LLM-TIKG: Threat Intelligence Knowledge Graph Construction Utilizing Large Language Model

Yuelin Hu[a], Futai Zou[a,*], Jiajia Han[b], Xin Sun[b], Yilei Wang[b]

[a]*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China*
[b]*State Grid Zhejiang Electric Power Co..Ltd Research Institute, Hangzhou, Zhejiang, China*

Abstract

Open-source threat intelligence is often unstructured and cannot be directly applied to the next detection and defense. By constructing a knowledge graph through open-source threat intelligence, we can better apply this information to intrusion detection. However, the current methods for constructing knowledge graphs face limitations due to the domain-specific attributes of entities and the analysis of lengthy texts, and they require large amounts of labeled data. Furthermore, there is a lack of authoritative open-source annotated threat intelligence datasets, which require significant manual effort. Moreover, it is noteworthy that current research often neglects the textual descriptions of attack behaviors, resulting in the loss of vital information to understand intricate cyber threats. To address these issues, we propose LLM-TIKG that applies the large language model to construct a knowledge graph from unstructured open-source threat intelligence. The few-shot learning capability of GPT is leveraged to achieve data annotation and augmentation, thereby creating the datasets for fine-tuning a smaller language model (7B). Using the fine-tuned model, we perform topic classification on the collected reports, extract entities and relationships, and extract TTPs from the attack description. This process results in the construction of a threat intelligence knowledge graph, enabling automated and universal analysis of textualized threat intelligence. The experimental results demonstrate improved performance in both named entity recognition and TTP classification, achieving the precision of 87.88% and 96.53%, respectively.

Keywords: Threat intelligence, large language model, knowledge graph, TTP classification

1. Introduction

The rapid development of zero-day attacks, advanced persistent threats (APTs), and other novel attack methods has elevated the complexity of cyberspace attacks to an unprecedented level. In response, the cybersecurity defense approach of "active defense, traceability, and countermeasures" is gradually gaining recognition[1], which relies on the collection and utilization of a large amount of threat intelligence. Cyber Threat Intelligence (CTI) is a type of information about cyber threats and attacks that is characterized by high accuracy and strong correlation. CTI can provide powerful data support for all stages of security analysis and help shift cyber security behavior from reactive to proactive. As such, CTI plays an immeasurable role in cyber security [2, 3].

In addition to commercially available threat information disseminated through security vendors, an increasing number of Open Source Threat Intelligence (OSCTI) resources have emerged as crucial components in the field of threat intelligence research [4, 5, 6, 7]. The nature of OSCTI often presents challenges due to its article-type content and unstructured format, rendering it unsuitable for direct application in subsequent detection and defense.

Traditional OSCTI collection methods [8, 9] concentrate on Indicators of Compromise (IoCs) such as hashes, IP addresses, and domains. However, attackers can easily change the characteristics of an attack, rendering them no longer matching the previously defined IoCs. Moreover, these approaches yield low-level, fragmented threat intelligence, which lacks high-level information (e.g., attackers, tools) and the relationships between intelligence entities that are essential for threat hunting and attack attribution [10, 11]. Discrete indicators make it challenging for analysts to comprehend the complete picture of the attack and give attackers the opportunity to evade detection.

Constructing a threat intelligence knowledge graph can help alleviate these issues and holds significant value. It integrates intelligence from multiple sources and graphically depicts threat relationships, allowing analysts to conduct more in-depth analyses through threat entities and their relationships. This aids in tracing the attack chain, deducing attacker intentions, and enhancing capabilities for responding to network security threats. Current research is beginning to focus on the construction of threat intelligence knowledge graphs. They employ Named Entity Recognition (NER) and Relationship Extraction (RE) from the field of Natural Language Processing (NLP) to analyze textualized threat intelligence [12, 13, 14], using various NLP techniques to extract threat entities and relations. These methods facilitate the extraction of a broader range of threat intelligence entities and the mining of relationships between entities.

*Futai Zou
Email addresses: `huyuelin@sjtu.edu.cn` (Yuelin Hu), `zoufutai@sjtu.edu.cn` (Futai Zou), `36155384@qq.com` (Jiajia Han), `16526452@qq.com` (Xin Sun), `wangyileichn@163.com` (Yilei Wang)

Despite their advance in constructing the threat intelligence knowledge graph , current approaches also suffer from the following limitations:

Accuracy of information extraction: Threat intelligence entities exhibit domain-specific characteristics, such as ambiguous entity boundaries and polysemous expressions, which complicate the named entity recognition [15]. The present approach also faces limitations related to the maximum sequence length [16], resulting in decreased effectiveness for long text. These two factors affect the accuracy of information extraction.

Demand for labeled data: Current approaches heavily depend on an extensive corpus of annotated data [17]. Moreover, there is a scarcity of authoritative open-source threat intelligence annotation datasets, and data annotation for model training necessitates significant labor investment.

Absence of attack behaviors: Current research ignores descriptions of attack behaviors in textual information [12, 14], such as "inject shellcode into file", from which it is challenging to extract entities. The Tactics, Techniques, and Procedures (TTP) framework proposed by MITRE ATT&CK [18] offers a summary of attack methods, and we can map such statements to TTPs. Incorporating these techniques into the knowledge graph and correlating them with other information can yield valuable insights into the tools and attack patterns employed by adversaries.

The emergence of Large Language Models (LLMs) provides a solution to the above problem by strong generalization capabilities across various downstream tasks [19]. Instruction tuning can further guide LLMs to more effectively address these tasks using natural language instructions. In this paper, we innovatively apply LLMs to the analysis of textualized threat intelligence, constructing a knowledge graph from unstructured open-source threat intelligence. We leverage GPT's few-shot learning capability by designing appropriate prompts for data annotation and data enhancement to construct the dataset required for model training. This avoids substantial manual effort. Subsequently, the Llama2-7B model is fine-tuned using the LoRA-based instruction tuning to accomplish topic classification, entity and relationship extraction, and TTP classification on the collected threat intelligence text. Finally, the threat intelligence knowledge graph is constructed, achieving automatic and universal analysis of textualized threat intelligence. The contributions of this paper are as follows:

1) We propose a method, LLM-TIKG, that leverages a large language model to analyze textualized threat intelligence. This method automates the extraction of entities and relationships in open-source threat intelligence and maps natural language descriptions of attack behaviors into TTPs, which facilitates the efficient construction of a knowledge graph that correlates low-level and high-level threat intelligence.

2) The few-shot learning capability of the GPT model is used for data annotation and data augmentation, assisting the creation of datasets for model fine-tuning while significantly diminishing the need for costly manual effort.

3) We manually correct a portion of the labeled dataset generated by GPT which used for the knowledge graph construction, and this dataset can be found at https://github.com/Netsec-SJTU/LLM-TIKG-dataset.

4) The results reveal enhanced performance in named entity recognition and TTP classification, achieving the precision of 87.88% and 96.53%, respectively.

2. Background

This section presents a practical example to illustrate the motivation behind the method proposed in this paper and introduces the large language model and its application in this context.

2.1. Motivation Example

Fig. 1 gives an example for knowledge graph construction from unstructured text. The text in the upper figure is a segment of a threat intelligence report encompassing a wealth of threat intelligence entities and relationships. After the initial intrusion, the attacker conducted a series of attack activities. The malicious DLL file was side-loaded using imjpuex.exe, which then loaded a .dat file, and imjpuex.exe started network services through svchost.exe. The hash value of imjpuex.exe is also provided. Most current research focuses on extracting standardized threat intelligence. However, isolated threat intelligence entities often lack the relationships between different entities and attack behaviors. Moverover, they neglect to extract the TTPs from the attack description, which is a higher level of threat intelligence.

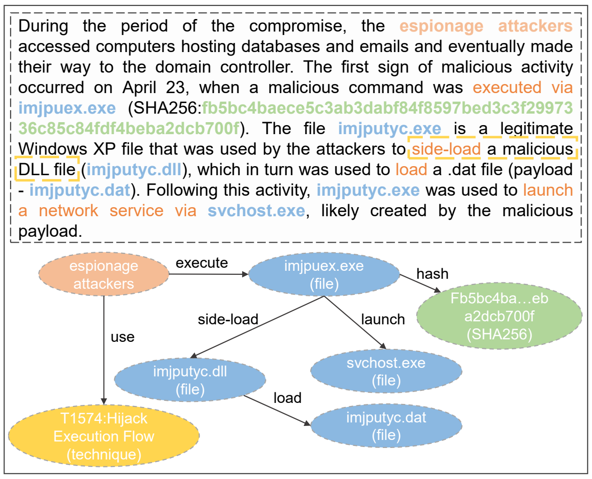
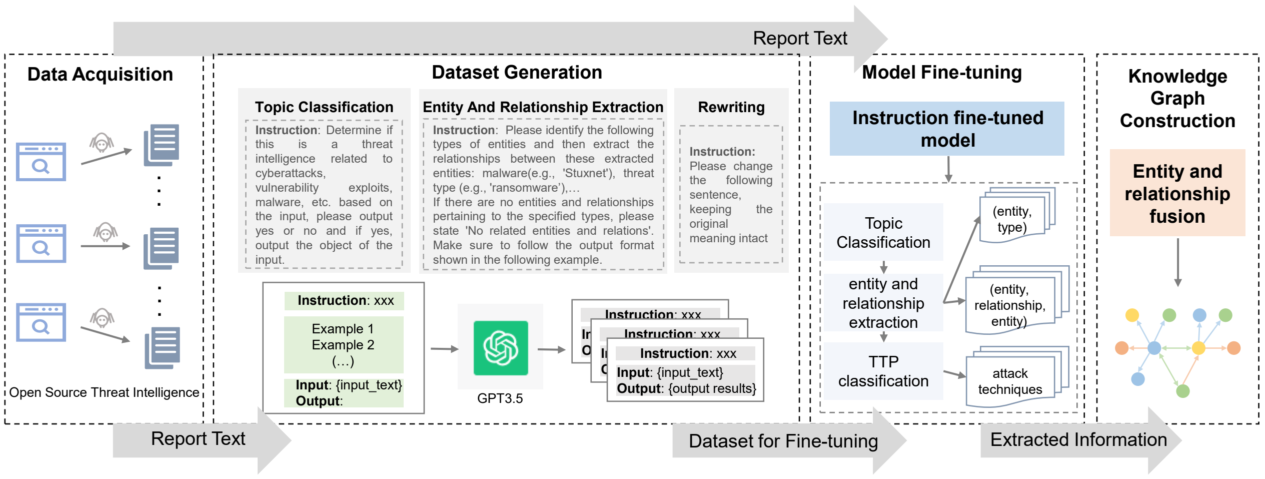

During the period of the compromise, the espionage attackers accessed computers hosting databases and emails and eventually made their way to the domain controller. The first sign of malicious activity occurred on April 23, when a malicious command was executed via imjpuex.exe (SHA256:fb5bc4baece5c3ab3dabf84f8597bed3c3f29973 36c85c84fdf4beba2dcb700f). The file imjputyc.exe is a legitimate Windows XP file that was used by the attackers to side-load a malicious DLL file (imjputyc.dll), which in turn was used to load a .dat file (payload - imjputyc.dat). Following this activity, imjputyc.exe was used to launch a network service via svchost.exe, likely created by the malicious payload.

Figure 1: An example of motivatio for espionage attackers's intrusion. The upper figure represents the text from a threat intelligence report, while the lower figure depicts the graph extracted from the text.

We expect to extract diverse threat intelligence entities and their relationships from this text, ranging from low-level IoCs to high-level details like attacker and TTP, as shown in the lower part of Fig. 1. The invocation relationships between malicious files in the graph imply the attack path and the text also includes the description of the attacker's behavior, such as "side-load a malicious DLL file" indicating that the attacker used the technique of Hijack Execution Flow (T1574). These attacker behaviors are the results of thorough analysis by security analysts and represent more advanced threat intelligence. If these pieces



Instruction: Determine if this is a threat intelligence related to cyberattacks, vulnerability exploits, malware, etc. based on the input, please output yes or no and if yes, output the object of the input.

Instruction: Please identify the following types of entities and then extract the relationships between these extracted entities: malware(e.g., 'Stuxnet'), threat type (e.g., 'ransomware'),…
If there are no entities and relationships pertaining to the specified types, please state 'No related entities and relations'. Make sure to follow the output format shown in the following example.

Instruction: Please change the following sentence, keeping the original meaning intact

Figure 2: The framework for LLM-TIKG, which generates labeled datasets using GPT and utilizes the fine-tuned model to construct a threat intelligence knowledge graph.

of information are missed in threat intelligence analysis, only some surface-level information is retained, and the full value of these analytical findings cannot be effectively utilized.

Constructing a knowledge graph with multiple levels of information, especially with the inclusion of TTP, fosters a better understanding of the attack's intent and the attribution of the attacker.

2.2. *Large Language Model*

Large Language Models (LLMs) [19] refer to Transformer language models with hundreds of billions or more parameters, such as GPT-3 [20], PaLM [21], and Llama [22]. Among them, ChatGPT [23] is currently the most popular model, which is trained to follow an instruction in a prompt and presents an amazing conversation ability with humans. And Llama2 [24] is a superior open-source model series ranging in scale from 7 billion to 70 billion parameters.

These models have emerged as powerful tools in the field of natural language processing. Researches have indicated that expanding pre-trained language models, either in terms of model parameter size or training data size, typically enhances the model's capacity for downstream tasks. This surprising capabilities (referred to as emergent abilities) can be used for a wide range of complex tasks [19]. These abilities are primarily manifested in three aspects: in-context learning, instruction following, and step-by-step reasoning.

In-context learning refers to the ability of a model to generate expected outputs according to task requirements when provided with a natural language instruction and a few task demonstrations, without the need for additional training or gradient updates. This capability enables the model to tackle few-shot or zero-shot tasks. Instruction following involves fine-tuning the model parameters using a task dataset described in natural language, also known as instruction tuning [25], which allows the model to achieve better generalization capabilities on unseen tasks. Step-by-step reasoning employs the chain-of-thought (CoT) prompting strategy [26], enabling the model to arrive at the final answer by utilizing prompts containing intermediate reasoning steps.

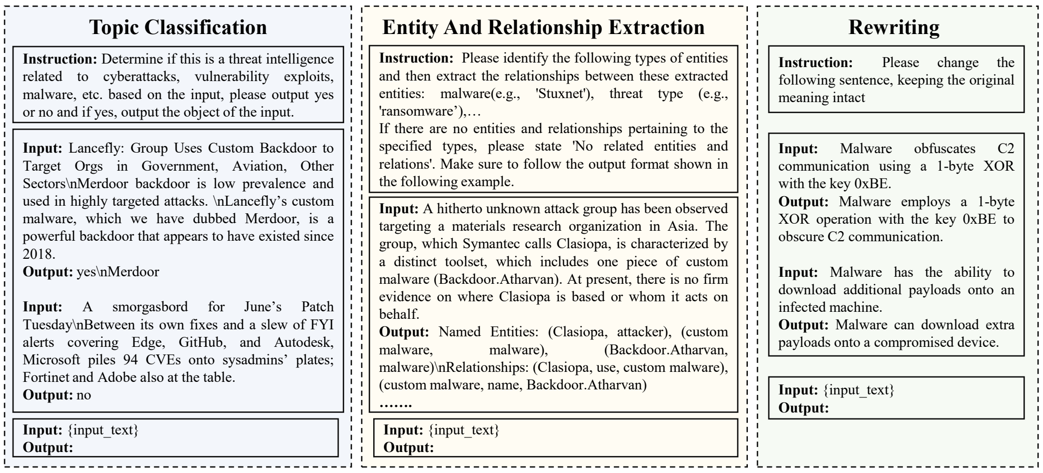
In this paper, we leverage GPT's few-shot learning capability by constructing instruction descriptions and task demonstrations corresponding to specific tasks. We utilize the out-of-the-box GPT-3.5 model for data annotation and data enhancement through a question-and-answer interaction, thereby obtaining the dataset required for the training and significantly reducing manual effort. Furthermore, we employ instruction tuning on the Llama2-7B model (considered a smaller language model compared to ChatGPT and Llama2-70B in terms of parameter size) to accomplish constructing a threat intelligence knowledge graph. Running the model locally reduces the risk of data leakage and enables the model to better adapt to the specific use case.

3. Proposed Method

This section provides a detailed description of the threat intelligence knowledge graph construction method proposed in this paper, LLM-TIKG, including an overview of the method and a description of each module, that is, data construction, model fine-tuning, and knowledge graph construction.

3.1. *Overview*

We propose an automated threat intelligence analysis method, LLM-TIKG, whose framework is shown in Fig. 2. It takes textual open-source threat intelligence as input and constructs a threat intelligence knowledge graph through the analysis of text content by large language models. Firstly, leveraging the powerful few-shot learning capability of the GPT model, we design task instructions and output examples for data annotation and data enhancement to obtain the dataset required for this method. The Llama2-7B model is then fine-tuned using the LoRA-based instruction tuning approach [27], enabling it to classify input texts, extract threat intelligence entities and their relationships, as well as extract TTP from attack descriptions.


Topic Classification

Instruction: Determine if this is a threat intelligence related to cyberattacks, vulnerability exploits, malware, etc. based on the input, please output yes or no and if yes, output the object of the input.

Input: Lancefly: Group Uses Custom Backdoor to Target Orgs in Government, Aviation, Other Sectors\nMerdoor backdoor is low prevalence and used in highly targeted attacks. \nLancefly's custom malware, which we have dubbed Merdoor, is a powerful backdoor that appears to have existed since 2018.
Output: yes\nMerdoor

Input: A smorgasbord for June's Patch Tuesday\nBetween its own fixes and a slew of FYI alerts covering Edge, GitHub, and Autodesk, Microsoft piles 94 CVEs onto sysadmins' plates; Fortinet and Adobe also at the table.
Output: no

Input: {input_text}
Output:

(a)

Entity And Relationship Extraction

Instruction: Please identify the following types of entities and then extract the relationships between these extracted entities: malware(e.g., 'Stuxnet'), threat type (e.g., 'ransomware'),…
If there are no entities and relationships pertaining to the specified types, please state 'No related entities and relations'. Make sure to follow the output format shown in the following example.

Input: A hitherto unknown attack group has been observed targeting a materials research organization in Asia. The group, which Symantec calls Clasiopa, is characterized by a distinct toolset, which includes one piece of custom malware (Backdoor.Atharvan). At present, there is no firm evidence on where Clasiopa is based or whom it acts on behalf.
Output: Named Entities: (Clasiopa, attacker), (custom malware, malware), (Backdoor.Atharvan, malware)\nRelationships: (Clasiopa, use, custom malware), (custom malware, name, Backdoor.Atharvan)
…….

Input: {input_text}
Output:

(b)

Rewriting

Instruction: Please change the following sentence, keeping the original meaning intact

Input: Malware obfuscates C2 communication using a 1-byte XOR with the key 0xBE.
Output: Malware employs a 1-byte XOR operation with the key 0xBE to obscure C2 communication.

Input: Malware has the ability to download additional payloads onto an infected machine.
Output: Malware can download extra payloads onto a compromised device.

Input: {input_text}
Output:

(c)

Figure 3: The prompt for different tasks with instruction, examples and input. The figures a, b, and c respectively represent prompts for three tasks: Topic Classification, Entity And Relationship Extraction and Rewriting.

Based on the extracted information, we integrate entities and relationships to construct a threat intelligence knowledge graph that connects low-level and high-level intelligence.

3.2. Dataset Construction

3.2.1. Data Acquisition

The purpose of this paper is to construct a threat intelligence knowledge graph through the analysis of open-source threat intelligence. To achieve this, we first collect a large amount of content published on various open-source threat intelligence platforms, including security company content platforms [4, 5, 6, 7], security news [28, 29], and influential personal security blogs [30]. These online resources provide numerous open-source threat intelligence reports, detailing various malware, vulnerabilities, threat actors and attack activities. They offer authoritative and accurate analyses of security incidents, making them rich and valuable sources of threat intelligence.

To collect the data, we design web crawlers tailored to the layout structure of each platform's website. After gathering the URLs of individual reports within these websites, we crawl the content of each report, extracting the text and removing extraneous information such as advertisements and sidebars. These reports often use subheadings to divide different sections, with each section describing content in different directions. We preserve the paragraph structure of each section based on the website's layout, removing unnecessary blank lines and using blank lines to separate different sections, which is beneficial for subsequent processing and analysis. The crawled reports are stored in the format of "title + link + content," serving as the data source for the threat intelligence analysis in this study.

3.2.2. Dataset Generation

Subsequently, we leverage the few-shot learning capability of GPT to generate the dataset for model fine-tuning. It requires setting instructions and demonstrations for different tasks, combining them with the input text, and feeding them to the model. The model then produces answers in the target format, which serve as the "output" in the fine-tuning dataset. We choose the "GPT-3.5-turbo" model and use the "instruction + examples + input" template shown in Fig. 3, where the instruction describes the generation task, examples are pairs of inputs and outputs that explain the tasks in the instructions and represent the desired output structure, and the input represents the statement to be processed. We now discuss the specific dataset construction methods for the three tasks: topic classification, entity and relationship extraction, and TTP classification.

Topic Classification: In open-source threat intelligence platforms, the information available is not solely limited to descriptions of malware and attack activities. It also encompasses product advertisements, security knowledge dissemination, irrelevant news, and other content unrelated to threat intelligence extraction. Analyzing such reports is not only resource-intensive but also potentially detrimental to the quality of the extracted threat intelligence information. Consequently, LLM-TIKG initially bicategorizes the reports based on their titles and first paragraph content to determine whether the report topic pertains to malware, security vulnerabilities, or attack activities. If so, the method further outputs the main object of this report, which is subsequently utilized in the knowledge graph construction.

To achieve this, we construct the prompt shown in Fig. 3(a). Additionally, due to the input length constraints of the large language model, we impose a limit on the input length, truncating the first paragraph of the report if it exceeds 150 words.

Entity and Relationship Extraction: The objective of this step is to identify and extract structured information from unstructured data sources, based on the target structure obtained from the extracted content. Traditional approaches typically involve employing named entity recognition methods to extract specific types of entity words from the text, followed by rule-based or syntactic analysis to extract inter-entity relationships. Leveraging the text comprehension and generation capabilities of large language models, we adopt a novel information extraction method, Instruction-based Information Extraction [30], which aims to require the LLM to adhere to specific instructions or guidelines for information extraction.

Consequently, we design the prompt shown in Fig. 3(b) in accordance with this objective, enabling the model to extract specified entity words from the input text and discern the relationships between entities. In this paper, we define the entity categories as illustrated in Table 1.

Table 1: Extracted entity categories and examples

Entity type	**Example**
Malware	Stuxnet, Truebot
Threat Type	Ransomware, APT
Attacker	Shuckworm, APT15
Technique	TA0004:Privilege Escalation, T1057: Process Discovery
Tool	PowerShell script, EHole
Vulnerability	CVE-2020-1472, Scripting Engine Memory Corruption Vulnerability (CVE-2020-0832)
IP	45.153.243.93, 92.242.62.131
Domain	personnel[.]bdm-sa[.]fr, xsph[.]ru
URL	hxxp://178.73.192[.]15/ca1.exe, https://myvaccinerecord.cdph.ca.gov/creds
File	rtk.lnk, shtasks.exe 2aee8bb2a953124803bc42e5c42935c9 (MD5)
Hash	91d42a959c5e4523714cc589b426fa83a aeb9228 (SHA1) c62dd5b6036619ced5de3a340c1bb2c9d 9564bc5c48e25496466a36ecd00db30 (SHA256)

We discover that when only specifying the entity types to be extracted in the instruction, GPT-3.5's responses may exhibit a certain degree of "LLM hallucination" phenomenon, extracting entity words that do not match the specified types. To mitigate this issue, we add examples corresponding to each type in the prompt, enabling the model to better understand the generation task. Since GPT-3.5 is not a specialized security domain model, there remains some bias in entity extraction, such as identifying security organizations like "Symantec" and "Check Point" as attackers, or extracting words containing IP or Domain as entities of this category. We design filtering rules to further refine these results and subsequently filter the extracted relationships based on this output. For the extracted relationships in the responses, we only keep the relationships between two entities existed in the list of extracted entities.

Moreover, GPT-3.5 is a generative model rather than a classification model. The model's output extracts entities from other categories in addition to the predefined ones. After analyzing the results, we retain entities of the "attack" and "device" categories in the dataset, filtering out unrelated entity types. Additionally, we manually correct a portion of the results to further enhance the quality of the extracted entities and relationships.

TTP Classification: Since the annotation of TTP classification requires sufficient knowledge of network security and TTP, the accuracy of TTP classification for attack descriptions directly using the GPT model is not very high. Therefore, we utilize a dataset derived from two sources of manually labeled data for this task. One part of the dataset uses the TTP classification of malware examples from MITRE ATT&CK [18] as the original data source, taking the description of the malware attack behavior as input and its corresponding Technique as output. The other part originates from the labeled dataset in Tram [31], which labels the technique presented in each sentence and includes sentences where the corresponding Technique is absent.

Due to the high similarity of the styles describing attacks in the dataset, and the original large language model's limited capacity to comprehend this aspect, we perform data augmentation on the collected data to enhance the number and diversity of the dataset. Data enhancement in natural language processing needs to preserve the correctness and original semantics of the sentences. Since specialized vocabulary in the security domain is involved, simple substitution of near-synonyms is not desirable. We employ two methods to extend the dataset, with the data being in English. One involves translating the original statements into another language and then translating them back into English, and the other method consists of obtaining alternative expressions of the input sentences by inputting the instruction "please rewrite the following sentence, keeping the original semantics unchanged" to the GPT model, which is shown in Fig. 3(c). In practice, this task only requires the input of an instruction and not the output examples.

3.3. Model Fine-tuning

In order to create an LLM tailored for the specific task, we utilize the aforementioned collected dataset to perform instruction tuning on Meta's Llama2-7B model[24], a publicly accessible LLM. Instruction tuning is a technique employed for the fine-tuning of pre-trained large language models using a curated collection of natural language instances. This approach enables these LLMs to refine their performance with respect to specific goals using their capacity to generalize effectively to unseen tasks. Since LLMs consist of a vast number of model parameters, performing full parameter optimization is costly. This issue can be addressed by training only a small subset of parameters, which may either be a subset of existing model parameters or a newly added set of parameters. This approach mitigates the resource-intensive nature of traditional fine-tuning techniques.

Here, we adopt Low-Rank Adaptation (LoRA)[27] for fine-tuning the model. Consider the case of optimizing the parameter matrix W, the update process can be written in a general form as: $W \leftarrow W_0 + \Delta W$. The fundamental idea of LoRA is to freeze the original matrix $W_0 \epsilon R^{m*n}$ and only update the ΔW parameters. This update process can be represented as shown in Equation 1, where only A and B are training parameters. During the forward process, both W_0 and ΔW are multiplied by the same input and subsequently added, as illustrated in Equation 2.

$$W_0 + \Delta W = W_0 + BA, B\epsilon R^{m*r}, A\epsilon R^{r*n} \tag{1}$$

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{2}$$

The Llama2-7B model is fine-tuned with the collected dataset so that it can output results corresponding to the target instructions. The data format of the training dataset is "instruction+input+output", where instruction is the natural language interpretation of the task, input is the corresponding segment to be processed, and output is the desired response. In the entity and relationship extraction task, instruction also contains several output examples for higher quality responses. The collected threat intelligence reports are then analyzed using the trained model.

3.4. Knowledge Graph Construction

Utilizing the fine-tuned model, we can perform topic classification, entity and relationship extraction, and TTP classification on the collected textual threat intelligence, as shown in Algorithm 1. When inputting a report, we first judge its topic based on the title and the first paragraph content (Line 2). If it belongs to the threat intelligence, the model outputs the main object of this report (Line 4). Otherwise, we skip the report. Some reports provide numerous IoCs at the end. While the model can recognize these entities, it is challenging to directly extract relationships between entities from such a list or table. Therefore, when extracting entities and relationships from the report, we first filter out the IoCs at the end of the report, extract the entity words and their types (e.g., Hash, Domain, IP) using regular expressions, and establish relationships between these IoCs and the main object of this report. We then incorporate this information into the knowledge graph (Line 5).

Algorithm 1: Threat Intelligence Report Processing

```
   Input: Trained model, Collected Report dataset
   Output: Entity and relationship
1  foreach Report in Collected Report dataset do
2      Perform topic classification;
3      if Belong to threat intelligence then
4          Output main object;
5          Process end-of-report IoCs;
6          Get each section of the report;
7          foreach Section in the report do
8              Conduct entity and relationship recognition;
9          if Main object belongs to malware or attacker
            then
10             foreach Sentence in the report do
11                 TTP classification;
12                 Establish relationship between TTP and
                    main object;
13         return Entities and relationships;
14     else
15         Skip this report;
```

Additionally, entities and relationships are extracted section by section using the trained model. Since we preserve the paragraph structure when crawling the text information, analyzing each section separately helps retain contextual information, as a section often introduces a complete topic. If a section exceeds the model's input length, we divide it according to the input length and paragraphs, striving to preserve the information within the paragraphs (Line 6-8).

Next, TTP classification is performed on reports whose main object belongs to attacker or malware. Since the input in the TTP classification training dataset consists of single sentences, we also use single sentences as input when employing the model for this step (Line 9-12). Finally, the entities and relationships contained in each report are output (Line 13).

However, directly using this information extracted through these steps can't complete a threat intelligence knowledge graph, and it needs further processing. One problem is that models tend to focus more on what the current sentence describes when extracting inter-entity relationships, and in some cases relationships can be interleaved. For example, a relationship may appear as malware A using PowerShell, followed by PowerShell executing the msf.ps1 file (A-PowerShell-msf.ps1), and another relationship may be B-PowerShell-cmd.exe. PowerShell is a commonly used tool in malware. Directly connecting PowerShell with msf.ps1 or cmd.exe may lead to confusion in relationships, as it would be unclear which malware executed the specific file due to multiple malware instances connecting with PowerShell. To address this issue, the main object of the report is taken as an attribute of the entity through which the threat intelligence correlation information about a particular malware or attacker can be depicted. The model may also extract relationships containing non-specific words such as "attacker", "campaign" or "malware" as entities from a sentence description. We further convert these words into corresponding entities that have appeared in the preceding text to enhance the accuracy and specificity of the extracted relationships in the knowledge graph.

Another problem is the data redundancy in the extracted entities and relationships, which necessitates consolidation. For entities with specific formats, such as URLs, IPs, CVEs, and files, different values carry distinct meanings, so we do not perform entity fusion for these types. For malware, the same malicious software may be described differently in various reports, such as "Backdoor.Pterodo" and "Pterodo". We merge these terms using rule-based methods. For attack types and relationships, we adopt a clustering-based approach for consolidation. First, we calculate the word embeddings for each term and use cosine similarity as the distance metric. We then perform hierarchical agglomerative clustering (HAC) on these terms. Taking into account the frequency of word occurrences and the distance to the cluster center, we calculate the representative term for each cluster using a method similar to Equation 3, with the highest weighted term serving as the representative word for a particular cluster.

$$score = \alpha * z(f) + \beta * z(\frac{1}{d}) \tag{3}$$

Where α and β are the set weights, z denotes Z-Score standardization, and f and d represent the number of occurrence and the distance from the clustering center, respectively. The threat intelligence knowledge graph is updated by the results of entity and relationship fusion.

4. Experiment & Analysis

This section evaluates our proposed method LLM-TIKG through experiments, giving the experimental setup and analyzing the evaluation results.

4.1. Experimental Setup

First, we collect a total of 12,545 pieces of content, including blogs, news articles, and security analysis reports in recent years, from various threat intelligence sharing platforms. The labeled data are obtained using the aforementioned methods, and the dataset size for each task is presented in Table 2. The model fine-tuning process is implemented using Python 3.9 on 2×3090 GPUs, with a maximum learning rate of 1×10^{-4} and a maximum of 10 epochs. The maximum sequence length was set to 1024 tokens for the Entity and Relationship Extraction due to the inclusion of additional output examples in the input instructions, and 512 tokens for the remaining two tasks.

Table 2: The Dataset Size for Each Task

Task	Dataset Size
Topic Classification	2000
Entity and Relationship Extraction	1600/15000
TTP Classification	38946

To evaluate the performance of the proposed model, we employ Accuracy, Precision, Recall, and F1-Score as evaluation metrics. Three experiments are designed as follows:

1) **Performance on tasks:** We evaluate the performance of the model in named entity recognition and TTP classification tasks through experiments, and compared it with some currently popular methods.

2) **Fine-tuning settings:** We observe the impact of dataset quality and the number of training epochs on results of the fine-tuning model through experiments.

3) **Performance on real datasets:** To test the model on a real dataset, we apply the trained model to construct a threat intelligence knowledge graph on all the collected data.

4.2. Performance on Tasks

4.2.1. Named Entity Recognition

Named Entity Recognition requires extracting entity words and their corresponding types from textual information. In order to compare this method with other models, we train the BERT-CRF model with 20 epochs using the same dataset (manually modified version), which is currently one of the most commonly used methods for named entity recognition. GPT3.5 and GPT4 are powerful out-of-the-box LLMs, and they are able to give valid answers based on input prompts on multiple domains. We use GPT3.5 and GPT4 with the prompt templates shown in Fig. 3(b) for named entity recognition. Then we test on BERT-CRF, GPT-3.5, and GPT4 models using the same test set.

The experimental results are shown in Table 3, where LLaMA-15000 and LLaMA-1600 represent different fine-tuned dataset sizes, respectively, as explained in more detail in 4.3.1. From the results, it can be seen that the BERT-CRF model has slightly lower results. This can be attributed to two primary factors: the relatively limited size of the training dataset and the over-representation of non-specific entity words within the dataset. The latter issue arises due to the fact that the corpus in the training data is the whole article, resulting in a reduced presence of specific types of entity words and, consequently, lower test results for the BERT-CRF model. Since GPT-3.5 and GPT4 have a strong comprehension of natural language text, their generated results have a high recall, but some misclassifications can occur. These errors are corrected in the training dataset, so the fine-tuned Llama2-7B model performs better in terms of precision.

Table 3: The Results of Named Entity Recognition

Model	Precision	Recall	F1-Score
BERT-CRF	70.98	66.40	68.61
GPT3.5	77.83	**95.72**	85.85
GPT4	79.30	94.73	**86.33**
LLaMA-15000	68.72	77.78	72.97
LLaMA-1600	**87.88**	83.99	85.89

Nevertheless, the Llama2-7B model's entity extraction capabilities are less effective when applied to very long text and non-natural language descriptions like tables. Consequently, some omissions may occur in these contexts. The accuracy of named entity recognition can be further improved by some methods such as increasing the number of certain types of entities in the dataset, in cooperation with regular expression matching, and increasing the input sequence length of the model using some advanced methods.

4.2.2. TTP Classification

The TTPs used in the attack can be classified based on the textual description of the attack process. The MITRE ATT&CK matrix includes both techniques and sub-techniques, with 229 distinct technique categories and 595 sub-technique categories incorporated in our dataset. Unrepresented techniques within the dataset are not considered as separate classification categories. We conduct tests using both classification methods. Moreover, the dataset is divided into a training set and a test set at a ratio of 10 : 1. Since the dataset contains negative samples (i.e., data without TTP), accuracy is added to the evaluation indices. Owing to the considerable difference in the number of categories, the model exhibits enhanced accuracy when a smaller number of categories are involved utilizing technique classification, as depicted in Table 4.

The same dataset with techniques is also tested on TTPDrill [32] and Tram [31]. TTPDrill, which employs rule-based matching and provides models and patterns in their github, is not very accurate. The rule matching method will fail when new categories and new expressions appear. Tram is a method based on logistic regression and Naive Bayes for extracting TTPs from textual reports. It also provides a containerized environment with Docker to automate the mapping of CTI reports

Table 4: The Results of TTP Classification

Model	Accuracy	Precision	Recall	F1-Score
TTPDrill	10.23	8.17	52.15	11.05
Tram	23.83	23.07	99.49	37.45
LLaMA-Tec	**97.47**	**96.53**	**99.95**	**98.21**
LLaMA-SubTec	83.60	77.89	99.80	87.50

to MITRE ATT&CK, considering only 50 techniques during model training. Consequently, its accuracy diminishes when applied to texts containing a broader range of techniques. However, its accuracy can reach 85.37% for the 50 techniques it is trained on. Our method can reach 97.47% accuracy in classifying all technique categories. The fine-tuned large language model gradually "understands" the meaning and distinctions of these techniques, demonstrating superior performance in extracting a more extensive array of TTP categories.

4.3. Fine-tuning Settings

4.3.1. Dataset Quality

The performance of the entity extraction dataset generated using GPT-3.5 in terms of accuracy is somewhat different from the performance of manual labeling. Despite the implementation of a rule-based filtering method for the extracted entities, inaccuracies persist in the form of omitted and misjudged information. On this basis, we conducted manual modifications on a subset of $1,600$ data samples, each comprising a paragraph. Subsequently, the model is trained on two distinct datasets: $15,000$ instances of coarsely filtered data and the $1,600$ manually modified samples with the same training parameter settings.

The results of LLaMA-15000 and LLaMA-1600 are shown in Table 3, and the use of manually modified data makes the results more accurate. The accuracy of the model is improved by manually correcting the labeling errors in the dataset. The results illustrate that when a model is trained using a dataset that includes inaccurate data, it may encounter a degree of mislabeling, consequently impacting the model's output, even though the number of datasets is larger with more correct labels contained in them. From this, we hypothesize that when fine-tuning a large language model, the quality of the training dataset is more important, and that a relatively small amount of high-quality data can lead to better results. Of course, increasing the size of the high-quality dataset can also improve the model's output to some extent.

4.3.2. The Number of Training Epoch

When studying deep learning models, choosing the appropriate number of training rounds is crucial to obtain excellent performance. Compared to BERT, the number of epochs required for fine-tuning the large language model decreases, but the training time increases, due to a significant increase in the model parameters. Since the size of the finetuning dataset for

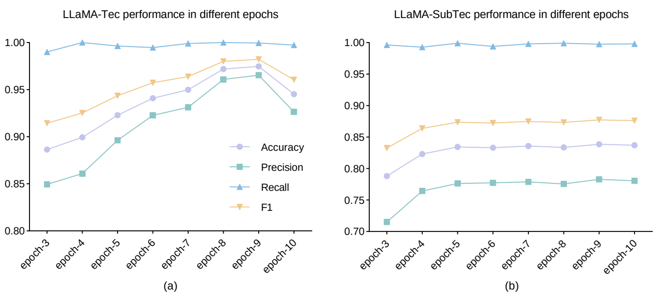


Figure 4: The Results Corresponding to Different Training Epochs. The figures a and b represent the results in the training dataset with technique and sub-tectechnique, respectively.

TTP classification is relatively large, we would like to know what epoch is the most appropriate value.

Therefore, we test it using the same data on models trained at different epochs. The results corresponding to different training epochs are illustrated in the Fig. 4. As the number of rounds increases, the model's performance gradually improves. Upon reaching a certain epoch, the model's learning capacity becomes saturated, meaning that the accuracy reaches a peak value. In subsequent epochs, the accuracy exhibits some fluctuations and experiences a decline.

4.4. Performance on Real Datasets

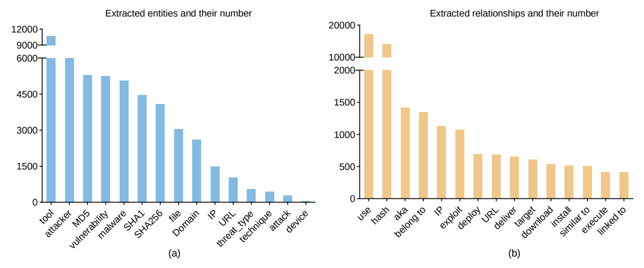


Figure 5: The number of Extracted entities and relationships. Figure a represents the extracted entities and Figure b represents the highest number of relationships.

Utilizing the trained model, we classify 9681 articles as threat intelligence reports from a collection of $12,542$ articles. The extracted and processed information is deposited into the graph database Neo4j, which converts the list into a graph. This process results in the extraction of $50,745$ entities and $64,948$ relationships.

The categories and quantities of entities are depicted in Fig. 5(a), while the 15 most frequent relationship types are illustrated in Fig. 5(b). Of these, the largest number of entities extracted is tool, followed by attacker and MD5. Based on the number of relationships associated with other entities, the most common tools are PowerShell, Cobalt Strike, C2 server, and Mimikatz. The attacker entity contains some aliases for the attacker due to the fact that different analysts may have different ways of naming the attacker, which corresponds to the relation "aka". Among the extracted relationships, "use" has the highest number, which indicates the invocation relationship between entities such as attackers, malware, and tools.

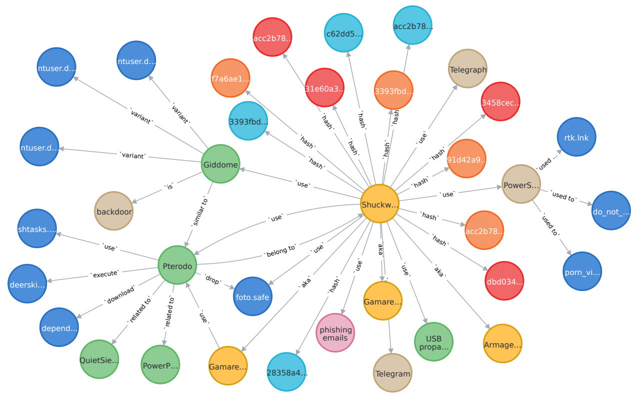


Figure 6: A portion of the constructedthe knowledge graph associated with Shuckworm, where yellow nodes represent attacker, green nodes represent malwares, blue nodes represent files, and light blue, red, and orange nodes represent hash values.

Fig. 6 presents a portion of the constructed knowledge graph, showcasing some threat intelligence information associated with the attacker, Shuckworm. The figure reveals some of the malware and tools used by Shuckworm, along with related IoC information. The graph illustrates the connections between various threat intelligence data, providing insights into the malicious activities of the attacker.

5. Case Study

In this section, we explain how the constructed Threat Intelligence Knowledge Graph can be applied to threat hunting and attack attribution through two practical examples.

5.1. Threat Hunting

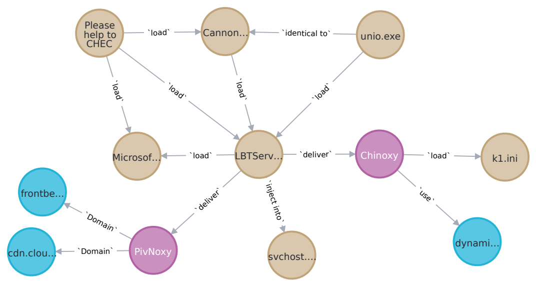


Figure 7: A portion of the constructedthe knowledge graph related to the intrusion process of PivNoxy, where purple nodes represent malwares, brown nodes represent files, and blue nodes represent domains.

By constructing the threat intelligence knowledge graph from attack analysis reports, textualized threat attack behavior can be converted into graph form. Fig. 7 illustrates the nodes and relationships in the constructed knowledge graph related to the intrusion process of PivNoxy. It uses phishing to lure the victim to download "Please help to CHECK.doc" in the email, which then loads three files which load each other, and eventually LBTServ.dll injects itself into svchost.exe and spreads the malicious payload.

To avoid detection, attackers usually update their used IPs and domains from time to time. Using only IoC values for intrusion detection is ineffective. However, a distinction exists between malicious attack behavior and normal behavior, notwithstanding the attacker's best efforts to minimize this distinction. Additionally, certain similarities can be discerned among malicious actions. As shown in Fig. 1 and Fig. 7, the threat intelligence knowledge graph contains actions of the attacker, which can serve as malicious samples for training the model, thereby enabling it to acquire knowledge about the attacker's behavioral traits.

5.2. Attack Attribution

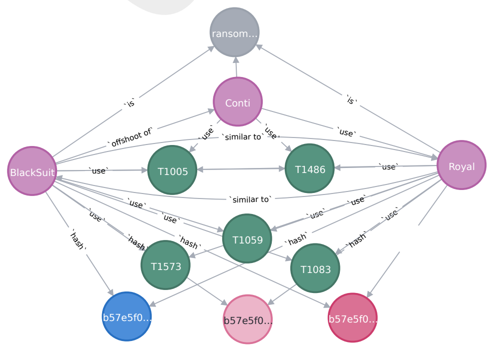


Figure 8: A portion of the constructedthe knowledge graph that shows the similarity between BlackSuit and Royal Ransomware,where purple nodes represent malwares, gray nodes represent threat types, green nodes represent techniques, and the remaining nodes represent hash values.

The information contained within the knowledge graph can be leveraged to discern similarities between entities, such as malware and attackers. In instances where multiple attackers employ identical malware and techniques, there is a high likelihood that they are affiliated with the same organization. Similarly, when distinct malware strains employ shared techniques, exhibit the same IoCs, and demonstrate comparable attack behavior, it is probable that they belong to the same malware family.

Fig. 8 shows some of the nodes and relationships where BlackSuit and Royal Ransomware have associations. It can be found that the similarity relationship between them has been labeled in the graph. The diagram reveals that both entities have established connections with specific nodes, signifying their affiliation with the ransomware category. Notably, they are both linked to another ransomware strain, Conti. Furthermore, they have common hash values present in their samples and employ several common techniques, such as Data from Local System (T1005), Command and Scripting Interpreter (T1059), and Data Encrypted for Impact (T1486). Through methods such as similarity matching or link prediction, we can conduct similarity analysis of attackers or malware in the graph, with the goal of attack attribution. With this information, a deeper understanding of cyberattacks can be gained to help take appropriate measures to counter the threat.

6. Related Work

In this section, we provide a detailed review of the work related to the threat intelligence.

6.1. Threat Intelligence Analysis

Open-source threat intelligence analysis primarily relies on unstructured text data as input and depends on rule matching or NER techniques to extract IoC entities and their corresponding labels [33, 34]. Generally, the extraction process in NER begins by defining the types of IoCs, such as file, domain name, IP address, attacker, and vulnerability. The B-I-O sequence tagging method is then employed to annotate the text data, followed by the use of NLP methods like Doc2Vec [35] to embed the textual language into feature vectors. The Bidirectional Long Short Term Memory + Conditional Random Fields (BiLSTM+CRF) model [36] is a commonly used model for entity recognition, tagging entity label types. On this foundation, some studies have incorporated self-attention mechanisms and multi-granularity attention mechanisms to enhance the accuracy of IoC entity extraction. Gao et al. [34] proposed an neural-based sequence labelling model using a self-attention module and contextual features to identify IoCs from cybersecurity articles. But they only extracted IoC entities, without extracting relationships between entities.

The diverse range of online platform resources offers various avenues for obtaining threat intelligence, making the extraction and application of threat intelligence information a crucial area of research. Social media platforms such as Twitter and Weibo are characterized by high interactivity, extensive information coverage, and strong timeliness, making them valuable sources of cybersecurity-related resources. Given these attributes, Dionísio et al. [37] employed an end-to-end neural network that does not require feature engineering to automatically filter out a large amount of irrelevant information. They then introduced Word-level Bi-LSTM layers and Tweet-level Bi-LSTM layers to achieve threat information processing and security entity recognition extraction. Additionally, some research has mined threat intelligence related to open-source projects and libraries from reported errors and issues in public code repositories such as GitHub and GitLab [38]. These methods are more suitable for threat intelligence extraction in specific scenarios, and do not apply in generel threat intelligence reports.

6.2. Knowledge Graph Construction

After extracting the entity words, it is necessary to further extract the relationships between the entity to construct the knowledge graph. This process is typically achieved through rule definition, semantic parsing, or sequence tagging [12, 39, 13]. ThreatKG [14] was proposed to collect OSCTI reports from diverse sources, extract high-fidelity threat knowledge and construct a threat knowledge graph, thereby automating these processes. CSKG4APT [40] leveraged the power of BERT for extraction of threat intelligence. The data was fed into BERT to obtain word vectors, which were then sequentially fed into BiLSTM, GRU and CRF modules to obtain entities and labels. Nevertheless, they extracted only a few predefined entities and relationships and ignored the attack behaviors in text messages.

7. Conclusion

In this paper, we propose an automated and universal method for constructing the threat intelligence knowledge graph based on the large language model. By leveraging the few-shot learning capability of GPT and constructing corresponding prompts, we achieve data annotation and augmentation, thereby creating the necessary datasets for model training and reducing the cost of manual annotation. Then, we fine-tune the Llama2-7B model to perform topic classification, entity and relationship extraction, and TTP classification for threat intelligence reports, resulting in the construction of a threat intelligence knowledge graph. Experiments prove that our method is more effective than other methods in named entity recognition and TTP classification. In future work, we will explore ways to further enhance model performance to release the greater potential of large language models.

Acknowledgment

This work is supported by the State Grid Corporation of China Science and Technology Project (5700-202319297A-1-1-ZN)

References

[1] Y. Zhou, Y. Tang, M. Yi, C. Xi, H. Lu, Cti view: Apt threat intelligence analysis system, Security and Communication Networks 2022 (2022) 1–15.
[2] B. Shin, P. B. Lowry, A review and theoretical explanation of the 'cyberthreat-intelligence (cti) capability'that needs to be fostered in information security practitioners and how this can be accomplished, Computers & Security 92 (2020) 101761.
[3] J. Kotsias, A. Ahmad, R. Scheepers, Adopting and integrating cyber-threat intelligence in a commercial organisation, European Journal of Information Systems 32 (1) (2023) 35–51.
[4] Symantec-enterprise-blog, https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence.
[5] Fortinet, https://www.fortinet.com/blog/threat-research.
[6] Trendmicro, https://www.trendmicro.com/en_us/research.html.
[7] Cisa-blog, https://www.cisa.gov/news-events/.
[8] Ibm x-force exchange, https://exchange.xforce.ibmcloud.com/.
[9] Alienvault otx, https://otx.alienvault.com/.
[10] F. K. Kaiser, U. Dardik, A. Elitzur, P. Zilberman, N. Daniel, M. Wiens, F. Schultmann, Y. Elovici, R. Puzis, Attack hypotheses generation based on threat intelligence knowledge graph, IEEE Transactions on Dependable and Secure Computing (2023).
[11] L. F. Sikos, Cybersecurity knowledge graphs, Knowledge and Information Systems (2023) 1–21.

[12] J. Zhao, Q. Yan, X. Liu, B. Li, G. Zuo, Cyber threat intelligence modeling based on heterogeneous graph convolutional network, in: 23rd international symposium on research in attacks, intrusions and defenses (RAID 2020), 2020, pp. 241–256.

[13] I. Sarhan, M. Spruit, Open-cykg: An open cyber threat intelligence knowledge graph, Knowledge-Based Systems 233 (2021) 107524.

[14] P. Gao, X. Liu, E. Choi, S. Ma, X. Yang, Z. Ji, Z. Zhang, D. Song, Threatkg: A threat knowledge graph for automated open-source cyber threat intelligence gathering and management, arXiv preprint arXiv:2212.10388 (2022).

[15] X. Wang, R. Liu, J. Yang, R. Chen, Z. Ling, P. Yang, K. Zhang, Cyber threat intelligence entity extraction based on deep learning and field knowledge engineering, in: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2022, pp. 406–413.

[16] L. Zhang, Y. Lei, Z. Wang, Long-text sentiment analysis based on semantic graph, in: 2020 IEEE International Conference on Embedded Software and Systems (ICESS), 2020, pp. 1–6. doi:10.1109/ICESS49830.2020.9301570.

[17] J. Chen, Z. Wang, R. Tian, Z. Yang, D. Yang, Local additivity based data augmentation for semi-supervised ner, arXiv preprint arXiv:2010.01677 (2020).

[18] Mitre att&ck, https://attack.mitre.org/.

[19] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A survey of large language models, arXiv preprint arXiv:2303.18223 (2023).

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[21] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, arXiv preprint arXiv:2204.02311 (2022).

[22] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).

[23] Openai, https://openai.com/blog/chatgpt.

[24] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: Open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288 (2023).

[25] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, arXiv preprint arXiv:2109.01652 (2021).

[26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Advances in Neural Information Processing Systems 35 (2022) 24824–24837.

[27] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685 (2021).

[28] sophos, https://news.sophos.com/en-us/category/threat-research.

[29] thehackernews, https://thehackernews.com/.

[30] krebsonsecurity, https://krebsonsecurity.com/.

[31] Tram, https://github.com/center-for-threat-informed-defense/tram/.

[32] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, X. Niu, Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources, in: Proceedings of the 33rd annual computer security applications conference, 2017, pp. 103–115.

[33] N. Luo, X. Du, Y. He, J. Jiang, X. Wang, Z. Jiang, K. Zhang, A framework for document-level cybersecurity event extraction from open source data, in: 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2021, pp. 422–427.

[34] Z. Long, L. Tan, S. Zhou, C. He, X. Liu, Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling, in: 2019 international joint conference on neural networks (IJCNN), IEEE, 2019, pp. 1–8.

[35] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International conference on machine learning, PMLR, 2014, pp. 1188–1196.

[36] Z. Huang, W. Xu, K. Yu, Bidirectional lstm-crf models for sequence tagging, arXiv preprint arXiv:1508.01991 (2015).

[37] N. Dionísio, F. Alves, P. M. Ferreira, A. Bessani, Cyberthreat detection from twitter using deep neural networks, in: 2019 international joint conference on neural networks (IJCNN), IEEE, 2019, pp. 1–8.

[38] L. Neil, S. Mittal, A. Joshi, Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports, in: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), IEEE, 2018, pp. 7–12.

[39] K. Satvat, R. Gjomemo, V. Venkatakrishnan, Extractor: Extracting attack behavior from threat reports, in: 2021 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2021, pp. 598–615.

[40] Y. Ren, Y. Xiao, Y. Zhou, Z. Zhang, Z. Tian, Cskg4apt: A cybersecurity knowledge graph for advanced persistent threat organization attribution, IEEE Transactions on Knowledge and Data Engineering (2022).