing techniques to enhance computational efficiency and further validate the scalability and trustworthiness of LLM-based threat intelligence automation.

Acknowledgements

We thank Aman Tiwari for feedback on the content of LLM outputs and human annotations that refined the JSON schema for better validation.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Zhang, J., Bu, H., Wen, H., Chen, Y., Li, L. and Zhu, H. (2024) When LLMs Meet Cybersecurity: A Systematic Literature Review.

[2] Chowdhary, K.R. (2020) Natural Language Processing. In: *Fundamentals of Artificial Intelligence*, Springer India, 603-649. https://doi.org/10.1007/978-81-322-3972-7_19

[3] Sun, P., Yang, X., Zhao, X. and Wang, Z. (2018) An Overview of Named Entity Recognition. 2018 *International Conference on Asian Language Processing* (*IALP*), Bandung, 15-17 November 2018, 273-278. https://doi.org/10.1109/ialp.2018.8629225

[4] Weegar, R. (2021) Applying Natural Language Processing to Electronic Medical Records for Estimating Healthy Life Expectancy. *The Lancet Regional Health—Western Pacific*, **9**, Article ID: 100132. https://doi.org/10.1016/j.lanwpc.2021.100132

[5] Mesbah, S., Lofi, C., Torre, M.V., Bozzon, A. and Houben, G. (2018) TSE-NER: An Iterative Approach for Long-Tail Entity Extraction in Scientific Publications. In: Vrandečić, D., *et al.*, Eds., *The Semantic Web—ISWC* 2018, Springer International Publishing, 127-143. https://doi.org/10.1007/978-3-030-00671-6_8

[6] Nadgeri, A., Bastos, A., Singh, K., Mulang', I.O., Hoffart, J., Shekarpour, S., *et al.* (2021) KGPool: Dynamic Knowledge Graph Context Selection for Relation Extraction. *Findings of the Association for Computational Linguistics*: *ACL-IJCNLP* 2021, August 2021, 535-548. https://doi.org/10.18653/v1/2021.findings-acl.48

[7] Piskorski, J. and Yangarber, R. (2012) Information Extraction: Past, Present and Future. In: Poibeau, T., *et al.*, Eds., *Multi-Source*, *Multilingual Information Extraction and Summarization*, Springer, 23-49. https://doi.org/10.1007/978-3-642-28569-1_2

[8] Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P. and Bernstein, M.S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *Proceedings of the* 36*th Annual ACM Symposium on User Interface Software and Technology*, San Francisco, 29 October-1 November 2023, 1-22. https://doi.org/10.1145/3586183.3606763

[9] Richards, T.B. (2023) AutoGPT: Autonomous GPT-3 Powered Task Solver. https://github.com/Significant-Gravitas/AutoGPT

[10] Li, X., Wang, S., Zeng, S., Wu, Y. and Yang, Y. (2024) A Survey on LLM-Based Multi-Agent Systems: Workflow, Infrastructure, and Challenges. *Vicinagearth*, **1**, Article No. 9. https://doi.org/10.1007/s44336-024-00009-2

[11] Porkorny, Z. (2018) What Are the Phases of the Threat Intelligence Lifecycle. The Threat Intelligence Handbook.

[12] Wagner, T.D., Mahbub, K., Palomar, E. and Abdallah, A.E. (2019) Cyber Threat Intelligence Sharing: Survey and Research Directions. *Computers & Security*, **87**, Article ID: 101589. https://doi.org/10.1016/j.cose.2019.101589

[13] Barnum, S. (2012) Standardizing Cyber Threat Intelligence Information with the Structured Threat Information Expression (STIX). *Mitre Corporation*, **11**, 1-22.

[14] You, Y., Jiang, J., Jiang, Z., Yang, P., Liu, B., Feng, H., *et al.* (2022) TIM: Threat Context-Enhanced TTP Intelligence Mining on Unstructured Threat Data. *Cybersecurity*, **5**, Article No. 3. https://doi.org/10.1186/s42400-021-00106-5

[15] Marchiori, F., Conti, M. and Verde, N.V. (2023) STIXnet: A Novel and Modular Solution for Extracting All STIX Objects in CTI Reports. *Proceedings of the* 18*th International Conference on Availability, Reliability and Security*, Benevento, 29 August-1 September 2023, 1-11. https://doi.org/10.1145/3600160.3600182

[16] Siracusano, G., *et al.* (2023) Time for Action: Automated Analysis of Cyber Threat Intelligence in the Wild.

[17] Perrina, F., Marchiori, F., Conti, M. and Verde, N.V. (2023) AGIR: Automating Cyber Threat Intelligence Reporting with Natural Language Generation. 2023 *IEEE International Conference on Big Data* (*BigData*), Sorrento, 15-18 December 2023, 3053-3062. https://doi.org/10.1109/bigdata59044.2023.10386116

[18] Alam, M.T., Bhushl, D., Nguyen, L. and Rastogi, N. (2024) CTIBench: A Benchmark for Evaluating LLMs in Cyber Threat Intelligence.

[19] Weerawardhana, S., Mukherjee, S., Ray, I. and Howe, A. (2015) Automated Extraction of Vulnerability Information for Home Computer Security. In: Cuppens, F., *et al.*, Eds., *Foundations and Practice of Security*, Springer International Publishing, 356-366. https://doi.org/10.1007/978-3-319-17040-4_24

[20] Li, T., Guo, Y. and Ju, A. (2019) A Self-Attention-Based Approach for Named Entity Recognition in Cybersecurity. 2019 15*th International Conference on Computational Intelligence and Security* (*CIS*), Macao, 13-16 December 2019, 147-150. https://doi.org/10.1109/cis.2019.00039

[21] Zhou, Y., Tang, Y., Yi, M., Xi, C. and Lu, H. (2022) CTI View: APT Threat Intelligence Analysis System. *Security and Communication Networks*, **2022**, Article ID: 9875199. https://doi.org/10.1155/2022/9875199

[22] Zhou, Y., Ren, Y., Yi, M., Xiao, Y., Tan, Z., Moustafa, N., *et al.* (2023) CDTier: A Chinese Dataset of Threat Intelligence Entity Relationships. *IEEE Transactions on Sustainable Computing*, **8**, 627-638. https://doi.org/10.1109/tsusc.2023.3240411

[23] Ranade, P., Piplai, A., Joshi, A. and Finin, T. (2021) CyBERT: Contextualized Embeddings for the Cybersecurity Domain. 2021 *IEEE International Conference on Big Data* (*Big Data*), Orlando, 15-18 December 2021, 3334-3342. https://doi.org/10.1109/bigdata52589.2021.9671824

[24] Wang, X., Liu, R., Yang, J., Chen, R., Ling, Z., Yang, P., *et al.* (2022) Cyber Threat Intelligence Entity Extraction Based on Deep Learning and Field Knowledge Engineering. 2022 *IEEE* 25*th International Conference on Computer Supported Cooperative Work in Design* (*CSCWD*), Hangzhou, 4-6 May 2022, 406-413. https://doi.org/10.1109/cscwd54268.2022.9776139

Appendix A. STIX Golden Template

Below is the STIX JSON schema that is used to reference the preferred output JSON data structure.

Listing 1. Golden STIX schema.

```
golden_stix_schema = {
    "$schema": "http://json-schema.org/draft-07/
    schema#",
    "type": "object",
    "properties": {
        "type": {"type": "string", "enum": ["
    bundle"]},
        "id": {"type": "string", "pattern": "^
    bundle--[a-z0-9\\-]+$"},
        "objects": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "type": {
                        "type": "string",
                        "enum": [
                            "indicator",
                            "malware",
                            "relationship",
                            "course-of-action",
                            "attack-pattern",
                            "intrusion-set",
                            "campaign",
                            "vulnerability",
                            "mitigation",
                            "identity",
                            "threat-actor",
                            "tool",
                            "report",
                            "vocabulary",
                            "technique",
                            "procedure"
                        ]
                    },
                    "id": {
                        "type": "string",
                        "pattern": "^((indicator|
    malware|relationship|course-of-action|attack-
    pattern|intrusion-set|campaign|vulnerability|
    mitigation|identity|threat-actor|tool|report|
    vocabulary|technique|procedure)--[a-z0
    -9\\-]+|[a-zA-Z0-9\\-]+)$"
                    },
                    "name": {"type": ["string", "
    null"]},
                    "description": {"type": ["
    string", "null"]},
                    "labels": {
                        "type": "array",
                        "items": {"type": "string
    "}
                    },
                    "goals": {
                        "type": "array",
                        "items": {"type": "string
    "}
                    },
                    "indicators_of_compromise": {
     # Added field
                        "type": "array",
                        "items": {"type": "string
    "}
                    },
                    "malware": {  # Added field
                        "type": "array",
                        "items": {"type": "string
    "}
```

```
                },
                "resource_level": {"type": ["
string", "null"]},
                "secondary_motivation": {  #
Added field
                    "type": ["string", "array
", "null"],
                    "items": {"type": "string
"}
                },
                "sophistication": {  # Added
field
                    "type": ["string", "null"]
                },
                "tactics": {  # Added field
                    "type": "array",
                    "items": {"type": "string
"}
                },
                "techniques": {  # Added field
                    "type": "array",
                    "items": {"type": "string
"}
                },
                "motivation": {
                    "type": ["string", "array
", "null"],
                    "items": {"type": "string
"}
                },
                "personal_motivation": {
                    "type": ["string", "array
", "null"],
                    "items": {"type": "string
"}
                },
                "primary_motivation": {
                    "type": ["string", "array
", "null"],
                    "items": {"type": "string
"}
                },
                "vulnerability_types": {
                    "type": "array",
                    "items": {"type": "string
"}
                },
                "mitigation_types": {
                    "type": "array",
                    "items": {"type": "string
"}
                },
                "indicator_types": {
                    "type": "array",
                    "items": {"type": "string
"},
                    "minItems": 1
                },
                "pattern": {"type": ["string",
 "null"]},
                "pattern_type": {"type": ["
string", "null"], "enum": ["stix", None]},
                "valid_from": {"type": ["
string", "null"], "format": "date-time"},
                "malware_types": {
                    "type": "array",
                    "items": {"type": "string
"},
                    "minItems": 1
                },
                "is_family": {"type": ["
boolean", "null"]},
                "kill_chain_phases": {
                    "type": "array",
```

```
                    "items": {
                        "type": "object",
                        "properties": {
                            "kill_chain_name":
 {"type": "string"},
                            "phase_name": {
                                "type": "
string",
                                "enum": [
                                    "
reconnaissance",
                                    "resource-
development",
                                    "initial-
access",
                                    "execution
",
                                    "
persistence",
                                    "privilege
-escalation",
                                    "defense-
evasion",
                                    "
credential-access",
                                    "discovery
",
                                    "lateral-
movement",
                                    "
collection",
                                    "command-
and-control",
                                    "
exfiltration",
                                    "impact",
                                    "espionage
",
                                    "
establishing long-term access"
                                ]
                            }
                        },
                        "required": ["
kill_chain_name", "phase_name"],
                        "additionalProperties
": False
                    }
                },
                "relationship_type": {"type":
"string"},
                "source_ref": {"type": "string
"},
                "target_ref": {"type": "string
"},
                "action_type": {
                    "type": "array",
                    "items": {"type": "string
"}
                },
                "external_references": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "source_name": {"
type": "string"},
                            "external_id": {"
type": "string"}
                        },
                        "required": ["
source_name"],
                        "additionalProperties
": False
```

```
                                }
                            },
                            "aliases": {
                                "type": "array",
                                "items": {"type": "string
"},
                                "minItems": 1
                            },
                            "identity_class": {"type": ["
string", "null"]},
                            "sector": {
                                "type": ["string", "array
"],
                                "items": {"type": "string
"},
                                "minItems": 1
                            },
                            "first_seen": {"type": ["
string", "null"], "format": "date-time"},
                            "last_seen": {"type": ["string
", "null"], "format": "date-time"},
                            "sectors": {
                                "type": ["string", "array
"],
                                "items": {"type": "string
"},
                                "minItems": 1
                            },
                            "created": {"type": ["string",
 "null"], "format": "date-time"},
                            "modified": {"type": ["string
", "null"], "format": "date-time"},
                            "spec_version": {"type": ["
string", "null"]},
                            "object_refs": {
                                "type": "array",
                                "items": {"type": "string
"}
                            },
                            "published": {"type": ["string
", "null"], "format": "date-time"}
                        },
                        "required": ["type", "id"],
                        "additionalProperties": False
                    }
                }
            },
            "required": ["type", "id", "objects"],
            "additionalProperties": False
        }
```

Error Summary for All Models
Talos Reports

Table A1. Error summary for Talos 10.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	0	0	0	0	0
Regex Match Error	0	0	3	6	2
Enumeration Error	18	6	7	10	2
Missing Value Error	0	0	0	0	0
Other	0	1	0	4	0

Table A2. Error summary for Talos 40.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	2	0	1	1	1
Regex Match Error	0	0	19	18	22
Enumeration Error	31	23	27	9	24
Missing Value Error	0	0	0	0	0
Other	27	9	23	6	3

Table A3. Error summary for Talos 100.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	2	0	3	7	5
Regex Match Error	0	2	48	37	33
Enumeration Error	84	45	56	41	60
Missing Value Error	0	0	0	0	0
Other	131	24	29	17	42

Table A4. Error summary for Talos 400.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	3	4	4	8	6
Regex Match Error	3	0	48	37	33
Enumeration Error	306	154	280	265	284
Missing Value Error	0	0	0	0	0
Other	352	45	279	267	292

Microsoft reports

Table A5. Error summary for Microsoft 10.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	0	0	0	0	1
Regex Match Error	0	0	7	6	5
Enumeration Error	3	6	6	3	2
Missing Value Error	0	0	0	0	0
Other	10	5	0	1	0

Table A6. Error summary for Microsoft 40.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	0	0	5	0	0
Regex Match Error	0	0	15	31	18
Enumeration Error	18	19	10	20	14
Missing Value Error	0	0	0	0	0
Other	24	8	5	1	4

Table A7. Error summary for Microsoft 100.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	1	2	3	5	2
Regex Match Error	0	0	35	44	47
Enumeration Error	57	52	63	59	45
Missing Value Error	0	0	0	0	0
Other	79	26	36	15	26

Table A8. Error summary for Microsoft 400.

Error Type	GPT-4o	Gemini	Llama 3	Qwen 32B	Llama 70B
JSON Decode Error	5	2	13	22	11
Regex Match Error	7	1	180	190	195
Enumeration Error	307	162	204	197	170
Missing Value Error	0	0	0	0	0
Other	331	77	100	82	105