Inferring Recovery Steps from Cyber Threat Intelligence Reports

Zsolt Levente Kucsván[1], Marco Caselli[3], Andreas Peter[1,2], Andrea Continella[1]

[1] University of Twente, Enschede, The Netherlands
{z.kucsvan,a.continella}@utwente.nl
[2] Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany
andreas.peter@uni-oldenburg.de
[3] Siemens AG, Munich, Germany
marco.caselli@siemens.com

Abstract. Within the constantly changing threat landscape, Security Operation Centers are overwhelmed by suspicious alerts, which require manual investigation. Nonetheless, given the impact and severity of modern threats, it is crucial to quickly mitigate and respond to potential incidents. Currently, security operators use predefined sets of actions from so-called playbooks to respond to incidents. However, these playbooks need to be manually created and updated for each threat, again increasing the workload of the operators. In this work, we research approaches to automate the inference of recovery steps by automatically identifying steps taken by threat actors within Cyber Threat Intelligence reports and translating these steps into recovery steps that can be defined in playbooks. Our insight is that by analyzing the text describing threats, we can effectively infer their corresponding recovery actions. To this end, we first design and implement a semantic approach based on traditional Natural Language Processing techniques, and we then study a generative approach based on recent Large Language Models (LLMs). Our experiments show that even if the LLMs were not designed to solve domain-specific problems, they outperform the precision of semantic approaches by up to 45%. We also evaluate factuality showing that LLMs tend to produce up to 90 factual errors over the entire dataset.

Keywords: CTI reports · Recovery steps · LLM · Semantic model

1 Introduction

The number of threats that organizations face nowadays increases annually [12], reaching volumes up to multiple thousands per day [2]. To defend themselves, organizations employ teams that are in charge of monitoring the infrastructure, identifying attacks (*alert evaluation*), and responding to incidents (*incident response*). These teams operate in so-called Security Operation Centers (SOCs).

Both alert evaluation and incident response rely on heavy and repetitive manual tasks, which eventually lead to burnout and alert fatigue of the human operators [35]. In recent years, several works proposed approaches to reduce the workload of the security operators [38], [13], [10] and automate different tasks within SOCs, such as alert triaging [14], [5], [11]. However, most of the literature only focuses on the alert evaluation phase, leaving automation in the incident response phase mostly unaddressed.

Security operators use playbooks to respond to incidents, which are manually-crafted sets of instructions containing *actions* and *artifacts*—e.g., in the instruction **delete malicious executable** the action is **delete** and the artifact **malicious executable**. When being under attack, these documents help security operators identify adequate countermeasures. Creating playbooks is often a complex task, most importantly for organization-specific playbooks, which are influenced by many factors [31]. Hence, the creation of the playbooks often results in a bottleneck in terms of manual work and time consumption, again contributing to the alert fatigue of the operators.

In this work, we research approaches to automate the generation of the key elements of playbooks, namely the actions and artifacts of the countermeasure steps to be deployed on the affected systems to recover from threats. We refer to these steps as *recovery steps*. Note, that while generating the recovery steps is a step towards automating the playbook creation process, constructing playbooks is out of scope in our work.

In our context, we consider recovery steps that can be potentially automated, such as deleting files, restoring backups, or blocking connections. Actions that are not related to affected systems from a technical perspective (i.e., legal actions, policy-related actions, or similar) are out of scope. In particular, we focus on analyzing the behaviour of threats and extracting the threat actions to infer recovery steps. We take the commonly used Cyber Threat Intelligence (CTI) reports as a reference, as they contain descriptions of attacks written by security operators. These reports are written in natural language and require NLP techniques to analyze them in an automated system.

To the best of our knowledge, there are no prior works that explore automated inference of recovery steps for incident response. However, several prior works aim to solve some underlying problems, such as extracting the threat actor's behaviour or artifacts. Legoy et al. [19] proposed a multi-label classifier to extract attack behaviour; Lim et al. [21] proposed a machine learning based framework to annotate security-related text, detecting malware actions and capabilities; Park et al. [25] designed a semantic NLP based framework to extract information from CTI reports and identify relations between these. While the previously mentioned works are based on traditional approaches, it is important to mention the recent advances of large language models (LLMs) and their promising capabilities to solve domain-specific problems despite the fact that they were trained on a general corpus of data. The industry already started to propose solutions for *incident response* that are based on the LLMs, such as Microsoft Security Copilot, which delivers a step-by-step guide on how to investigate and respond to incidents [18], or the Trend Micro's Trend Companion AI, which helps security operators with information to accelerate response time [23]. Unfortunately, these tools are not openly available, nor researched properly. This stresses the importance of this work of comparing LLMs and traditional approaches for extracting CTI knowledge to infer actionable recovery steps.

In this paper, we compare traditional NLP techniques that are based on semantic models to generative models that rely on advanced LLMs. First, motivated by the lack of available systems, we design and implement our own semantic-based pipeline (Section 3.1). Second, we study and experiment with different types of LLMs to assess how they perform against semantic approaches. In our assessment, we include both an open-source LLM (LLaMa 2 [37]) and multiple versions of a closed-source LLM (GPT 3 [8] and GPT 4 [4]).

For the assessment, due to the unavailability of an existing dataset, we create a new labeled dataset of recovery steps from 200 CTI reports, randomly chosen from the Trend Micro Encyclopedia [1]. This dataset is composed of sentences labeled with their corresponding recovery steps for each report.

Our results show that both the semantic and generative models have their own advantages and disadvantages. The semantic model is more consistent, based on factual rules that give a better explainability, while LLMs outperform it up to 45% in precision and 53% in recall. On the other hand, LLMs can produce up to 90 factual errors over the entire dataset, that are inferred recovery steps unconnected to the information given in the input.

The contributions of our work are the following:

- We design the first system, based on traditional NLP techniques to analyze CTI reports and infer threat recovery steps.
- We build the first labeled dataset of recovery steps, composed of labeled sentences from 200 CTI reports.
- We evaluate semantic and generative approaches, assessing and comparing the performances and capabilities of NLP-based models against multiple off-the-shelf LLMs. Based on our empirical evaluation, we provide insights and suggestions for future research in this field.

In the spirit of open science, we provide public access to the dataset we created and the source code of the implemented semantic model at `https://github.com/utwente-scs/recovery-inference`.

2 Background & Challenges

In this section, we describe how CTI reports can be used as input for our research and how can we leverage different NLP approaches to process threat descriptions. Additionally, we describe the challenges we need to overcome for designing a recovery steps inferring system.

2.1 Cyber Threat Intelligence reports

CTI reports are documents with descriptions of threats, containing information such as attacker behavior, indicators of compromise (IOCs), or courses of action to mitigate the threat [30]. We consider the following simple CTI example:

The malware arrives on a system via a USB device. It **encrypts** the **folder**: %User Profile%\Documents. It **drops** a **malicious file** to the desktop. After the execution, the malware deletes itself.

By isolating core actions in the example above, such as **encrypts folder** or **drops malicious file**, a human operator, understanding the behavior of the malware, takes appropriate course of action to recover from the attack. This is useful, especially in cases, when the available playbooks are too generic, with high-level instructions instead of concrete actions for specific threats. Extending generic playbooks with information extracted from CTI reports results in a more comprehensive and complete course of action to recover from the threat, but without an automated system, this requires additional manual effort by searching, reading, and understanding reports.

2.2 Traditional NLP techniques

To process text written in natural language, we employ NLP techniques. Traditional NLP techniques that rely on capturing the meaning and relationships between words, through linguistic analysis are often used. Linguistic analysis in NLP offers a variety of tasks, such as *tokenization*, *lemmatization*, *part of speech (POS) tagging*, *dependency parsing*, etc. In the following, we give a short insight for each of these tasks, that will be used later on in this paper.

Tokenization is the process of splitting the text into multiple pieces. These pieces are usually words or parts of words. E.g., The malware deletes the file. can be tokenized into: [The], [malware], [deletes], [the], [file], [.].

Lemmatization is the process of transforming tokens into their dictionary base form to enhance text understanding. Due to the grammar, words might have different forms (e.g., delete, deleted, deletion) and by lemmatizing we make sure we identify all forms as the same word.

POS tagging is the process of labelling the words into their corresponding grammatical category, such as nouns or verbs.

Dependency parsing is the process of identifying connected tokens that create phrases. All the connections create a dependency tree that describes the grammatical structure of a sentence or text.

2.3 Generative LLMs

In the last couple of years, LLMs went through an impressive development, enabling them to address a wide range of general tasks. Especially they are very good at generating coherent, logical text based on an input prompt. Input prompts are usually instructions or questions fed into LLMs to initiate responses. Based on a prompt and the background knowledge of the acquired from the large corpora of training datasets, the LLMs provide a piece of text as output, which is statistically the most likely continuation of the initial prompt.

There are two main categories of generative LLMs: *open-source* LLMs that are open for the users, and *closed-source* LLMs that have their inner architecture hidden from the public. Another difference between these two types of LLMs is their training process. Open-source LLMs usually use public data, while closed-source LLMs mostly leverage proprietary datasets for the training.

2.4 Challenges

Extracting recovery steps involves multiple research domains with unsolved challenges, including Cyber Threat Intelligence and NLP-based Information Extraction. This section outlines these challenges and their relevance to our research.

Cyber Threat Intelligence. A research by Abu et al. [3] on Cyber Threat Intelligence emphasizes two main challenges relevant for our work: the large quantity of CTI data and their timeliness. Time to process the data is a crucial factor in incident response, to avoid as much damage as possible. Moreover, having a large quantity of CTI data implies an increased effort for security operators to search, analyze, and identify relevant information from the reports to respond to incidents. These challenges require automated systems to reduce manual effort for operators. Our research therefore focuses primarily on *automation*.

Information extraction. There are several solutions proposed to extract different types of information from CTI reports such as [17], [21] or [19]. One general challenge comes from the unstructured nature of these reports which often contain ambiguities that might be hard to interpret properly by the NLP system, leading to wrong parsing of the sentences, wrong POS tagging, or misidentified dependencies. These ambiguities might refer to the narrative style of the report (i.e., active or passive voice), but also to other noises such as special characters or words that are not common in natural language (e.g., IOCs). Another challenge is identifying and filtering relevant information for recovery steps. As an example, if the extracted information refers to a user executing a non-malicious activity such as copying a file, we need to distinguish it from the threat actions. The last challenge in this process is to link threat actions to the recovery steps which requires a deeper understanding of the threat behaviour. To sum up, the three challenges of inferring recovery steps from CTI reports are the following: *text ambiguity*, *information relevance*, and *recovery linking*.

3 Methodology & Implementation

To address the challenges described in Section 2.4 we take two approaches: semantic-based and generative. This section describes the methodology and implementation details of the two approaches.

3.1 Semantic approach

To address the *automation* challenge described in Section 2.4, we propose an automated system that infers recovery steps based on the description of an attack. To process the threat descriptions, first, we need to model the sentences,

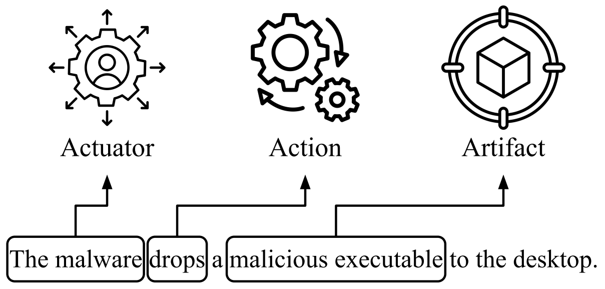

Fig. 1: **Example of the identified components by the semantic model.** When extracting the information from this sentence, we identify **The malware** as actuator, **drops** as action and **malicious executable** as artifact.

which gives a structured form that is more interpretable for the computers. One common approach used in similar works is identifying three main components: *action*, which is the main event described in the sentence, *actuator*, which is the entity performing the action, and *artifact*, which is the entity acted upon or affected by the action [30] (e.g., see Figure 1).

Our system has two main components at its core: the *extractor* and the *mapper*. The extractor is responsible for extracting information from a given text based on our sentence model described above. The mapper searches the extracted actions in a predefined list of threat actions that have a mapping to a recovery action. For the actions that are found in this mapping, the inferred recovery action is given to the operator. However, some actions are actually threat actions, such as **to steal** in the context of stealing personal information, but they do not have a recovery action that can be deployed on the system. Instead, these threat actions require making legal actions, change of policies or raise awareness of the employees. In this research, we consider non-technical recovery actions out of scope.

The mapping is a tool that can be customized based on the preferences and needs of every SOC. While this mapping becomes another resource that needs maintenance over time as the policies of the organization change, maintaining it does not imply a significant increase in the manual effort of the operators. Giving the flexibility to customize the mapping is needed, because if we take as an example the artifacts created by a threat on a system, some organizations might want to delete the artifact completely, others might put it just in a quarantine to be able to analyze it later.

To implement our system we create pipelines where each component of the pipeline has a clearly defined task. The main components of our pipeline are the following: *text preprocessor* for cleaning the text, the extractor for identifying the (actuator, action, artifact) triplets, and the mapper for linking the threat actions to recovery actions. The full pipeline (see Figure 2) is implemented in Python, using different libraries that we describe in the remainder of this section.

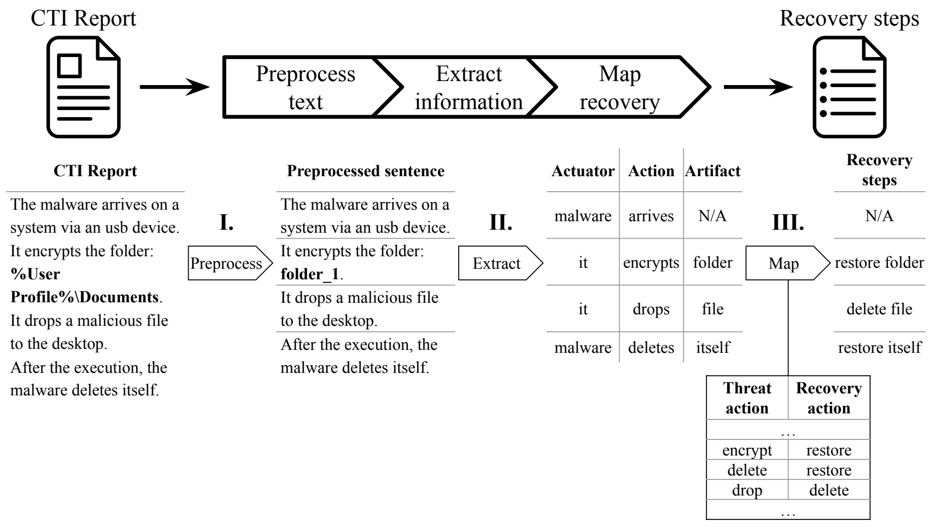
CTI Report
Preprocess text
Extract information
Map recovery
Recovery steps
CTI Report
The malware arrives on a system via an usb device. It encrypts the folder: %User Profile%\Documents. It drops a malicious file to the desktop. After the execution, the malware deletes itself.
I.
Preprocess
Preprocessed sentence
The malware arrives on a system via an usb device.
It encrypts the folder: folder_1.
It drops a malicious file to the desktop.
After the execution, the malware deletes itself.
II.
Extract
Actuator | Action | Artifact
malware | arrives | N/A
it | encrypts | folder
it | drops | file
malware | deletes | itself
III.
Map
Recovery steps
N/A
restore folder
delete file
restore itself
Threat action | Recovery action
…
encrypt | restore
delete | restore
drop | delete
…

Fig. 2: **Example of how the designed system works.** As input the system gets a CTI report in an unstructured format which in the I. step is preprocessed to mask IOCs and split the input text into sentences, then in the II. step the actuator, action and artifact triplets are extracted from the sentences, and in the III. step the recovery steps are inferred from the triplets based on a mapping.

Text preprocessing. As described in Section 2.4, CTI reports often contain ambiguous words (e.g., file paths, IP addresses, or hashes) that are unusual in the generic English text. To partly address the *text ambiguity* challenge, we use the IOCSearcher [9] tool to mask the IOCs.

In the reports, paragraphs are delimited by new lines, but we observe that in some cases (e.g., bullet point enumeration) some information positioned in new lines is still connected logically to a previous paragraph. Using simple heuristics, we split the text into logically connected parts to facilitate proper sentence parsing at the next step.

As the last preprocessing step, we use the SpaCy library [16], to tokenize and parse the sentences. The motivation of using SpaCy is that this library provides a large set of advanced natural language processing functions for extracting information from text. Also, this tool is commonly used in research that is connected to NLP. For our work, we use the built-in `en_core_web_trf` model that allows us to process English text.

Triplet extraction. This part of the pipeline is responsible of identifying the actuator, action, and artifact triplets, using the POS taggings and dependency tree got from processing the sentences. First, we search for the main action in a sentence. Then, using the dependency relations, we check for other connected actions to allow processing sentences, where instead of a single action we find a chain of actions (e.g., the malware **connects** to remote site and **exfiltrates** information). The last step is to iterate through the nouns and check for their dependency towards the tokens in the list of actions, which determines if the noun represents a subject (i.e., this is the actuator of the action), a direct object (i.e., this is the artifact of the action) or another dependency.

Recovery action mapping. Inferring recovery steps relies on a mapping between threat actions and recovery actions. The predefined list of threat actions has the role to address the *information relevance* challenge, while the mappings address the *recovery linking* problem described in Section 2.4. This mapping is stored in a JSON file that can be configured based on the needs of the user. The JSON file contains key-value pairs, where the keys represent the threat action while the values represent the corresponding recovery actions.

3.2 Generative approach

In the semantic approach, we used a pipeline, where individual components solve a smaller sub-task in inferring recovery actions. Conversely, generative models can be used to directly request the desired output, e.g., using their question-and-answer capabilities. In this setup, we need to engineer an input prompt that instructs the LLM to infer recovery steps. Some good practices of AI-experts in prompt engineering is to be explicit about the task specification and requirement and give examples of desired interactions [40]. Based on these practices, we design our prompt, that has three components: a system prompt that instructs the LLMs on how to behave, an example prompt showing the desired interaction, and the actual prompt for the task. For the system prompt, we first define the role the LLM should assume (i.e., incident responder in a SOC), then we give the instructions for inferring the recovery steps with the clarification of what to do if there is no recovery step that can be inferred, and last, we indicate the desired output format (JSON). For the example prompt we include a short CTI report and provide the expected, correct recovery steps in JSON format. For the actual prompt, we always include the CTI report from which we would like the LLM to infer the recovery steps. The final prompt template we have created is included in Appendix A.

The interaction with the LLMs is different for the open-source and closed-source LLMs. While the open-source LLMs are available to download and run on a local infrastructure, the closed-source LLMs offer only an API service for the interaction.

4 Evaluation

We have implemented a semantic-based model that infers recovery steps by analyzing CTI reports. To evaluate this model, we compare it against different off-the-shelf generative LLMs to gain a deeper understanding of what are the advantages and disadvantages of each model. We start by creating a labeled dataset of recovery steps, described in Section 4.1. Next, we perform our experiments presented in Section 4.2. Then Section 4.3 describes the metrics used to measure the performance of the approaches. Finally, we evaluate the results of the experiments in Section 4.4.

4.1 Dataset

To evaluate our results we use CTI reports collected from the Trend Micro Threat Encyclopedia [1]. These reports describe several technical aspects of the malware such as arrival details, autostart technique, routines, modification it does in the system, information it gathers for exfiltration or propagation.

The reports are written in natural language, thus we create a manually labeled subset of the original reports. The dataset, manually labeled by us, consists of 200 randomly selected reports. The manual labels of one report contain the following:

- manually parsed sentences from the report;
- manually extracted actuator, action, artifact triplets for each sentence; and
- manually mapped recovery action and artifact pairs for each triplet that has a recovery countermeasure.

In this manually labeled dataset, we have a total of 1,603 sentences of which 818 are unique. Amongst the unique sentences, we identified 829 actuator, action, artifact triplets, from which 580 have recovery steps based on our evaluation.

4.2 Experimental setup

For this research, we conduct an experiment of processing all the CTI reports from our labeled dataset presented in Section 4.1. This experiment is repeated on multiple models, that we detail in the remaining of this subsection.

Running the experiment on the semantic model does not require special hardware, hence we run them on regular workstation with the following characteristics: 11th Gen Intel Core i7-11800H processor running at 2.3GHz, 16GB of RAM, running Windows 11.

For the open-source LLM experiment, we choose the 13 billion parameter version of LLaMa 2. This requires a more powerful hardware than a regular workstation, so we run the experiments on a GPU cluster using an NVIDIA RTX 6000 GPU with 48 GB of memory.

For the closed-source LLM experiment we do not have the possibility to run the experiment locally. Instead, using the paid API service provided by OpenAI, we send our prompts to a specific version of GPT, the LLM remotely generates the answer and sends us back the result. We repeat the experiment on multiple versions of GPT, namely GPT 3.5 Turbo, GPT 4, and GPT 4 Turbo, which at the time of experiments point to the following models (in the same order): `gpt-3.5-turbo-1106`, `gpt-4-0613` and `gpt-4-0125-preview`.

Table 1: **Overall results of average metrics and the total number of factual errors made over the entire dataset.**

Model	**Avg. precision (%)**	**Avg. recall (%)**	**Factual errors (count)**
GPT 3.5 Turbo	65.11	76.42	22
GPT 4	52.81	**84.57**	24
GPT 4 Turbo	**78.02**	83.15	**16**
LLaMa	45.24	63.97	90
Semantic model	32.85	31.50	N/A

4.3 Metrics

The problem of inferring recovery steps does not fit the traditional definition of a classification task. While in a classification task the main task is to assign a label or category to a given input, in our case the emphasis is on identifying and representing entities and their relations within sentences. In NLP this is referred as a relation extraction task [7].

To evaluate this relation extraction task, we use the analogous metrics to those employed in classification tasks. These metrics are computed at sentence-level, considering the following:

- **True Positives (TP)**: the count of correctly inferred recovery steps based on the ground truth.
- **False Positives (FP)**: the count of incorrectly inferred recovery steps that are absent within the ground truth.
- **False Negatives (FN)**: the count of ground truth recovery steps that were not inferred by the system.

To evaluate the performance of the system at report-level, we aggregate the respective TP, FP, and FN counts across all sentences within a report and we compute the precision (P) and recall (R) by their classical definition.

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}$$

We also measure the runtime of each model on a report-level. Furthermore, in the experiments with the LLMs we include two additional measurements regarding factuality and output format of these models. LLMs tend to infer recovery steps cannot be mapped to any existing sentence in the input report. We call these *factual errors.* To evaluate the factuality, we compute on a report level the number of factual errors. LLMs were designed to provide natural-language-like output, however they can be instructed to provide a more specific format. To evaluate our models, we count on the dataset-level for how many input reports LLMs provide a JSON formatted output as they were instructed. Note, that for evaluating the performance and factuality of the models, we include only the outputs that provided a JSON format.

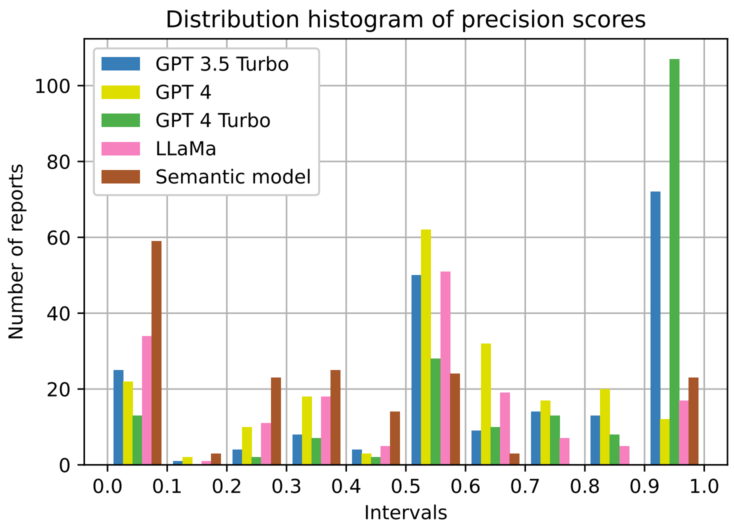

Fig. 3: **Distribution histogram of precision scores for the different models.** Vertical grid lines mark intervals, and the bars show the number of reports with precision values within each interval.

4.4 Results

Performance. Figure 3 presents the distribution histogram of precision scores obtained for each model, while Figure 4 shows the analogous histogram for recall scores. Additionally Table 1 presents the average precision and recall over the entire dataset. Note, that we removed the data points where the precision or recall were undefined (i.e., both TP and FP or FN counts were zero).

From the results, we can observe that overall the entire dataset, GPT 4 Turbo performs the best for precision, while GPT 4 outperforms it for recall. This means, that while GPT 4 Turbo produces fewer FPs (i.e., inferred steps that are not recovery steps), GPT 4 is better at inferring the correct recovery steps, leaving out fewer unidentified steps. Both in terms of precision and recall, the semantic model performs the worst compared to the LLMs.

Factuality. The semantic model has a limited set of possibilities to output depending on the actions included in the mapping. The output depends on the extracted information combined with this mapping, hence it is not possible to get an output that cannot be mapped to a word extracted from the input report.

One example would be the following factual error: **restore user folder**, which according to the LLM comes from the sentence **The executable encrypts the user folder.** At first glance, this recovery step suggested by the LLM makes sense with the provided explanation. However, upon closer investigation, we found that there was no mention of encrypting user folders in the

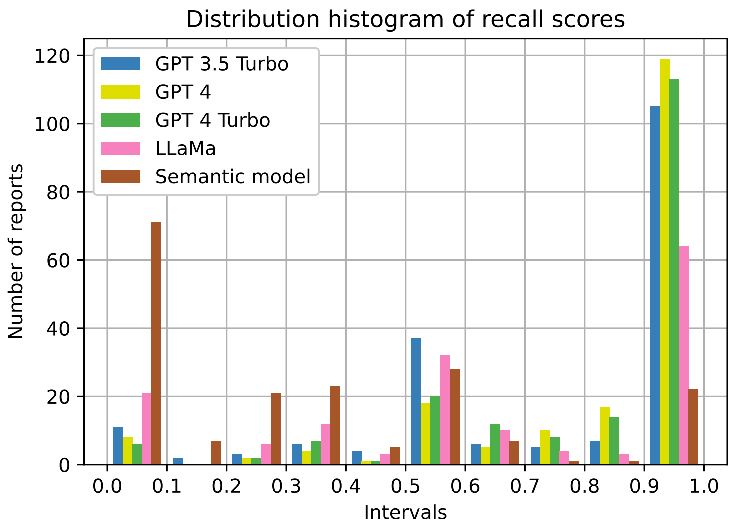

Fig. 4: **Distribution histogram of recall scores for the different models.** Vertical grid lines mark intervals, and the bars show the number of reports with recall values within each interval.

input CTI report in the first place. Table 1 shows the total number of factual errors in the dataset.

Output format. Our semantic model always provides JSON formatted output by design. The generative LLMs were instructed to provide the output in JSON format, however, only the GPT models were able to comply to this request formulated in the input prompt. The LLaMa model provided for 170 reports the output in a JSON format and for the rest 30 reports it provided only natural-language-like answers.

Runtime. Table 2 shows a comparison of the runtime to process a single report from our dataset. We note that the semantic model outperforms the LLMs in terms of runtime, the average runtime of this model being almost ten times faster than the LLMs. While on average both LLaMa and GPT models perform similarly, we note that the minimum and maximum runtime is significantly higher for GPT.

5 Discussion

This section of the paper discusses the results presented in Section 4.4. First, we compare the two approaches based on the results and discuss key takeaways. Then, we present the limitations and future directions of our work.

Table 2: **Runtime measurements of the different models.** The table shows the minimum, average, and maximum processing time per report for each model. *For GPT, the times shown represent the average runtime of the three models.

	Time (s)		
Model	**Min.**	**Avg.**	**Max.**
Semantic model	0.43	2.64	28.51
LLaMa	2.91	20.67	104.44
GPT*	7.33	19.61	612.66

5.1 Semantic vs. generative models

While in terms of performance, the results shown in Section 4.4 clearly show that LLMs outperform the semantic approach, there are other aspects we can compare the approaches on. In the following, we make a theoretical comparison between the semantic and generative models and we formulate the key takeaways of our work.

Factuality. Semantic approaches rely on different rules and heuristics, thus in this case we cannot speak about their factuality. On the other hand, our results show, that factuality is a problem of generative LLMs that tend to produce factual errors.

Output format. When implementing a semantic approach, the developer has always the choice to decide on the format of the output. Using LLMs, we have the option to provide the desired output format in the input prompt, but there is no guarantee that this will be strictly applied.

Predictability. The semantic model we implemented is deterministic, meaning that users can expect to always get the same answer for the same input. More than this, the mapping used to infer the recovery steps guarantees that for a certain threat action the inferred recovery action is always the same. Conversely, the LLMs tend to use multiple synonyms for the same recovery action (e.g., **delete**, **remove**, **clean**) instead of using one, deterministic form of the action, which might bring challenges in the further processing of the output.

Hardware requirements. While we are able to run the semantic model on a personal computer, LLaMa requires a system with high-performance GPU. The GPT models are not available to run locally, but given that they have significantly more parameters in their architecture than LLaMa we can safely assume that these models have the highest requirements from our experiments.

Takeaways. The most important takeaways of this work are the following:

– LLMs reach higher performance in terms of precision and recall than semantic approaches, even if they were not specifically trained to solve the specific problem at hand.
– Semantic approaches focus on factual accuracy and consistent reasoning, in contrast to LLMs which tend to produce unpredictable or factually incorrect answers.
– A hybrid approach, combining semantic NLP with LLMs, has a potential to surpass the limitations of both approaches.

5.2 Limitations

Our results show that both approaches have several limitations. While the semantic model has the focus on understanding the meaning of the sentences, it lacks a broader understanding of the context. One example of this limitation is the sentence **It deletes itself after execution.** In this example **delete** seems to be a threat action that has **restore** as recovery, but focusing on a broader context, **itself** refers to the malware, which means in this case the system should not infer any recovery steps.

On the other hand, LLMs are better at taking into account this contextual information, but have their own limitations in explainability. LLMs are to some extent able to provide explanations to which sentence of the input is the recovery step connected to, but occasionally these explanations are factual errors. This leads to the main limitation of the LLMs, namely the factualness of their output: not every answer coming from the LLMs is based on facts, instead the answers can be sometimes just creative choosing of seemingly correct words.

5.3 Future work

To address the limitations above, we propose as a future direction the combination of the two approaches. Leveraging the wide-range background knowledge of the LLMs together with the explainability of the semantic approach could allow for a high-performance system in which the decisions can be traced back to specific knowledge sources or rules. Such a system would avoid factual errors while having a consistent set of recovery actions to choose from.

Another possible direction for future work is to use the inferred recovery steps to generate actual playbooks tailored to a specific infrastructure that can be used in incident response. This would result in new challenges that need to be addressed such as ordering the recovery steps by their dependencies (e.g., first stop malicious process, then restore backup) and taking into consideration the infrastructure, the security tools, the policies of the organization.

6 Related work

We can categorize the related work into three main parts: *CTI*, *knowledge extraction*, and *incident response playbook* related works. These three are the key components that our work relies on. CTI is the key source of information for our system, knowledge extraction is the main process we use to analyze the behaviour of the threats and playbooks represent the direction towards which our system is headed by inferring recovery steps. In the following, we present some of the most important works on these three fields that are related to our research.

6.1 Cyber Threat Intelligence

In 2018 Abu et al. [3] present in their paper an important challenge that we still face: the overload of CTI data, which requires to automation solutions for operators not to be overwhelmed with their work. Tounsi et al. [36] in their survey highlight another important issue, namely the quality of the available threat intelligence, especially describing the technical details such as potential attack vectors and TTPs. Besides the overload and quality, another challenge identified by Wagner et al. [39] is focused on sharing the intelligence, which raises issues of trust establishment but also interoperability of the data. A more recent study is presented by Schlette et al. [30] on the usage of CTI, analyzing different formats and identifying core concepts required for incident response.

The previously mentioned works provide valuable perspectives on recent advancements in the CTI domain. They shed light on the technical characteristics of CTI reports and the mechanisms by which these reports are shared. This highlights the potential of using CTI reports as input data for designing a recovery step inferring system.

6.2 Knowledge extraction

Knowledge extraction is a common process in the field for gathering, analyzing, and leveraging information that helps us improve the defense of IT systems. There are lots of works proposing solutions for this processes. Two main categories of approaches are the traditional NLP techniques and data-driven models.

Traditional NLP techniques. These models focus on capturing the meaning of data beyond the meaning of the individual words by identifying different entities, relationships, and attributes. There are several works that build upon traditional NLP techniques to extract information such as IOCs, attack patterns, or temporal information, that are useful for security operators.

One of the most influential papers is TTPDrills [17], which presents an approach that combines NLP with information retrieval techniques to extract threat action. Park et al. [25] describes a full-stack information extraction system, SecIE, that extracts entities, relationships, and even temporal information with high accuracy. Another work, iACE [20], presents a technique based on semantic models to extract IOCs from blog posts and articles crawled from the internet. Some works focus on creating knowledge graphs from processing CTI reports, such as Extractor [28], that build a complex, detailed knowledge graph including the temporal order of the attack.

Data-driven models. We can distinguish two subcategories of data-driven models: supervised and unsupervised models.

First, we present some of the related works that are based on supervised models. Lim et al. [21] were the first ones to introduce an ML-based annotator that targets to label CTI reports by identifying entities, relations, and attributes, while classifying sentences based on their relevance (i.e., if they describe malware actions and capabilities or not). Similar to this, Manikandan et al. [22] presents a conditional random fields approach combined with a convolution neural network-based system to predict malware token labels and classify sentences (with the same criteria as [21]). Some other works combine ML with traditional NLP techniques, such as CASIEs [29], that combines deep learning with linguistic features to extract information about cybersecurity event.

In the following, we present the related works based on unsupervised models, where we accentuate the focus on the novel generative LLMs that are a special type of data-driven models trained on large corpora of data to learn producing human-like text. LLMs are not designed for solving domain-specific problems, instead they act as a very knowledgeable human (i.e., they produce natural-language-like answers, just as a human knowing the answer to the question would). One recent work that looks into extracting information from CTI reports using the question-and-answer capabilities of the LLMs is presented in the paper of Siracusano et al. [32] describing a system for extracting entities and attack patterns. Several other papers look into the usage of these models on different areas of the security domain such as writing reports [27], generating secure code [15], investigating software vulnerabilities [26], or creating honeypots [33], but all these works leverage the text generation ability rather than investigating the domain-specific question and answer capabilities of the LLMs.

6.3 Incident response playbooks

Since playbooks play an important role in the incident response process, researchers actively look into this topic. In 2018 MITRE introduced their RECAST [6] framework for efficiently choosing prioritized courses of action. Later, Onwubiko et al. [24] proposes SOTER, which is a general incident management playbook, and together with this, the authors create an incident management lexicon and propose an incident classification and prioritization scheme. A more recent study by Stevens et al. [34] looks into the process of creating playbooks using standard playbook design frameworks and identifies that these playbooks often lack sufficient information to be used in an actual attacks. Probably the most up to date and most comprehensive study on openly available playbooks is presented by Schellte et al [31]. In this study the authors categorize playbooks into community and organization-specific playbooks, while analyzing the key characteristics of community playbooks and identifying the influencing factors that play a role in creating organization-specific playbooks.

7 Conclusion

In this work we implemented a semantic approach to infer recovery steps from CTI reports, comparing it with off-the-shelf LLMs. To measure the performance of the approaches, we constructed a labeled dataset composed of malware CTI reports. Our experiments show that LLMs outperform the semantic approach by 45% in precision and 53% in recall. However, we also found that the semantic approach is more consistent, inferring fewer incorrect recovery steps than LLMs.

Acknowledgments

We would like to thank our reviewers for their valuable comments. This work has been supported by the SeReNity project, Grant No. cs.010, funded by Netherlands Organisation for Scientific Research (NWO). The findings reported herein are the sole responsibility of the authors.

References

1. Threat Encyclopedia — trendmicro.com. `https://www.trendmicro.com/vinfo/us/threat-encyclopedia/`
2. 2023 State of Threat Detection. `https://www.vectra.ai/resources/research-reports/2023-state-of-threat-detection` (2023)
3. Abu, M.S., Selamat, S.R., Ariffin, A., Yusof, R.: Cyber Threat Intelligence–Issue and Challenges. Indonesian Journal of Electrical Engineering and Computer Science **10**(1), 371–379 (2018)
4. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: GPT-4 Technical Report. arXiv preprint arXiv:2303.08774 (2023)
5. Aminanto, M.E., Ban, T., Isawa, R., Takahashi, T., Inoue, D.: Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis. IEEE Access **8**, 217977–217986 (2020)
6. Applebaum, A., Johnson, S., Limiero, M., Smith, M.: Playbook Oriented Cyber Response. In: National Cyber Summit, NCS. IEEE (2018)
7. Bach, N., Badaskar, S.: A Review of Relation Extraction. Literature review for Language and Statistics II **2**, 1–15 (2007)
8. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language Models are Few-Shot Learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
9. Caballero, J., Gomez, G., Matic, S., Sánchez, G., Sebastián, S., Villacañas, A.: The Rise of GoodFATR: A Novel Accuracy Comparison Methodology for Indicator Extraction Tools. Future Generation Computer Systems **144**, 74–89 (2023)
10. Chyssler, T., Burschka, S., Semling, M., Lingvall, T., Burbeck, K.: Alarm Reduction and Correlation in Intrusion Detection Systems. In: Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA). LNI, GI (2004)

11. van Ede, T., Aghakhani, H., Spahn, N., Bortolameotti, R., Cova, M., Continella, A., van Steen, M., Peter, A., Kruegel, C., Vigna, G.: DeepCASE: Semi-Supervised Contextual Analysis of Security Events. In: Proceedings of the IEEE Symposium on Security and Privacy (S&P) (2022)
12. Fleck, A.: Cybercrime Expected To Skyrocket in Coming Years. `https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/` (2022)
13. Ghribi, S., Makhlouf, A.M., Zarai, F., Guizani, M.: Fog-cloud distributed intrusion detection and cooperation. Transactions on Emerging Telecommunications Technologies **33**(8), e3835 (2022)
14. Hassan, W.U., Guo, S., Li, D., Chen, Z., Jee, K., Li, Z., Bates, A.: NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. In: Proceedings of the Symposium on Network and Distributed System Security (NDSS) (2019)
15. He, J., Vechev, M.T.: Large Language Models for Code: Security Hardening and Adversarial Testing. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2023)
16. Honnibal, M., Montani, I.: spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing (2017), to appear
17. Husari, G., Al-Shaer, E., Ahmed, M., Chu, B., Niu, X.: TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources. In: Proceedings of the 33rd annual computer security applications conference (2017)
18. Jakkal, V.: Introducing Microsoft Security Copilot: Empowering defenders at the speed of AI. `https://blogs.microsoft.com/blog/2023/03/28/introducing-microsoft-security-copilot-empowering-defenders-at-the-speed-of-ai/` (2023)
19. Legoy, V., Caselli, M., Seifert, C., Peter, A.: Automated Retrieval of ATT&CK Tactics and Techniques for Cyber Threat Reports. arXiv preprint arXiv:2004.14322 (2020)
20. Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., Beyah, R.: Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2016)
21. Lim, S.K., Muis, A.O., Lu, W., Ong, C.H.: MalwareTextDB: A Database for Annotated Malware Articles. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2017)
22. Manikandan, R., Madgula, K., Saha, S.: TeamDL at SemEval-2018 task 8: Cybersecurity text analysis using convolutional neural network and conditional random fields. In: Proceedings of The 12th International Workshop on Semantic Evaluation (2018)
23. Murphy, S.: Your New AI Assistant: Trend Vision One™ – Companion. `https://www.trendmicro.com/en_my/research/23/f/companion-ai-assistant-trend-vision-one.html` (2023)
24. Onwubiko, C., Ouazzane, K.: SOTER: A playbook for cybersecurity incident management. IEEE Trans. Engineering Management **69**(6), 3771–3791 (2022)
25. Park, Y., Lee, T.: Full-Stack Information Extraction System for Cybersecurity Intelligence. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing: Industry Track (2022)
26. Pearce, H., Tan, B., Ahmad, B., Karri, R., Dolan-Gavitt, B.: Examining Zero-Shot Vulnerability Repair with Large Language Models. In: Proceedings of the IEEE Symposium on Security and Privacy (S&P) (2023)

27. Ranade, P., Piplai, A., Mittal, S., Joshi, A., Finin, T.: Generating Fake Cyber Threat Intelligence Using Transformer-Based Models. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN (2021)
28. Satvat, K., Gjomemo, R., Venkatakrishnan, V.: Extractor: Extracting Attack Behavior from Threat Reports. In: Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P) (2021)
29. Satyapanich, T., Ferraro, F., Finin, T.: CASIE: Extracting Cybersecurity Event Information from Text. In: Proceedings of the AAAI conference on artificial intelligence (2020)
30. Schlette, D., Caselli, M., Pernul, G.: A Comparative Study on Cyber Threat Intelligence: The Security Incident Response Perspective. IEEE Communications Surveys Tutorials **23**(4), 2525–2556 (2021)
31. Schlette, D., Empl, P., Caselli, M., Schreck, T., Pernul, G.: Do You Play It by the Books? A Study on Incident Response Playbooks and Influencing Factors. In: Proceedings of the IEEE Symposium on Security and Privacy (S&P). IEEE Computer Society (2024)
32. Siracusano, G., Sanvito, D., Gonzalez, R., Srinivasan, M., Kamatchi, S., Takahashi, W., Kawakita, M., Kakumaru, T., Bifulco, R.: Time for aCTIon: Automated Analysis of Cyber Threat Intelligence in the Wild. CoRR **abs/2307.10214** (2023)
33. Sladić, M., Valeros, V., Catania, C., Garcia, S.: LLM in the Shell: Generative Honeypots. arXiv preprint arXiv:2309.00155 (2023)
34. Stevens, R., Votipka, D., Dykstra, J., Tomlinson, F., Quartararo, E., Ahern, C., Mazurek, M.L.: How Ready is Your Ready? Assessing the Usability of Incident Response Playbook Frameworks. In: Proceedings of the CHI Conference on Human Factors in Computing Systems (2022)
35. Sundaramurthy, S.C., Bardas, A.G., Case, J., Ou, X., Wesch, M., McHugh, J., Rajagopalan, S.R.: A Human Capital Model for Mitigating Security Analyst Burnout. In: Proceedings of the Eleventh Symposium On Usable Privacy and Security (SOUPS) (2015)
36. Tounsi, W., Rais, H.: A survey on technical threat intelligence in the age of sophisticated cyber attacks. Comput. Secur. **72**, 212–233 (2018)
37. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288 (2023)
38. Vermeer, M., Kadenko, N., van Eeten, M., Gañán, C., Parkin, S.: Alert Alchemy: SOC Workflows and Decisions in the Management of NIDS Rules. In: "Proceedings of the ACM Conference on Computer and Communications Security (CCS)" (2023)
39. Wagner, T.D., Mahbub, K., Palomar, E., Abdallah, A.E.: Cyber threat intelligence sharing: Survey and research directions. Comput. Secur. **87** (2019)
40. Zamfirescu-Pereira, J., Wong, R.Y., Hartmann, B., Yang, Q.: Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In: Proceedings of the CHI Conference on Human Factors in Computing Systems. pp. 1–21 (2023)

A Prompt used for the LLM experiments

```
<s>[INST] <<SYS>>
You are an Incident Responder from a Security Operations Center.
Your task is to execute the following instructions:
1. analyze the given report;
2. extract the recovery actions to be deployed under the described
attack;
3. identify the object on which the recovery action should be
executed.
If there is no recovery action for a certain sentence, discard it
and don't include it in the output.

Output the answer in JSON using the following format:
[
  {
    "sentence": sentence,
    "recovery_action": recovery_action,
    "object": object
  }
]
<</SYS>>

The report of the attack is the following:

The malware adds a malicious executable. The executable encrypts
the user folder. This attack has high risk. [/INST]
[
  {
    "sentence": "The malware adds a malicious executable.",
    "recovery_action": "delete",
    "object": "executable"
  },
  {
    "sentence": "The executable encrypts the user folder.",
    "recovery_action": "restore",
    "object": "user folder"
  }
]
</s>

<s>[INST]
The report of the attack is the following: {content}
[/INST]
```