Doctoral Dissertation

Applications of Natural Language Processing and Large Language Models to Cybersecurity

MAI, Trong Khang

Supervisor Beuran Razvan

Division of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science

September, 2025

Abstract

Like other domains, cybersecurity knowledge can be stored in textual data to facilitate cybersecurity practitioners' analysis, interpretation, and communication. For example, Cyber Threat Intelligence (CTI) reports, network design documents, system logs, and security guidelines are cybersecurity assets in a textual format. This leads to the crucial role of Natural Language Processing (NLP) in cybersecurity. However, state-of-the-art NLP models based on Deep Learning (DL) architecture (e.g., transformer) necessitate extensive training with significant labeled data. This issue demands the strong involvement of experts to create sufficient labeled data to ensure the model's performance. Additionally, the fast-changing nature of the cybersecurity field causes training data and the developed models to be quickly outdated, diminishing the practical applicability of these approaches. As a result, addressing the labeled data insufficiency and developing models with greater generalizability have become emerging trends in NLP and DL.

There are significant NLP and DL trends that can address the insufficiency of annotated data in developing and maintaining cybersecurity applications:

1. The Weak Supervision approach harnesses existing labeled data sources to create a new weak dataset suitable for model development. Weak Supervision-based approaches significantly reduce the need for extensive data labeling when developing new models for emerging problems.

2. The use of published open-source frameworks for NLP, like SpaCy, simplifies the development of NLP applications for those with limited linguistic knowledge. These frameworks provide tools to analyze sentences, paragraphs, and documents to gain linguistic insights. Then, the developers can take advantage of the analyzed results to solve their target problems. For example, the obtained grammatical relationships can be used to replace the actual cybersecurity relationships among entities.

3. The recent appearance of Large Language Models (LLMs) introduces promising approaches to resolving data insufficiency problems in cybersecurity. With billions of parameters pre-trained on vast datasets, LLMs demonstrate strong generalizability, enabling them to tackle unseen tasks with very few labeled examples.

In this work, we aim to address the insufficiency problems of annotated data in developing cybersecurity applications. To achieve this goal, we studied advancements in NLP, including Weak Supervision and LLMs, and their feasibility to tackle practical downstream tasks. Additionally, we sought to create applications that can support cybersecurity experts. We applied these advancements to specific downstream tasks to develop practical applications, such as report analysis and policy generation.

Our first application was a framework called RAF-AG, which supports the information-sharing process in cybersecurity. RAF-AG can transform CTI reports into simplified versions, such as attack paths. In developing RAF-AG, we utilized Weak Supervision and open-source NLP tools to utilize already annotated data from similar problems to solve the target problem. For evaluating RAF-AG, we collected 30 CTI reports from various sources and compared its results with those generated by a similar report analysis framework, AttacKG. It was shown that RAF-AG can outperform AttacKG in precision, recall, and F1 scores, recording values of 0.717, 0.722, and 0.708 compared to 0.337, 0.535, and 0.393, respectively.

We recognized the limitations of RAF-AG and aimed to study new models that demonstrate high generalizability, eliminating the need for text normalization. The emergence and popularity of LLMs brought up new potential for this thesis. We utilized commercial LLMs to develop a framework for policy generation. This application aimed to assist experts in creating fine-grained access control policies tailored to a specific IT environment. We employed a typical ICS network as a case study to create 181 fine-grained ABAC policies. To enhance the access control performance of generated policies, we implemented priority optimization for policy conflict resolution. Our tests with various optimization algorithms showed that optimized priority values can significantly improve the effectiveness of the generated policies, resulting in an F1 score of 0.994.

We examined the benefits and drawbacks of previous applications. This turned the focus to open-source LLMs to develop CyLLM-DAP, a framework designed to support the specialization of LLMs in cybersecurity. This effort promotes an effective DL technique for data scarcity, namely transfer learning, where we inject cybersecurity knowledge into open-source LLMs so that the models can be reused to better solve cybersecurity downstream tasks. The aim of this effort is to create cybersecurity-specific LLMs (CyLLMs). Our experiment showed that cybersecurity-specific LLMs can lead to significant performance enhancements (up to 4.75%) in downstream tasks such as text classification and Q&A when compared to the general base and instruct counterparts. Additionally, using insights from previously developed applications such as RAF-AG, CyLLM-DAP, and CyLLMs, we developed

a methodology to work with cybersecurity problems where annotated data insufficiency is present. We also included a report analysis approach based on the proposed methodology.

For each of the mentioned tasks, we began by conducting a survey to identify the advantages and disadvantages of current approaches. Next, we developed a novel methodology to tackle the current issues. Based on the availability of existing approaches published in other research, our experiments successfully (1) demonstrated the effectiveness of the proposed techniques and (2) identified the best methodology among those available. Ultimately, the methodologies, models, and data in this work were published to assist in addressing similar downstream tasks in cybersecurity.

Keywords: Weak Supervision, Large Language Models, data insufficiency, CTI report analysis, policy generation, cybersecurity

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisor, Associate Professor Razvan Beuran. I am thankful to Beuran Sensei for his care and support during my three years at JAIST. His valuable guidance on research methods and my personal well-being has significantly influenced my journey. I am grateful for the friendly and welcoming environment he fostered in our laboratory. His guidance will have a lasting impact not only on my time at JAIST but also on my future endeavors.

I would also like to thank my second supervisor at JAIST, Professor Yasuo Tan, for his insightful feedback on my research proposal and outlines. His comments and recommendations helped me identify limitations within my research, particularly concerning the precise phrasing needed for my thesis. Additionally, I extend my thanks to Associate Professor Inoue Naoya for his guidance during my minor research project. The results of this project, along with his insightful instructions, enabled me to view my main thesis from a different perspective, ultimately enhancing its quality.

I am grateful to NEC Corporation for the opportunity to engage in joint research with NEC members and JAIST. This collaboration taught me valuable lessons about teamwork and research methods. The financial support from this joint research allowed me to focus on my thesis, and the findings contributed significantly to my work. I would like to thank JAIST and its departments for supporting all related procedures during my study and research. JAIST Kagayaki has been the primary computing architecture I used for my thesis. I wish to express my gratitude to the members of Beuran Lab, especially Mr. Lee Jongmin and Miss Nguyen Huynh Phuong Thanh, for their assistance and for maintaining a friendly research environment. I wish to acknowledge the use of the Grammarly software for paraphrasing and language correction during the preparation of this dissertation.

I want to express my heartfelt gratitude to my mother and sister, who have always stood by my side through ups and downs. I also want to extend my sincere thanks to Mr. Le Duy Tan, who set the first step on my journey to JAIST, which has been the most important event of my life.

Contents

List of Abbreviations

ABAC Attribute-Based Access Control. 4

CTI Cyber Threat Intelligence. 2

CVEs Common Vulnerabilities and Exposures. 1

CyLLMs cybersecurity-specific LLMs. 21

DL Deep Learning. 1

ICS Industrial Control System. 19

IOC Indicators of Compromise. 17

IR Information Retrieval. 7

LLM Large Language Models. 2

NER Named Entity Recognition. 7

NLP Natural Language Processing. 2

RE Relation Extraction. 7

TTP Tactics, Techniques, and Procedures. 6

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Topic Overview

The complexity of modern cyberattacks negatively impacts the cybersecurity of computer systems. To address the newly emerged cyber incidents, cybersecurity practitioners must update their knowledge frequently. However, the sheer volume of newly generated information makes it difficult for them to maintain comprehensive knowledge in cybersecurity. For example, a record-breaking 40,009 new Common Vulnerabilities and Exposures (CVEs) were published in 2024, representing a 38% increase compared to 2023. Therefore, improving the automation of computer-based applications to support the work of cybersecurity experts has become a crucial focus of cybersecurity research. However, implementing and maintaining such applications require a large amount of annotated data. For example, a typical Deep Learning (DL) based cybersecurity application starts with the data collection and annotation process, where thousands of data samples are analyzed and labeled one by one by experts. After that, the DL models are trained with labeled data and used in the system to solve specific tasks. Whenever the need to update arises, these models are re-trained with newly annotated data. This typical development process necessitates the strong engagement of cybersecurity experts.

A significant amount of cybersecurity data is encoded and stored in a textual format because it is versatile and human-readable. This data type enables cybersecurity practitioners to analyze, consume knowledge, and share insights with others. Text data can arise from various cybersecurity activities, including logs that record past events on computer systems or reports that explain what occurs before, during, and after an attack campaign. The analysis of thousands of cybercriminal cases can accumulate a wealth of cy-

bersecurity knowledge, leading to the establishment of various Cyber Threat Intelligence (CTI) frameworks, such as MITRE ATT&CK [1]. The automatic analysis of such textual cybersecurity data is facilitated by Natural Language Processing (NLP), similar to other domains. Currently, state-of-the-art DL models are employed in many NLP-based systems to address issues in cybersecurity, such as threat detection and policy generation. Similar to applications related to other data types, the development of this text-based system also faces challenges related to annotated data insufficiency. Experts must engage deeply in annotating the text for training purposes. For instance, when developing DL approaches for automatic report analysis, it is crucial to ensure that experts' manual annotation of the data is conducted. During this annotation process, experts must read thousands of sentences and manually label important keywords and word dependencies. The labeled data generated from this process is then used to train the DL model, enabling it to analyze sentences or paragraphs to detect the attack techniques mentioned in the report.

To address the shortage of labeled data in the field of cybersecurity, a recent technique in NLP called Weak Supervision has emerged as a promising solution. Weak Supervision involves creating labeled data from existing sources that are only loosely related to the target problem. This approach reduces the costs associated with hiring experts to annotate data manually. In addition, many open-source NLP frameworks are published to enable easy analysis of cybersecurity texts for valuable insights. For instance, SpaCy [2] allows users with minimal linguistic knowledge to analyze sentences for sentence dependency trees, helping to understand the grammatical structure of sentences for further analysis.

Developing models with higher generalizability serves as an alternative to traditional DL models, which require extensive training on large labeled datasets. Recent Large Language Models (LLM) [3] demonstrate strong generalizability with zero-shot and few-shot in-context learning [4], significantly impacting cybersecurity research, particularly for problems that rely on textual data as input. Few-shot in-context learning enables LLMs to tackle problems with only a limited number of examples, while zero-shot learning eliminates the need for any examples at all. These capabilities of LLMs are achieved through a pre-training process involving data equivalent to millions of books. While commercial LLMs demonstrate robust problem-solving capabilities, recent advancements in LLM development are facilitating the creation of smaller, open-source LLMs tailored to meet specific niche requirements. This trend toward utilizing open-source LLMs in cybersecurity is gaining attention as it enhances data transparency and reduces dependence on third-party companies and organizations.

The main objectives of this thesis are as follows:

- To investigate the feasibility of emerging trends in NLP and DL, including Weak Supervision and LLMs, for developing cybersecurity applications. These trends can help address the current annotated data insufficiency problems, particularly for new systems.
- To develop various applications that assist cybersecurity experts, focusing on high-performance capabilities for tasks such as CTI report analysis and access control policy generation.
- To create a framework that supports key stages of domain-adaptive pre-training for open-source LLMs in cybersecurity. This framework will assist the experts in creating cybersecurity-specific LLMs that can later be reused in cybersecurity downstream tasks.

1.2 Thesis Contributions

The contributions of this thesis are threefold, as follows:

1. **A methodology to develop cybersecurity applications aiming at addressing annotated data insufficiency:** The fast-changing nature of the cybersecurity field, with new knowledge being generated daily, poses challenges for the implementation of Deep Learning-based cybersecurity applications. One major challenge is the insufficiency and rapid obsolescence of annotated training data, which complicates the development and maintenance of these applications. Consequently, significant expert effort is necessary to manually analyze and annotate the data. This ultimately prevents a timely response to emerging cyberattack trends. This thesis focuses on addressing labeled data insufficiency problems by leveraging recent advancements in NLP and DL, such as Weak Supervision and LLMs. While Weak Supervision allows us to reuse existing labeled data from similar systems and projects, LLMs with high generalizability require a small amount of labeled data to complete the task.
2. **Cybersecurity applications to assist security experts:** This thesis introduces a list of cybersecurity applications designed to assist security experts. The first application enables the rapid analysis of CTI reports, transforming them into attack paths to provide deeper insights into cyberattacks. This process accelerates information sharing among cybersecurity practitioners to stay alert to new cyberattack trends. The

second application focuses on the generation of Attribute-Based Access Control (ABAC) policies, which are created by LLMs instead of relying solely on human experts. These applications are published to support the ongoing development of similar initiatives in the cybersecurity field.

3. **Framework for cybersecurity-specific pre-training of LLM:** With the rising popularity of open-source LLMs, enhancing their performance in cybersecurity has become an essential task. Open-source LLMs offer similar generalizability to commercial models while ensuring data transparency and model customization. Continual pre-training, also known as domain-adaptive pre-training, is a type of transfer learning used to integrate domain-specific knowledge into open-source LLMs. This process enhances the performance of these models in their respective fields. While this approach has been shown to be effective in various domains, it remains underexplored in the field of cybersecurity. In this thesis, we develop and publish a framework for simplifying the domain-adaptive pre-training of open-source LLMs in cybersecurity. Researchers can utilize this framework to develop their own cybersecurity-specific LLMs with reduced implementation effort.

1.3 Thesis Outline

The remainder of this thesis is structured as follows:

1. **Chapter 2** presents the background and foundational concepts essential for understanding the subsequent chapters. Relevant and interesting works from other research are also presented briefly.

2. **Chapter 3** presents the research philosophy, how the ideas are established and connected throughout the thesis. The mapping between ideas and applications/chapters, the thesis's practicality, and limitations are also presented.

3. **Chapter 4** presents the CTI report analysis framework, namely RAF-AG, to transform a sophisticated CTI report into attack paths.

4. **Chapter 5** discusses the use of highly generalizable models (LLMs) to generate fine-grained access control policies in specific IT networks.

5. **Chapter 6** presents a framework for domain-adaptive pre-training of LLMs, namely CyLLM-DAP. CyLLM-DAP simplifies the process of creating cybersecurity-specific LLMs to facilitate the use of open-source LLMs in cybersecurity.

6. **Chapter 7** presents the methodology to combine previous approaches in solving cybersecurity downstream tasks with data insufficiency. We then present a different version of the CTI report analysis, serving as an example for the proposed methodology.

7. **Chapter 8** summarizes the key findings and contributions of the dissertation. It also discusses potential future work that can follow up on this thesis.

Chapter 2

Background and Related Work

This chapter introduces the background knowledge and related research essential to understanding subsequent chapters in this thesis.

2.1 MITRE ATT&CK Knowledge Base

MITRE ATT&CK is a CTI framework that analyzes real cyber incidents to develop its Tactics, Techniques, and Procedures (TTP) knowledge base. Tactics outline the goals of malicious activities, while Techniques describe methods to achieve these goals. For example, to perform the "Initial Access" (TA0001) tactic, attackers might use the "Phishing" (T1566) technique to send malicious messages to users. Procedures in ATT&CK provide specific examples of how techniques are implemented, offering low-level information for technique recognition. For example, the procedure "APT1 has sent spearphishing emails containing malicious attachments" helps in recognizing the "Phishing: Spearphishing Attachment" technique in specific scenarios.

ATT&CK framework is utilized in many research works [5, 6, 7, 8] as the main knowledge base. For example, Xiong et al. [8] analyze ATT&CK data to create systematic threat models representing cyber threats and their relations, which can simulate real-world enterprise attacks. However, it requires frequent expert reviews of the ATT&CK knowledge base to remain updated. This heavy reliance on cybersecurity expertise led to exploring a new, less expertise-dependent approach which can be implemented and updated more easily.

2.2 CTI Report Analysis

In the field of cybersecurity, natural language serves as a prevalent medium for information exchange among professionals. Generally, casual readers prefer natural language text over more complex formats like STIX when reading. CTI reports are a particular form of cybersecurity information, articulated in natural language to depict entities such as cyber attackers, malware, and cyber incidents. These reports are made available on various websites and blogs to ensure easy and global access for all. By following well-known cybersecurity bloggers, end-users and defenders can stay informed about the latest attack strategies.

In this computer-based systems and data mining era, CTI reports serve as valuable resources due to their informative and plentiful nature. However, utilizing these reports directly in computer systems presents challenges because they are written in natural language. To automate the process of analyzing these reports, researchers have implemented NLP techniques within the cybersecurity field. Recently, transformer-based language models, such as BERT [9] and Roberta [10], are used to solve language-related tasks. For instance, Chen et al. [11] develop and train a BERT-based DL model for malicious sentence identification inside CTI reports, resulting in a high F1 score of 94.70%. This thesis introduces a framework for CTI report analysis, namely RAF-AG (see Chapter 4). In RAF-AG, we identify cybersecurity events in sentences and build a graph for the input report. These cybersecurity events are ultimately used to recognize the attack techniques via graph alignment, which is a more robust approach.

Information Retrieval (IR) is a field of research dedicated to the extraction of significant data from unprocessed text. Within IR research, Named Entity Recognition (NER) and Relation Extraction (RE) are widely used methods for identifying key entities and their relationships within the input data, respectively. We show a data sample with annotation for IR purposes in the cybersecurity domain in Figure 2.1.

In LADDER [12], the authors employ three unique DL models, each trained on relevant datasets to identify entities, related sentences, and relationships within a specific text. While a single DL model can extract entities and relations simultaneously, the challenges in dataset annotation remain the same. For example, in the CyberEntRel [13] framework, Alam et al. manually annotate 100 CTI reports to create training data, using the BIEOS (Begin, Inside, End, Outside, Single) tagging scheme. They used a sophisticated DL architecture, namely RoBERTa-BiGRU-CRF, for IR purposes. The methodology demonstrates promise with a 7% increase in the F1 score for report analysis. However, in practical applications, it is essential to

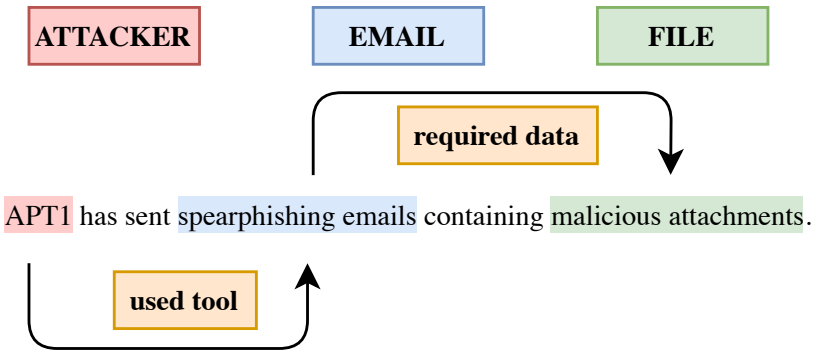


Figure 2.1: An example of Information Retrieval data sample with annotated word phrases (e.g., APT1, spearphishing emails, malicious attachments) and their important relationships.

continually annotate new CTI reports to sustain the system's high performance. In RAF-AG, we recognize the difficulty of this annotation task and aim to reduce the need for annotation.

To address the data insufficiency issues in developing DL-based models, training data is often manually labeled by cybersecurity researchers [12, 13]. This imposes a huge cost of hiring cybersecurity experts for data labeling, preventing efficient implementation in practice. For example, researchers behind AttacKG [6] have manually created a private NER dataset to implement its entity extraction function.

2.3 Weak Supervision and Sentence Dependency Tree

Weak supervision Weak supervision is a research area focused on addressing the challenge of insufficient labeled data. This method utilizes existing resources to create weakly annotated data, eliminating the need for extensive expertise. It consists of two primary stages to create the training data. The first stage entails a set of labeling functions, which can be derived through heuristic pattern analysis, keyword searches, or pre-existing DL models. The second stage involves modeling and combining the noisy outputs from these labeling functions to produce the final label. However, it is important to note that labels produced through a weak supervision approach are considered weak, as they do not match the precision of those provided by cybersecurity specialists.

Sentence dependency tree The sentence dependency tree [14] describes the grammatical relation between words in a specific sentence. The relationship between words (for instance, *nsubject* for nominal subject, *dobj* for direct object, and so on) can serve as an alternative to domain-specific relations. The method of creating a dependency tree from an input sentence is known as dependency parsing, which is a well-established area in NLP research. Numerous frameworks are available that can generate dependency trees for sentences with high precision, such as SpaCy [2]. Initially, a set of grammatical rules must be established to work with the sentence dependency tree. Following that, a tree traversal algorithm, in conjunction with the grammatical rules, can be employed to navigate the dependency tree to extract the desired relations.

Many cybersecurity research [5, 6, 7] have utilized sentence dependency trees for text analysis. For example, TTPDrill [5] utilizes the dependency tree to identify possible threat actions from the report text. The identified threat actions are then matched against those present in a threat action ontology. However, this ontology has been manually created by specialists through a thorough review of CTI frameworks like MITRE ATT&CK and CAPEC.

2.4 Machine Learning for Attribute-Based Access Control Policy Generation

ABAC is a dynamic method that allows or denies access to resources based on the assessment of policies in relation to specific attributes. An ABAC policy is a statement that combines attributes to set restrictions and conditions for access control management.

Recently, Machine Learning (ML) has been widely utilized to generate ABAC policies from various data sources. For instance, Narouei et al. [15] developed a dataset of 2,660 annotated sentences for policy generation using real-world documents. Heaps et al. [16] extracted access control information from user stories written by software developers, which involved identifying actors, data objects, and operations. Additionally, access logs serve as a valuable resource for policy mining, as demonstrated in [17, 18].

After generation, access control policies are optimized to reduce unnecessary complexity. One method involves clustering decision effects to minimize redundancy, as demonstrated in the work of Ait El Hadj and his team [19]. Additionally, recent research by Mitani et al. [20] introduces QI-ABAC, which leverages the intentions behind a policy manager's decision-making. This approach enhances the refinement of policies based on neural networks

using a limited set of initial policies. While the initial results are promising, several significant challenges remain related to generating initial policies, understanding intentions, and obtaining training data in real-world scenarios.

2.5 Large Language Models

In 2017, researchers at Google introduced transformers [21] featuring a multi-head attention mechanism, which marks a new era in DL research. This distinctive attention mechanism enables transformers to capture dependencies and patterns in the input data simultaneously. Furthermore, the efficacy of transformers relies on transfer learning, where models leverage previously acquired knowledge to tackle a new but related challenge. For instance, pre-training on textual financial data can enhance a model's effectiveness in addressing finance-related tasks. Typically, there are three primary transformer architectures: sequence-to-sequence (encoder-decoder) models, decoder-only models, and encoder-only models. Our discussion primarily focuses on decoder-only and encoder-only models, which represent the most successful architectures based on transformers.

Encoder-only architectures, like BERT [9], function as an embedding module by taking an input sequence and producing its representative vector. To adapt these models for specific tasks, various layers, such as the softmax layer for classification purposes, are incorporated on top of the encoder to yield appropriate outputs (e.g., class labels). Decoder-only architectures (GPT models) focus solely on the decoder part of the original transformer framework. These GPT models [22] exist in various sizes, from millions to billions of parameters. LLMs are those GPT models with billions of parameters designed for language comprehension.

The next token prediction is a core mechanism used in training LLMs. In this mechanism, the model learns to predict the next word (or token) in a sequence based on the context of the preceding words. In the training phase, when a sequence of k tokens $(s_1, s_2, ..., s_k)$ is inputted, the model learns to maximize the probability $\sum_i^k \log P_\Theta\left(s_i|s_1, s_2, ..., s_{i-1}\right)$ with Θ is the model's parameters. The model's parameters are then updated, allowing it to minimize the prediction error between its output token and the expected one. This process iterates over vast amounts of text data, allowing the model to learn natural language's statistical properties and patterns.

LLMs have been utilized in cybersecurity for different purposes. For example, LLMs can be used in various scenarios and training roles in cybersecurity training and education. Greco et al. [23] presents various promising strategies utilizing LLMs for PETA (Phishing Education, Training, and

Awareness). LocalIntel [24] is an LLM-based framework for generating organizational threat intelligence. Moreover, LLMs can also be used for software vulnerability detection [25], malware dynamic analysis [26], hardware security and policy generation [27], etc.

The growing interest in open-source LLMs, as opposed to commercial options like GPT-4 [3], can be linked to their long-term benefits. Open-source models provide enhanced customization, enabling users to modify model parameters entirely or partially to fit their training needs. They also allow users to train models from the ground up for experimentation or personal purposes. Additionally, the increasing accessibility of more affordable and higher-performing computers makes it more economical to train and host models in operational settings. Concerns about privacy also influence the preference for open-source frameworks, as businesses and organizations seek to retain control over their data. Recently, the Llama series [28] and the Mistral series [29] stand out as some of the top open-source LLMs, competing in performance with closed-source LLMs like GPT-4.

2.6 Domain-Adaptive Pre-training

Table 2.1: Comparison between CyLMM-DAP and other domain-adaptive frameworks/models in cybersecurity.

Frameworks/ Models	Model Type	Dataset Publication	Model Publication	Data Collecting Scripts	Filtering and Preprocessing Scripts	Domain-Adaptive Pre-training Scripts	Framework Source Code Publication
SecureBert	encoder-only		✓			✓	
CyBERT	encoder-only		✓				
CySecBert	encoder-only	✓	✓	✓	✓		
CyLLM-DAP	decoder-only	✓	✓	✓	✓	✓	✓

The available open-source LLMs are primarily designed to solve a wide range of problems. They tend to underperform when required to comprehend specialized knowledge areas. This limitation arises because LLMs are essentially statistical systems that remember patterns from textual data after exposure to large volumes of text. The lack of domain-specific data during the pre-training phase diminishes their effectiveness in the intended domain.

Domain-adaptive pre-training, or continual pre-training, is an advanced technique to customize a general-purpose language model for a specific domain by continuing its training using the domain's data. This process aims to improve the model's performance in a particular field without starting from scratch. As shown in the survey [30] regarding the LLM domain specialization, this technique has been extensively applied in various domains (e.g., finance, law) to improve the LLM performance in such domains. For

example, Wu et al. [31] develop PMC-LLaMa by pre-training the base model (Llama 2) with biomedical papers and books and subsequently fine-tuning for following instructions in the medical domain. In the evaluation of this research, domain-adaptive pre-training allows the model to reach 2.94% of the performance gain. In a recent study [32], the author reveals that employing transformers with pre-trained knowledge surpassed traditional ML approaches in the area of software vulnerability repairing. Additionally, following the pre-training of these models on a vast amount of codebase data, there is a significant improvement of 9.4% in accuracy. The authors recommend that pre-training models with information closely related to the specific downstream task is an effective strategy for boosting their performance.

In the field of cybersecurity, the concept of domain-adaptive pre-training has been introduced in various encoder-only models tailored for cybersecurity, such as SecureBERT [33], CyBERT [34], and CySecBERT [35]. However, to the best of our knowledge, there is currently no publicly available framework that facilitates this task of domain-adaptive pre-training for LLMs within the cybersecurity realm. In a comprehensive survey [36] of more than 180 projects across over ten downstream tasks on the use of LLMs in cybersecurity, only one mentioned paper [37] integrated domain-adaptive pre-training into its approach for binary code analysis. This finding highlights the current lack of discussion regarding the use of domain-adaptive pre-training for LLMs in cybersecurity applications.

From the above reason, we recognize that a framework for the cybersecurity domain-adaptive pre-training of LLMs would greatly benefit the cybersecurity research community. CyLLM-DAP—the framework developed in this thesis—makes it easier to utilize open-source LLMs in cybersecurity. In the Table 2.1, we can see a comparison of CyLLM-DAP with similar frameworks and models designed for the pre-training process specifically within the cybersecurity domain. The detailed implementation of CyLLM-DAP is introduced in Chapter 6.

Chapter 3

Methodology Overview

This chapter introduces the research philosophy, how the ideas are established throughout the thesis and their mappings with applications and chapters presented in this thesis. The thesis's practicality and limitations are also discussed.

3.1 Research Philosophy

In this section, we present the research philosophy behind the methodology presented in this thesis. A summary of the research philosophy can be observed in Figure 3.1.

Motivating Factors The main motivation for our thesis is derived from the following factors:

- Annotated data scarcity poses a significant challenge in the development of DL-based applications. In cybersecurity, the problem of insufficient annotated data becomes even more pronounced due to the rapidly changing nature of the field.
- Although expert knowledge is available, hiring these experts for data annotation is prohibitively expensive due to the complexity of the data annotation task and the high volume of labeled data required.
- Human resources are a valuable asset. Therefore, when developing cybersecurity applications, experts should prioritize auditing and refining the applications instead of manually labeling data.

- Data scarcity is not unique to cybersecurity; similar issues have been addressed in other domains. We believe that leveraging existing advancements from these fields and adapting them for cybersecurity can significantly enhance the development of cybersecurity applications.

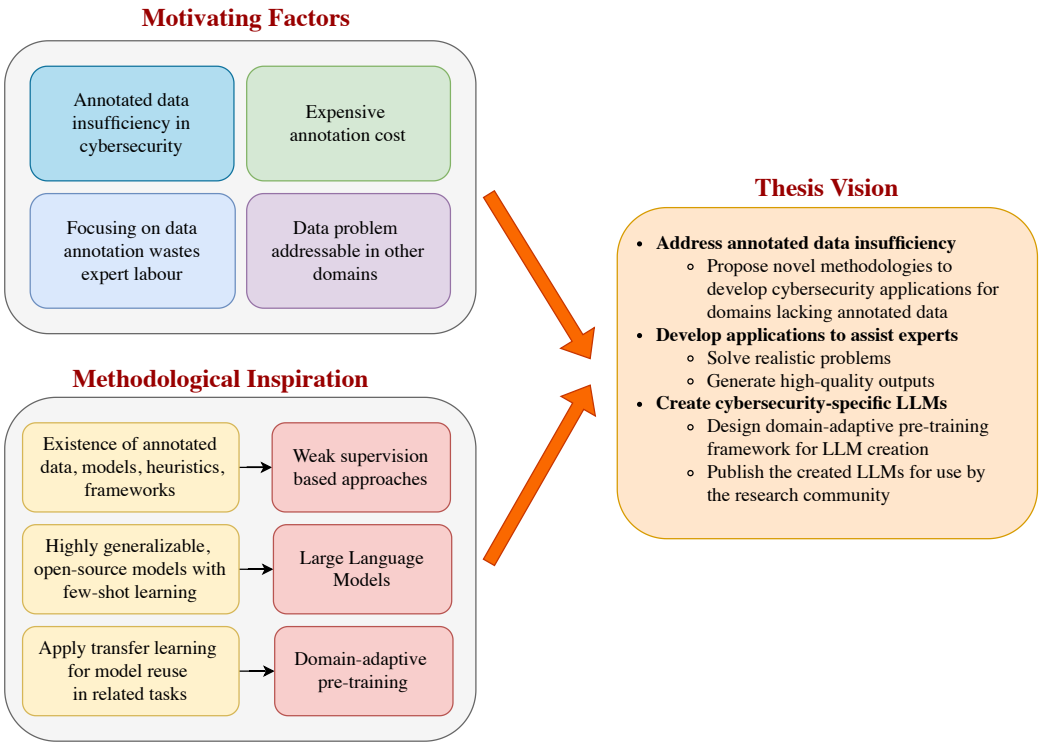


Figure 3.1: The research philosophy with motivating factors, methodological inspiration, and vision.

Methodological Inspiration Based on the knowledge established through the courses at JAIST (Machine Learning, Advanced Machine Learning) and a survey of cybersecurity research, the methodological inspiration for our approach is as follows:

- While annotated data insufficiency presents a significant challenge for cybersecurity, there are already established data from the research community, such as labeled data, trained models, rules, heuristics, patterns, and knowledge bases, CTI frameworks, such as MITRE ATT&CK, etc. We believe that utilizing these data sources can help address data scarcity when developing cybersecurity applications for new problems. Weak supervision is a method focused on reusing available data to

generate weakly labeled data suitable for the new task. By programmatically combining these weak signals and resolving any conflicts, we can create large, noisily labeled corpora that facilitate cybersecurity problem-solving without hiring annotators.

- The emergence of Large Language Models (LLMs), which possess strong language problem-solving abilities and high generalizability, allows them to work with various language patterns. Furthermore, the few-shot learning ability of LLMs requires only a limited number of examples to complete tasks, eliminating the necessity of annotating vast quantities of data. In addition, the publication of open-source LLMs by high-tech companies (e.g., Meta, Mistral) allows higher customization of LLMs for cybersecurity tasks.

- Transfer learning is a DL technique where a model trained on one task can be utilized to solve another, related task. By injecting cybersecurity-specific knowledge into LLMs through a method known as domain-adaptive pre-training, we can further enhance their performance in cybersecurity contexts. This pre-injection of cybersecurity knowledge represents a form of transfer learning, where unsupervised cybersecurity knowledge is integrated into the open-source LLMs in advance. Later, only a few model parameters must be altered during the fine-tuning process with supervised data to achieve high performance in solving the downstream tasks.

Thesis Vision Based on the main motivation and methodological inspiration, the thesis's general vision is as follows:

- The advancements addressing annotated data scarcity in other research domains can also benefit cybersecurity. We will study these advancements and incorporate them into methodologies for specific cybersecurity tasks. Finally, we will connect these methodologies and applications to create a novel method applicable to other tasks where labeled data insufficiency is also a problem.

- The developed applications should focus on realistic cybersecurity problems and generate high-quality outputs. In creating these applications, we will adhere to one key principle: aiming to avoid expert data annotation. These applications serve as examples of how the proposed methodologies can be applied, which is also applicable to future downstream tasks.

- The focus on open-source LLMs and domain-adaptive pre-training methods as a form of transfer learning inspires the development of a framework for implementing cybersecurity-specific LLMs. In the long term, the framework could bring the cybersecurity research community's attention to model customization of open-source LLMs. One can continuously update the framework to include additional data, diverse functions, and various model sizes to meet the diverse needs of the research community.

3.2 Flow of Ideas

This section presents the flow of ideas developed throughout this thesis. As shown in Figure 3.2, we begin with a survey (discussed in Section 3.2.1) to identify the current challenges in developing computer-based applications to solve cybersecurity problems. Next, we present the potential use of NLP and DL advancements, such as Weak Supervision and Sentence Dependency Tree (discussed in Section 3.2.2), for addressing the insufficiency of labeled data in developing cybersecurity applications. Subsequently, Section 3.2.3 focuses on utilizing LLMs that exhibit high generalizability, while Section 3.2.4 addresses the use of open-source LLMs along with a domain-adaptive pre-training approach.

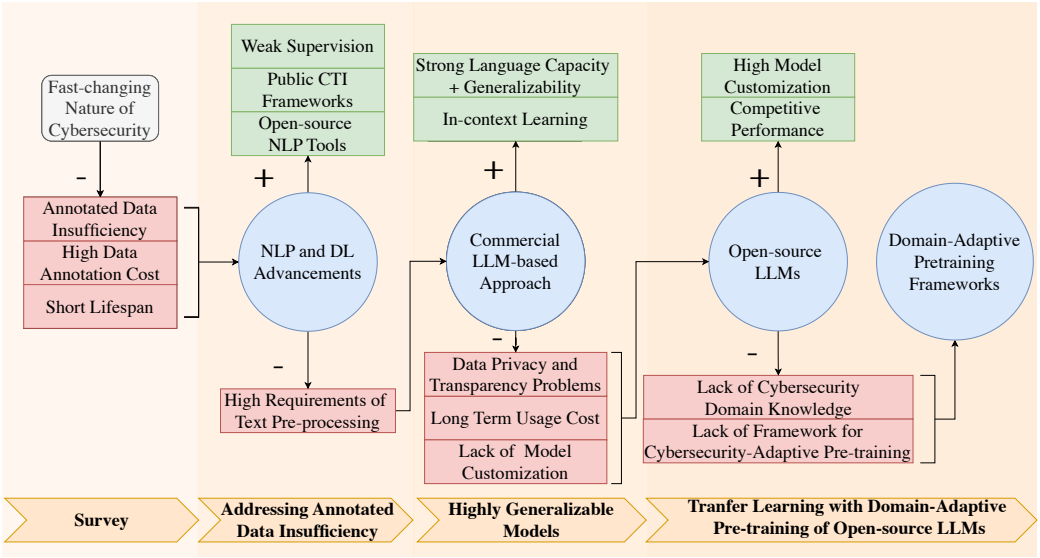


Figure 3.2: Flow of ideas in this thesis; the plus sign (+) shows the benefits, while the minus sign (-) shows the drawbacks of a proposed idea.

3.2.1 Survey and Main Thesis Theme

As previously mentioned, the cybersecurity field constantly changes with new devices, applications, and technologies, and attacking and defending methodologies are introduced daily. This rapid evolution makes it challenging for experts to maintain comprehensive knowledge to handle emerging attacking trends. Automated computer-based applications have been developed to assist experts in these cybersecurity tasks.

A preliminary idea for this thesis is to develop applications for cybersecurity tasks where labeled training data is insufficient. Additionally, recruiting cybersecurity experts for the annotation tasks should be avoided. The thesis began with a comprehensive survey to understand the challenges of developing such cybersecurity applications. We could determine some prominent problems commonly mentioned in the limitations of related research:

1. There is a lack of standard labeled data in cybersecurity. In many research studies, manually annotating the data to develop and evaluate new approaches for a new domain becomes unavoidable. For example, in the creation of training data for CTI report analysis, the authors of [13] used the BIEOS (Begin, Inside, End, Outside, Single) tagging scheme to annotate 100 CTI reports manually.

2. The data annotation in cybersecurity is not an easy task. To perform the annotation work, the expert must possess a sufficient level of cybersecurity knowledge and be familiar with the tagging mechanism. For example, data annotators with insufficient cybersecurity expertise are likely unable to correctly identify sophisticated Indicators of Compromise (IOC) in the text. Additionally, recruiting experts to annotate the data is costly in both time and money.

3. The annotated data and train models become easily outdated due to the fast-changing nature of the cybersecurity landscape. Therefore, the short lifespan of applications requires more frequent updates to keep pace with emerging cybersecurity knowledge.

Based on the survey results, we recognize the necessity to alleviate the expertise work required for implementing and maintaining such applications. Based on this, we determine the main theme of this thesis, which is represented via two points:

1. The utilization of new advancements in NLP and DL to partially address the insufficiency of labeled data in developing cybersecurity applications.

2. The employment of highly generalizable models to handle new problems. The generalizability reduces text pre-processing costs and the frequency of updates needed.

3.2.2 NLP and DL Advancements as a Potential Solution for Data Insufficiency Problems

In line with the dissertation's theme, we start by exploring new trends in DL and NLP that aim to address the insufficiency of labeled data in cybersecurity. One effective approach is Weak Supervision, which leverages existing sources to create weakly labeled data of cybersecurity entities. This eliminates the need for manual annotation of a new training dataset to develop new systems. In addition to this, we analyze the grammatical relationships among cybersecurity entities in the text to replace the cybersecurity relationships. This approach utilizes the Sentence Dependency Tree outputted by public NLP tools, completely eliminating the need to manually label the cybersecurity relationships in thousands of sentences. We study these approaches and employ them in our first application, RAF-AG, focusing on speeding up the information sharing among cybersecurity practitioners. The detailed implementation of RAF-AG will be presented in Chapter 4.

For RAF-AG, we implemented a Weak Supervision-based approach to eliminate the need for annotating thousands of sentences to gather sufficient data for model training. This approach also utilizes a published NLP framework called SpaCy [2], which allows users with minimal linguistic knowledge to analyze sentences effectively. We focus on leveraging grammatical relationships (outputted by SpaCy) instead of cybersecurity relationships. Additionally, a public CTI framework called MITRE ATT&CK is utilized for information linkage purposes, supporting further knowledge exploration of the detected attack techniques. While this application can successfully analyze the CTI report while minimizing the expert involvement, there are still some issues. The results analyzed by SpaCy require sentences to follow common language patterns to be stable. Although we have implemented a robust preprocessing function to normalize the input data, not all cases are covered.

3.2.3 Models with Higher Generalizability

The need to introduce new approaches with better generalizability has arisen to avoid the sophisticated implementation of text normalization. Recently, LLMs have been introduced and have demonstrated an ability to understand

information encoded in text, even when presented in different grammatical forms or paraphrased. This strong generalizability of LLMs allows us to develop text-based applications while avoiding text pre-processing costs. Additionally, LLMs' few-shot learning capacity requires a small number of labeled examples to do the task. This helps with addressing the data insufficiency problems.

Inspired by this, we have developed an LLM-based application to generate access control policies for Industrial Control System (ICS) networks. These policies are created in Python function format, which can be run directly in a Python environment. We also tested these policies with the help of experts from NEC Corporation. The results indicate that our method is effective in handling these complex tasks. We present the detailed methodology for this policy generation approach in Chapter 5.

However, the utilization of commercial LLMs in the mentioned application faces difficulties:

1. To conduct an API call to prompt the LLM, the data is leaving the premises. Depending on the LLM supplier's policy, we can configure it to prevent the storage of this data on the third-party side. However, attackers may still conduct a man-in-the-middle attack to intercept the data during transmission.

2. For tasks that require repeated API calls, thousands of requests could cost us a significant amount of money. This is not economically viable compared to hosting a local LLM to perform the task in the long term.

3. The lack of model customization makes it less attractive for researchers who desire greater control over the model's functionality.

3.2.4 Cybersecurity Adaptation of Open-source LLMs

The recent publication of open-source LLMs (e.g., LLama [28], Mistral [29]) with performance comparable to that of commercial LLMs has brought new insight to our thesis. We aim to utilize these published open-source LLMs in cybersecurity applications. After conducting a survey, we found that domain-adaptive pre-training is a crucial technique in LLM research. This method involves injecting specific domain knowledge into LLMs to gain better performance in that target domain. Domain-adaptive pre-training can be considered a form of transfer learning because the domain knowledge acquired through pre-training is reused to solve tasks in the same or related domains. However, in cybersecurity, this approach is often overlooked due to its cumbersome implementation. To the best of our knowledge, there are currently

no frameworks that support domain-adaptive pre-training in cybersecurity. We have developed a framework to facilitate this approach for LLMs. We believe that this framework will make a significant contribution to the research community by raising awareness of both open-source LLMs and domain adaptation. After successfully implementing the framework, we use it to develop cybersecurity applications to prove its effectiveness. We introduce a LLM-based CTI report analysis version that utilizes both RAF-AG, CyLLM-DAP, and CyLLM to solve the task.

3.3 Idea to Chapter Mapping

Figure 3.3 shows the relationships between the ideas established in this thesis and the developed cybersecurity applications. We also present where these ideas and applications are located in this thesis.

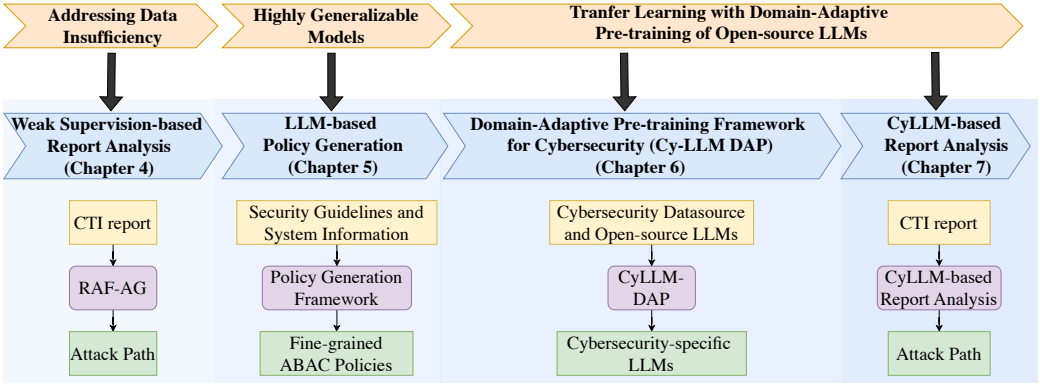


Figure 3.3: Mapping between ideas and applications/chapters in this thesis.

Aiming to address the data insufficiency problems in developing cybersecurity applications, we study the potential use of Weak Supervision and various publicly available open-source tools and frameworks. From this, we successfully develop a report analysis approach (see Chapter 4). The preliminary results were published at the non-peer-reviewed conference, the SCIS 2024 (2024 Symposium on Cryptography and Information Security) under the title "**Attack Path Extraction via Semi-Automatic Analysis of Cyber Threat Reports**". After that, we published a completed approach in the Computer & Security journal under the title "**RAF-AG: Report Analysis Framework for Attack Path Generation**". While we successfully achieved several predefined goals, such as generating attack paths and eliminating the need for annotation, this approach still requires robust text

normalizing functions. Therefore, the necessity for a highly generalizable approach arises.

To address the preprocessing cost, an approach utilizing high generalizability models such as commercial LLMs is studied and developed. We create an application for generating fine-grained ABAC policies using commercial LLMs (see Chapter 5), such as OpenAI's GPT models. This application was published in the ICISSP 2025 conference (The International Conference on Information Systems Security and Privacy) under the title "**LLM-based Fine-grained ABAC Policy Generation**". The models in this approach possess strong language capability and generalizability, allowing them to handle diverse text patterns without normalization. However, third-party models like GPT-4 often require data transmission to external environments, which can be undesirable in specific scenarios.

Consequently, we turn to leverage open-source models to tackle cybersecurity challenges. The release of high-level open-source LLMs by well-known companies like Meta and Mistral shows promise in this area. To enhance their performance in cybersecurity applications, we develop a framework (see Chapter 6) for collecting cybersecurity knowledge and integrating it into these models. This effort results in CyLLM-DAP (a framework for domain-adaptive pre-training of LLMs in cybersecurity) and cybersecurity-specific LLMs (CyLLMs). We published CyLLM-DAP and CyLLMs in the ICISSP 2025 conference under the title "**CyLLM-DAP: Cyber-security Domain-Adaptive Pre-training Framework of Large Language Models**". This paper received the Best Student Paper Award.

Finally, we combine RAG-AG, CyLLM-DAP, and CyLLMs to create an application for LLM-based report analysis (see Chapter 7). This application serves as an example of how previously discussed advancements and applications are combined into a methodology for cybersecurity problem-solving. We published the preliminary results of this approach in the SCIS 2025 conference under the title "**LLM-based Cyber Threat Intelligence Report Analysis**".

3.4 Practicality of the Thesis

This thesis studies various advancements in NLP and DL to establish novel methodologies for developing cybersecurity applications. Both the methodology and applications have their practical use in real-world scenarios:

- RAF-AG is a framework designed to generate high-quality attack paths for CTI reports. With an acceptable time frame for analyzing a CTI

report, as discussed in Chapter 4, the framework can process thousands of CTI reports daily. It is particularly useful in a Security Operations Center (SOC), where staying updated on the latest cybersecurity incidents is crucial. The generated attack paths serve as a compact version of the lengthy CTI reports, facilitating quick information consumption and sharing among cybersecurity practitioners.

- Being utilized as the main methodology for implementing RAF-AG, Weak Supervision, and Sentence Dependency Trees are valuable approaches for cybersecurity tasks related to text analysis. Unlike traditional DL-based approaches that require researchers to annotate thousands of sentences with cybersecurity entity labels and relationships, utilizing Weak Supervision allows the reuse of existing efforts to create weakly labeled data. Meanwhile, the Sentence Dependency Tree helps extract cybersecurity events based on grammatical relationships rather than cybersecurity-specific ones.

- The policy generation approach based on LLMs (presented in Chapter 5) can be applied in developing access control systems for organizations. In typical scenarios, experts must manually create and occasionally update policies. The proposed approach offers a faster solution for generating these policies, which can be directly employed to manage access control in the target system.

- Both CyLLM-DAP and CyLLMs discussed in Chapter 6 can be used for various language-related tasks in cybersecurity. CyLLM-DAP is a framework designed to support the specialization of open-source LLMs in cybersecurity. Researchers can use CyLLM-DAP to develop private LLMs focused on cybersecurity and its subdomains when issues with commercial LLMs, such as data transparency and model customization, arise. Additionally, our CyLLMs can serve as foundational models for tackling cybersecurity tasks. Users can further enhance the performance of these CyLLMs by conducting an additional fine-tuning process with their supervised datasets. Moreover, although CyLLM-DAP is designed for working with cybersecurity, one can modify its Relevance Filtering modules to work with other domains.

3.5 Limitations

While this dissertation provides beneficial contributions to the cybersecurity research community, there are limitations that should be acknowledged:

- RAF-AG (see Chapter 4) outputs attack paths in chronological order for a specific CTI report, ensuring that each attack technique is mentioned only once. We designed RAF-AG in this way because: (1) sequential orders of events are a common phenomenon in most cybersecurity incidents, and (2) we want to investigate the causal relationships between attack techniques. However, there may be exceptions, particularly in the case of complex attack campaigns.

- In the LLM-based policy generation approach in Chapter 5, relying on ground truth data for priority optimization is unavoidable. Due to context length limits, LLMs cannot grasp the comprehensive picture of all security guidelines to generate policies with optimized priority values simultaneously. As a result, the policy generation task is split and the LLM can only address each guideline one at a time. Additionally, conflicts can arise within security guidelines because they are often recommendations suitable for multiple systems rather than specific ones. These factors lead to priority values generated by LLMs not always being fully optimized. However, it has been shown that only a small amount of ground truth data is needed to effectively optimize these priority values.

- The cybersecurity-specific LLMs developed through CyLLM-DAP (see Chapter 6) are currently limited in size. At this time, we only support Llama 8.5 B (with 8.5 billion parameters) and Mistral 7 B. While these models are useful for tasks that require a low level of reasoning, larger models are necessary to meet the diverse demands of the research community. It is important to note that developing models larger than 30 B can cost hundreds of thousands of dollars.

- Cybersecurity-specific LLMs are pretrained with 30 GB of cybersecurity-related text. However, this training data is still limited when looking at the entirety of the cybersecurity landscape. Despite the high generalizability of LLMs, which reduces the need for frequent updates, it remains essential to collect more data and update the models regularly.

Chapter 4

Weak Supervision-based CTI Report Analysis

This chapter presents the first application in this thesis, which is a framework for CTI report analysis, namely RAF-AG. Weak supervision and the use of public NLP tools and CTI frameworks are key points for improving the automation of this framework.

4.1 Problem Introduction

The sophisticated nature of contemporary cyberattacks adversely affects the security of computer systems. It is vital for both cybersecurity professionals and everyday users to stay informed about cybersecurity developments. CTI reports from internet sources (e.g., online news, blogs) provide a beneficial resource for cybersecurity information. However, to fully comprehend these reports, we need a solid foundation in cybersecurity knowledge and a significant investment of time. Additionally, given that many CTI reports are produced daily, it can be challenging to extract the most critical information from them.

The recent emergence of publicly available CTI frameworks, such as MITRE ATT&CK [1] and CAPEC [38], has accelerated the advancement of defense strategies in the field of cybersecurity. Labeling the steps of cyberattacks using standardized tactic/technique names from these frameworks can facilitate the sharing and processing of information. This can be achieved, for instance, by transforming an extensive CTI report into a more concise version that highlights only the key information, such as a series of attack technique IDs. This type of representation offers a straightforward summary of the report, enabling further analysis to enhance the reader's understand-

ing. For instance, the technique IDs are ideal for broad searches within CTI frameworks to acquire additional related information (e.g., mitigation strategies).

While various DL techniques have been proposed to enhance report analysis for extracting crucial information such as attack paths, the lack of sufficient training data hinders the effectiveness of these methods in real-world scenarios. In line with the main theme of the thesis, the solution for report analysis (namely RAF-AG) presented in this chapter also aims to alleviate the expert involvement. To analyze CTI reports, RAF-AG employs a Sentence Dependency Tree-based approach for extracting entities and their relationships, alongside a Weak Supervision method for labeling entities. This process is then followed by constructing graphs and aligning them to generate the attack paths. Our methodology addresses the issue of data scarcity in the cybersecurity field by reducing the necessity for expert engagement. We assessed RAF-AG by comparing its attack paths and those from AttacKG [6]. AttacKG is a leading automatic report analysis framework. RAF-AG successfully discerned cyberattack steps by correlating their sequence within the report and connecting them to techniques from the MITRE ATT&CK knowledge base, achieving a better F1 score than AttacKG (0.708 compared to 0.393).

4.2 CTI Report Analysis with Weak Supervision for Expertise Work Reduction

We present RAF-AG, which is a **R**eport **A**nalysis **F**ramework for **A**ttack path **G**eneration. The framework utilizes MITRE ATT&CK to create the knowledge base, including a set of procedure graphs. The procedure graphs are established from the procedure examples via cybersecurity events extraction and graph building. CTI reports are the primary inputs of RAF-AG and are also transformed into the report graphs. The graph alignment is conducted to analyse the report graph using the framework's internal knowledge base. Finally, the framework's output is the attack path, which results from the graph alignment results. We show the overall architecture of RAF-AG in Figure 4.1, including three main components.

Information Retrieval: This component aims to extract cybersecurity events from the text.

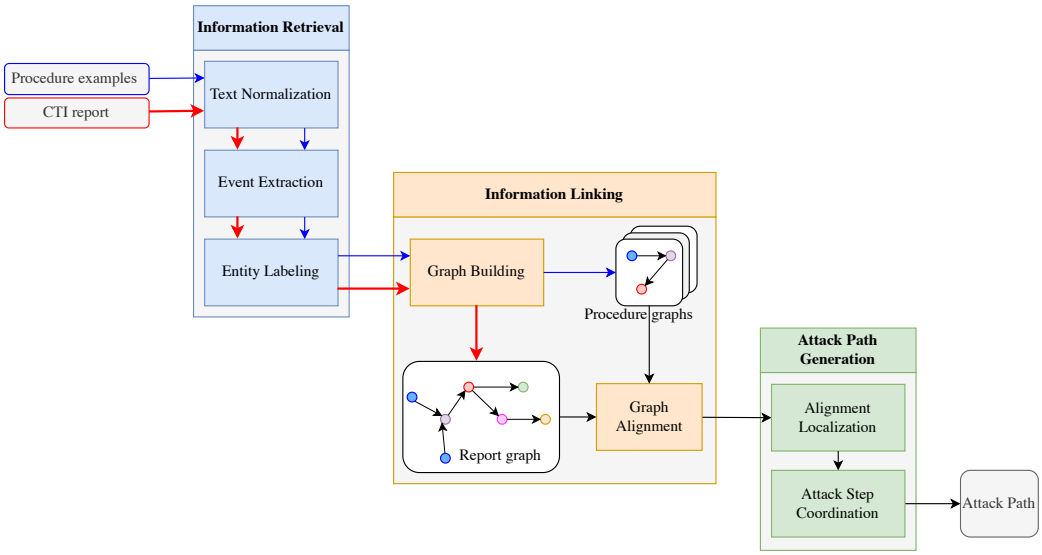


Figure 4.1: The general structure of the RAF-AG framework. Procedure examples are transformed into procedure graphs (blue path); report text is converted into report graph (red path); the attack path is derived from the report graph (black path).

Information Linking: This component aims to (1) construct the report graphs from cybersecurity events and (2) link the cybersecurity information to MITRE ATT&CK via graph alignment.

Attack Path Generation: This component aims to generate the attack path corresponding to the input CTI report using the graph alignment results.

4.2.1 Information Retrieval

In this section, we describe the information retrieval function of RAF-AG, including text normalization and cybersecurity event extraction.

4.2.1.1 Text Normalization

In NLP, text normalization serves as an essential first step to ensure that text data is uniform before any additional operations are conducted on it. By normalizing the text, we minimize linguistic and spelling variations, which simplifies processing and analysis. Inspired by studies in NLP and cybersecurity text analysis, RAF-AG has established four primary functions for text

normalization. We will briefly present these functions, with additional details provided in Appendix A.

1. **Unicode Fixing**: This function identifies and removes errors caused by Unicode characters in the text.
2. **Text Simplification**: This function simplifies the text by replacing complex phrases with equivalent simpler phrases.
3. **Subject Ellipsis Handling**: This function detects sentences without a subject and attempts to recover it.
4. **IoC Replacement**: This function detects Indicators of Compromise (IoC) and replaces them with placeholders to simplify the text.

4.2.1.2 Event Extraction

The objective of this module is to collect valuable information from the input text. Generally, there are three key types of information to be extracted in RAF-AG: cybersecurity noun phrases, their corresponding cybersecurity labels, and the action relations among them. Cybersecurity noun phrases are identified from the text using SpaCy. To enhance the subsequent string comparison in our system, the obtained noun phrases are simplified by eliminating common words (e.g., "a", "an").

As noted in Section 2.3, the dependency relationship between words can serve to substitute domain-specific relations. To achieve this, we examine the sentence dependency tree produced by SpaCy to derive the action relations. Initially, we define the patterns for recognizing significant relations as grammar rules. The primary grammar rules employed and the extracted events are presented in Table 4.1. Each rule encompasses the part-of-speech tags (such as noun, pronoun, preposition, etc.) for the subject, the action's object, and the action itself [39]. As the sentence dependency tree is traversed, patterns will be matched against sub-trees to identify the expected relationships.

RAF-AG is capable of detecting cybersecurity events within a sentence by analyzing the identified noun phrases and their connections. Each cybersecurity event is represented as a (Subject, Verb, Object) tuple, where the subject and object are the noun phrases, and the verb signifies the action linking them. The position of the verbs within the sentence is also considered to indicate the chronological sequence of the events described. It is important to note that the term *verb* is used here in an expansive manner; thus, any word indicating the relationship between two phrases is treated as a verb in our implementation (including verbs and prepositions).

Table 4.1: RAF-AG's main grammar rules used for extracting important cybersecurity events.

Rule Type	Sentence Example	Extracted Event
Main verb	X **use** Y.	(X, use, Y)
Preposition	X use Y **in** Z.	(Y, in, Z)
Method	X use Y **via** K.	(Y, via, K)
Name	X use Y, **namely** F.	(Y, name, F)
Example	X use Y, **such as** A, B.	(Y, such as, A) (Y, such as, B)

4.2.1.3 Entity Labeling

Entity labeling in RAF-AG aims to classify phrases into different cybersecurity categories. In RAF-AG, the entity labeling function is developed using a Weak Supervision framework, namely Snorkel [40]. The detailed architecture of the entity labeling modules is presented in Figure 4.2. We analyzed recent cybersecurity research to define a set of available sources for building various labeling functions. The four types of labeling functions used in RAF-AG are:

1. **Keyword searching**: A collection of keywords is heuristically established (for example, "user" and "attacker"). For a given input phrase (such as "malicious attackers"), the root word (like "attackers") will be initially extracted and then matched against the keywords in the dictionary using the Levenshtein ratio [41]. If the similarity score exceeds 0.95, the labeling function will provide the appropriate label; if not, it will return "ABSTAIN".

2. **Special phrase matching**: For this category of labeling function, we initially gathered proper nouns (such as Windows, VPN, C2) from current cybersecurity NER datasets [42]. Additionally, phrases that have been directly annotated from the ATT&CK framework are also compiled. For instance, the Zebrocy software can be identified inside the ATT&CK data via: "[Zebrocy](.../software/S0251)". The approach outlined in keyword searching is utilized to establish the result of this labeling function.

3. **Pattern searching**: We use pattern search to detect IoC phrases. The regex patterns are collected from the text analysis research [43, 7] in cybersecurity.

4. **Existing NER model**: A NER model that has been trained on

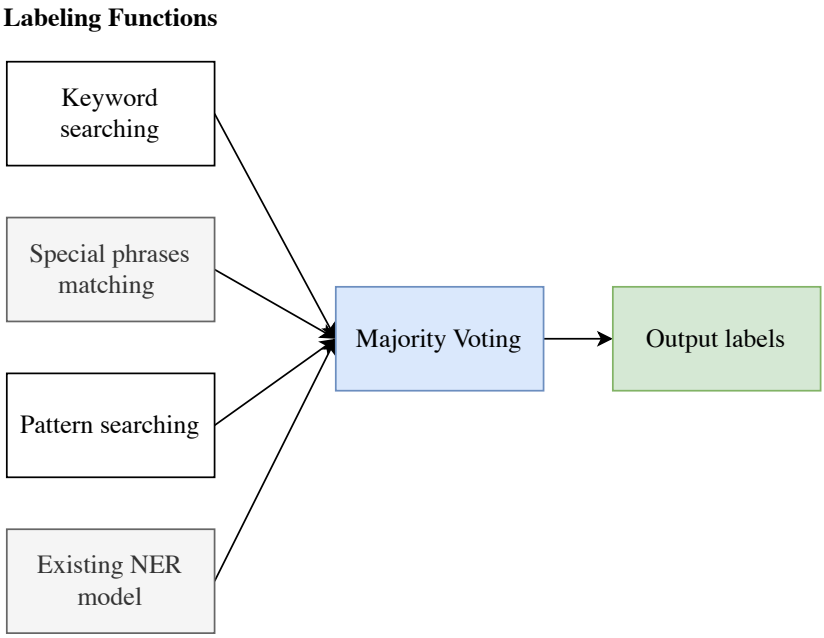


Figure 4.2: The architecture of the Weak Supervision-based entity labeling module.

the APT-NER dataset [42] is incorporated into this labeling function. APT-NER consists of annotations intended for named entity recognition related to advanced persistent threat (APT) texts. Because the APT-NER dataset employs a different set of labels, a conversion rule has been established to address this difference.

The keywords, unique phrases, and patterns are utilized to develop a dictionary structure encompassing ten primary categories. The statistics related to this dictionary are presented in Table 4.2. Each category contains sub-categories to further categorize phrases. For instance, the NETWORKING category can be divided into networking devices, networking protocols, IP addresses, websites, and email. When a specific phrase is input into the entity labeling function, the labeling function generates an initial list of possible labels. A majority voting mechanism is then employed to determine the most representative label from this list.

Once the cybersecurity events are identified, the entity labeling module categorizes both the subject and object of each event into cybersecurity categories. Cybersecurity events in which both the subject and object are marked as "OTHER" are deemed irrelevant and removed. At the end of the Information Retrieval (IR) stage, RAF-AG can compile a list of cybersecurity events

Table 4.2: Statistical information of the dictionary used in the Weak Supervision-based entity labeling.

Entity type	Description	Count of Phrases	Regex Patterns
DATA	data, information	854	Yes
DIRECTORY	directory, file path	205	Yes
ENCRYPTION	encryption, compression	301	Yes
FUNCTIONALITY	functions, malware, tools, etc.	2917	Yes
NETWORKING	networking, communicating	777	Yes
COMPONENT	cloud, container, OS, etc.	1018	No
REGISTRY/DLL	registry, dll	349	Yes
USER	user	76	No
VULNERABILITY	vulnerability	67	Yes
ATTACKER	attacker	170	No
OTHER	unclassifiable entities	215	No

along with the labels for each noun phrase associated with those events for a given input text. An example illustrating the input and output for the IR module component is depicted in Figure 4.3.

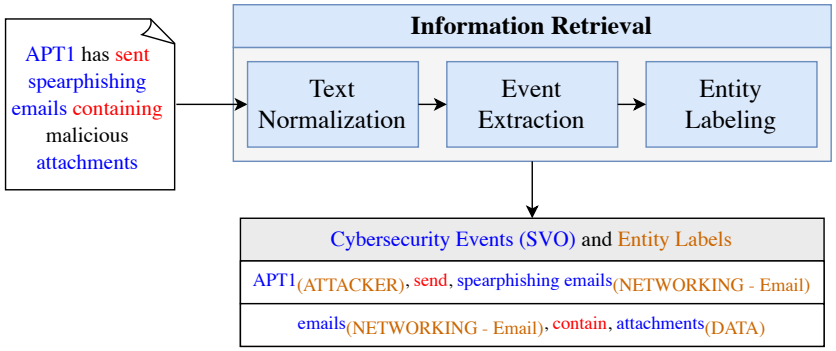


Figure 4.3: Input and output example of the Information Retrieval function.

4.2.2 Information Linking

There are two layers of information linking in RAF-AG. Initially, individual cybersecurity events are linked to create a graph based on the input text.

Next, we examine the graph and associate the extracted information with MITRE ATT&CK techniques through graph alignment. This section elaborates on both of these layers in detail.

4.2.2.1 Graph Building

In this module, we link distinct cybersecurity events for a particular text, resulting in a graph object. The graph's nodes represent the events' subjects and objects, whereas the edges are formed based on the verbs associated with those events. Figure 4.4 demonstrates the graph-building mechanism in RAF-AG via an example.

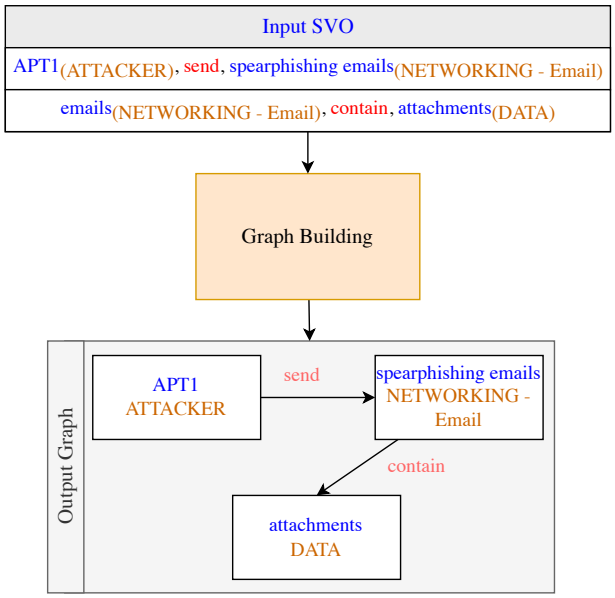


Figure 4.4: The RAF-AG's graph building functionality with only the text and label of a node presented.

Table 4.3 presents the detailed information contained within each graph node. Each graph node comprises seven fundamental pieces of information: the noun phrase, four distinct types of IDs, the label, and the verb. The various IDs enable RAF-AG to examine graph nodes from different perspectives, such as a phrase within the text or a node in the graph. They also retain valuable information necessary for further analysis, including the order of appearance of cybersecurity entities. Besides the verb information indicated by the graph's edges, we also include verb information within each node. The utilization of this additional verb information will be outlined in Section 4.2.2.4.

Table 4.3: The graph node's main information.

Element	Description	Example
Noun phrase	Extracted noun phrase	docx file
sentID	Identifier of sentence in the document	3
tokenID	Identifier of token in the sentence	5
nodeID	Absolute ID of the node (ID = sentID*1000 + tokenID)	3005
orderID	Sequential order of the node	2 (second node)
Label	Cybersecurity label	DATA
Verbs	Verb information	send

For each procedure example in the MITRE ATT&CK framework, RAF-AG will create a procedure graph. Since procedure examples frequently consist of brief sentences, these procedure graphs have a limited number of nodes and edges. Likewise, every CTI report provided will be converted into a report graph. The report graphs are more complicated than the procedure graphs because they are derived from lengthy documents.

We subsequently conduct co-reference handling and procedure deduplication to refine the obtained graphs.

Co-reference Handling Co-reference is an NLP concept in which two or more entities refer to the same person or thing. Co-reference is not originally included in the sentence dependency tree. RAF-AG uses the `coreferee` library [44], which is also integrated into SpaCy, to extract this important information. There are two strategies to handle co-reference depending on the type of the graph:

1. The compactness of the **procedure graph** can influence both the precision and execution time of RAF-AG's report analysis. Consequently, nodes containing phrases from the same coreference chain will be merged, resulting in a more compact procedure graph.

2. In the **report graph**, the sequence in which cybersecurity events appear is crucial for generating attack paths in RAF-AG. When distant nodes are merged, it can distort this information. Therefore, we implement co-reference resolution, where ambiguous pronouns (like "it" or "they") are substituted with their corresponding phrases (such as "attackers").

Procedure Deduplication As previously mentioned, ATT&CK data is formed based on the observation of actual incidents. For each incident, the way an attacker executes a technique is documented as a single procedure example. When a procedural pattern for a particular technique is observed across multiple cyber incidents, it is represented as similar procedure examples within that technique's data. For instance, the procedure examples for "admin@338 has sent emails with malicious Microsoft Office documents attached" and "APT1 has sent spearphishing emails containing malicious attachments" from the "T1566.001 Spearphishing Attachment" technique are alike. We categorize these closely related procedure examples as duplicates and perform procedure deduplication to eliminate this redundancy. To achieve this, we apply the graph alignment algorithm discussed in Section 4.2.2.2 on the procedure graphs of the technique. Procedure graphs that exhibit a high degree of similarity will be consolidated into a larger graph. Consequently, this approach decreased the number of procedure graphs in our dataset from 12456 to 10502.

4.2.2.2 Graph Alignment

In RAF-AG, a graph alignment method is utilized to identify the attack steps within the report graph and link them to ATT&CK information. Because the graphs are derived from textual content, the writing style of the author can influence the structure of the graph, even if the underlying information remains the same. This issue leads to a high rate of False Negatives (FN) when using precise graph alignment. To address this, we employ a fuzzy graph alignment approach, inspired by the method described in the AttacKG paper [6], to facilitate the analysis.

Algorithm 1 describes the fuzzy graph alignment with two graphs X and Y as inputs. In step 1, we first determine similar nodes from Y for each node of X. We then combine similar nodes into subgraphs of Y in step 2. Step 3 determines the best subgraph of Y that resembles graph X. We present these steps in more detail as follows:

- **Step 1**: We determine similar nodes from Y for each node $x_i \in X$ in this step from from line 6 to 13.

  The similarity scores of node $x \in X$ with respect to all nodes $y_j \in Y$ are determined via the *node_sim* function (see Section 4.2.2.3). The node similarity threshold is set to 0.8, which removes any node $y_j \in Y$ with a similarity score less than 0.8 from the analysis.

  The analysis results for all $x_i \in X$ define a two-dimensional array V. The first dimension (denoted by n) of this array is the number of X's

Algorithm 1 Fuzzy Graph Alignment

procedure ALIGN(X,Y)
    $X \leftarrow$ graph with n nodes of a procedure example
    $Y \leftarrow$ graph with m nodes of a CTI report
    $V \leftarrow \{\}$
    $t \leftarrow$ node similarity threshold of 0.8
    for each node $x_i \in X$ **do**
        $V[i] \leftarrow \{\}$
        for each node $y_j \in Y$ **do**
            if $node\_sim(x_i, y_j) \geq t$ **then**
                $V[i] \leftarrow V[i] \cup \{y_j\}$
            end if
        end for
    end for
    $\mathcal{S}(Y) \leftarrow n$-ary Cartesian product over n sets $V[i]$, with $1 \leq i \leq n$
    $M \leftarrow \{\}$
    for each $C \in \mathcal{S}(Y)$ **do**
        $score \leftarrow g\_score(X, C)$
        $M.add((C, score))$
    end for
    $C_{best} \leftarrow best(M)$
    return C_{best}
end procedure

nodes. Each entry $V[i] \in V$ is an array of candidate nodes from Y similar to node $x_i \in X$.

- **Step 2**: After inputting the two-dimensional array V, this step generates a set of subgraphs $\mathcal{S}(Y)$ by calculating the n-ary Cartesian product over n sets $V[1]$, ..., $V[n]$. Each subgraph in $\mathcal{S}(Y)$ is potentially similar to the graph X.

  Note that we use the n-ary Cartesian product over n sets from set theory. In the case of two sets A and B, the Cartesian product yields the set of all possible ordered pairs created by combining each element of A and B.

- **Step 3**: The third step (presented from line 16 to 20) determines the most similar subgraph of Y with respect to X. First, the graph alignment score is calculated using the g_score function (see Section 4.2.2.5) between X and each $C \in \mathcal{S}(Y)$. Subsequently, the subgraph C_{best} with the highest graph alignment score will be returned.

The Algorithm 1 time complexity is $\mathcal{O}(n^n)$. Two primary loops require the most computing resources in the alignment process. A $\mathcal{O}(n^2)$ complexity in the initial loop (line 6) where each node $x_i \in X$ will be compared with each node $y_j \in Y$. The computational cost in the subsequent loop (line 16) depends on the number of subgraphs generated by the n-ray Cartesian product. In an optimal scenario, a complexity of $\mathcal{O}(1)$ is obtained where each $x_i \in X$ can be matched with only one node $y \in Y$. In the worst-case scenario, where each $x_i \in X$ is similar to all nodes $y_j \in Y$, the Cartesian product yields m^n subgraphs, leading to a complexity of $\mathcal{O}(m^n)$ of the second loop. In a practical environment, n is usually small (≤ 5) because it is the number of nodes of the procedure graph. The procedure graph is built from the short paragraph. We set the node similarity threshold (step 1 of the algorithm) to a high value of 0.8 so that each $x_i \in X$ will only match a small set of nodes $y_j \in Y$. This will hopefully prevent the worst-case scenario from happening.

When executing RAF-AG, the input for the graph alignment algorithm, Algorithm 1, consists of the two graphs X and Y, representing the procedure graph and the report graph, respectively. The algorithm is subsequently employed to align all procedure graphs with the report graph.

To support the work of the graph alignment function, we implement three sub-functions: node similarity, edge similarity, and graph alignment score. We will present these sub-functions in the following sections.

4.2.2.3 Node Similarity

We present the node similarity for two nodes x and y in Equation 4.1. Only the node's label and the text phrase are considered, with a weighting parameter γ introduced to scale their importance.

$$node\_sim(x, y) = \gamma + (1 - \gamma) \cdot text\_sim \tag{4.1}$$

The γ parameter controls the significance of labels across various scenarios. When two nodes share the same label, we set $\gamma = 0.4$. Conversely, when the labels differ, we apply $\gamma = 0.3$ to highlight the significance of the text phrase. These values are derived from initial experiments we carried out using real node and edge data to establish the γ values (along with α and θ in Section 4.2.2.4).

The variable $text\_sim$ represents the similarity between two phrases, defined as the cosine similarity of their embedding vectors, $text\_sim(t_1, t_2) = cosin(emb_1, emb_2)$. We opted for cosine similarity as the primary measure for text comparison in RAF-AG due to its lower computational requirements compared to alternative methods (e.g., BertScore [45]). Cosine similarity provides strong performance in text comparisons, particularly in environments without GPU support. We utilize the Google Universal Sentence Encoder (USE) [46] to generate the embedding vectors for the two phrases.

4.2.2.4 Edge Similarity

The edge similarity between two edges, $e1(a_1, a_2)$ and $e2(b_1, b_2)$, is calculated using Equation 4.2. In this equation, a_1 and a_2 represent the vertices of edge $e1$, while b_1 and b_2 represent the vertices of edge $e2$. Nonetheless, we expand the definition of edges here, as two graph nodes do not have to be directly connected to constitute an edge.

$$edge\_sim(e1, e2) = \frac{\sqrt{node\_sim(a_1, b_1) \cdot node\_sim(a_2, b_2)}}{dist(a_1, a_2) \cdot dist(b_1, b_2)} \tag{4.2}$$

The distance between two nodes, $dist(i, j)$, is described in Equation 4.3, with $short\_dist(i, j)$ used to denote the shortest distance for two node i and j in the same graph, and $sent\_diff(i, j)$ denoting the difference in sentences.

$$dist(i, j) = min(short\_dist(i, j), sent\_diff(i, j)) \tag{4.3}$$

Each node includes a sentence ID (`sentID`) that indicates its corresponding sentence. The formula used to compute the sentence difference between two nodes is $sent\_diff(i, j) = 1 + \alpha \cdot abs(sentID_i - sentID_j)$. This approach

softens the calculation, moving away from simply using the absolute difference between two `sentIDs`. When nodes are part of the same sentence, the $sent\_diff(i, j)$ yields a minimum value of 1.0 instead of 0 to prevent the occurrence of a **Division by Zero** error. As the sentence difference increases by 1, the value of $sent\_diff(i, j)$ will rise by a factor of α, which is set to 0.3 in our experiments.

Verb Similarity We note that, in some cases, the action verbs hold greater importance than the cybersecurity noun phrases when determining a technique. For example, the action of "deleting" is more vital to the technique "T1070 Indicator Removal" than the particular items being deleted. RAF-AG integrates these verbs to improve performance using a reward mechanism defined by parameter θ, as described in Equation 4.4, which utilizes the geometric mean formula.

$$edge\_sim(e1, e2) = \sqrt{edge\_sim(e1, e2) \cdot \theta} \tag{4.4}$$

To achieve this, we compile a collection of 515 verbs that are initially gathered from the cybersecurity events found in the procedure examples of ATT&CK. These verbs are then categorized into 44 groups based on their semantic similarity within the cybersecurity context. For instance, "remove" and "delete" are classified together since both denote "the action of eliminating something". Furthermore, RAF-AG distinguishes between two categories of verb groups: normal and strong. Strong verb groups consist of verbs essential for recognizing techniques, including verbs related to deleting and masquerading.

When assessing the similarity between edges, the verb group information for each edge's verb is utilized. Let v_1 and v_2 represent the verbs of $e1(a_1, a_2)$ and $e2(b_1, b_2)$, respectively. VG_i signifies the i^{th} verb group; whereas $strong(VG_i)$ is a function that returns True if VG_i is classified as a strong verb group. The value of θ in Equation 4.4 varies based on the conditions of v_1 and v_2, as illustrated in Equation 4.5.

$$\theta = \begin{cases} 1.0 & \text{if } v_1 \in VG_i \quad \text{and} \quad v_2 \in VG_i \\ 0.3 & \text{if } v_1 \in VG_i \quad \text{and} \quad v_2 \in VG_j \text{ and } \quad i \neq j \\ & \text{and} \quad (strong(VG_i) \text{ or } strong(VG_j)) \\ 0.5 & \text{otherwise} \end{cases} \tag{4.5}$$

Note that edges formed from nodes that are disconnected or far apart lack verb information. In such instances, we utilize the verb information found in the object node of the edge for our computations. Specifically, for the edges $e1(a_1, a_2)$ and $e2(b_1, b_2)$, the object nodes are a_2 and b_2, respectively.

4.2.2.5 Graph Alignment Score

The alignment score between two graphs, X and C, is calculated using Equations 4.6, 4.7, and 4.8. In these formulas, m represents the number of edges in graph X, while n indicates the number of nodes. For graphs X and C to apply these equations, two conditions must be met. First, both X and C should contain the same number of nodes, n. Second, the node $x_i \in X$ must be highly similar to the corresponding node $c_i \in C$. In RAF-AG, these conditions are always fulfilled, as C is created through the n-ary Cartesian product of n sets $V[i]$, where $1 \leq i \leq n$.

In Equation 4.6, we compute the node-based similarity $g\_node(X, C)$ between two graphs by focusing solely on their nodes. The calculation of $g\_node(X, C)$ involves averaging the $node\_sim$ values for all pairs of nodes. Each pair consists of nodes from graphs X and C that share the same index. The $node\_sim$ values are derived through the node similarity metric outlined in Section 4.2.2.3.

$$g\_node(X, C) = \frac{1}{n} \sum_{\substack{x_i \in X \\ c_i \in C}} node\_sim(x_i, c_i) \tag{4.6}$$

The graph similarity score that focuses solely on the edges between two graphs, X and C, is represented by Equation 4.7. This value is calculated by averaging the edge similarity values, $edge\_sim$, for all pairs of edges, with $edge\_sim$ being determined through the metric outlined in Section 4.2.2.4. An edge pair consists of edge $e1(a_1, a_2) \in X$ and edge $e2(b_1, b_2) \in C$ if both $(a_1,\ b_1)$ and $(a_2,\ b_2)$ are recognized as node pairs. Moreover, edge $e1(a_1, a_2) \in X$ must be a genuine edge, indicating that it connects two nodes directly with $dist(a_1, a_2) = 1$.

$$g\_edge(X, C) = \frac{1}{m} \sum_{\substack{e_i \in X \\ e_j \in C}} edge\_sim(e_i, e_j) \tag{4.7}$$

Finally, in Equation 4.8, the overall graph alignment score, $g\_score(X, C)$, is calculated as the average of $g\_node(X, C)$ and $g\_edge(X, C)$.

$$g\_score(X, C) = \frac{1}{2}(g\_node(X, C) + g\_edge(X, C)) \tag{4.8}$$

4.2.3 Attack Path Generation

The method for aligning graphs between the procedure graph X and the report graph Y results in C_{best}, denoting the subgraph of Y that attains the

highest alignment score concerning X. This section describes the steps for identifying C_{best} in the report and how to construct the final attack path.

In RAF-AG, we assume that a report describes the attack steps sequentially. In cases where the events and methods in a report are not displayed in chronological order, possibly because of the type of attack reported, RAF-AG might not completely reflect this non-linear arrangement. Nevertheless, RAF-AG preserves the relationship between techniques and the information included in the text. This still enables cybersecurity analysts to use the attack paths generated by RAF-AG to support their manual examinations.

4.2.3.1 Alignment Localization

As illustrated in Table 4.3, every graph node possesses an `orderID` that signifies its sequence of appearance within the graph. To determine where C_{best} aligns within the report graph, the `orderID` values for each $c_i \in C_{best}$ will initially be gathered to create a list referred to as `orderIDs`.

We use two criteria for identifying the best `orderIDs` to be the alignment location, as follows:

1. For the *heuristic* criterion, we evaluate and rank the nodes in C_{best} to identify the most significant one. The method for ranking is outlined in Table 4.4. According to this table, every node in C_{best} will receive a score. The node with the highest score will serve as the representative node for C_{best}, and its `orderID` will indicate the position of C_{best}. In cases where multiple nodes share the same score, the *maximum* criterion will be utilized.

2. For the *maximum* criterion, the highest value in `orderIDs` will indicate the position of C_{best}.

Table 4.4: The ranking score for node importance. This score is used in finding the best alignment location.

Ranking score	Condition	Examples
4	VULNERABILITY or REGISTRY nodes	"CVE-2020-1789"
3	Nodes with strong verbs	"delete"
2	Proper noun nodes	"PowerShell"
1	Common noun nodes	"malicious files"
0	Other cases	

4.2.3.2 Attack Step Coordination

We initially present the idea of Graph Alignment Result (GAR) to signify the alignment outcome between a particular procedure graph and a report graph. Table 4.5 shows the GAR information. The technique ID found in the table is derived from a procedure-technique mapping, aiming to label attack steps with ATT&CK information.

To thoroughly analyze the report graph, the process of graph alignment is performed multiple times until all procedure graphs have been examined, leading to the creation of a collection of Graph Alignment Results (GARs). A hyperparameter, referred to as the graph alignment threshold, is utilized to eliminate any procedure graph that is not similar enough to the target report graph. For instance, if the graph alignment threshold is set at 0.9, any procedure graph with a graph alignment score (as mentioned in Section 4.2.2.5) of less than 0.9 in relation to the target report graph will be discarded. Consequently, the size of the GAR set can be managed through this parameter. It is important to note that the graph alignment threshold is distinct from the node similarity threshold described in Section 4.2.2.2. While the node similarity threshold identifies whether two nodes are alike, the graph alignment threshold assesses whether two graphs are similar.

The RAF-AG organizes GARs to reconstruct the sequence of attack steps. Once coordinated, each unique alignment location is recognized as an attack step paired with a technique ID. Furthermore, arranging these unique locations in a sequential list will create the attack path. As a result, the total length of the attack path corresponds to the number of distinct locations.

Table 4.5: The Graph Alignment Result (GAR) object's main information.

Name	Description	Examples
Basic information	Graph nodes and edges for the procedure graph	
Score	Graph alignment score	0.95
orderIDs	orderID values of all the nodes in the C_{best} graph	[4,6]
nodeIDs	nodeID values of all the nodes in the C_{best} graph	[1009,2003]
Alignment location	Location of the alignment	6
Procedure ID	ATT&CK procedure ID	rela...b5645
Technique ID	ATT&CK technique ID	T1041

The coordination of GARs aims to: (1) determine the representative GAR for a specific technique, and (2) determine the representative technique to label the attack step, as follows:

1. **Representative GAR of the technique** In the MITRE ATT&CK framework, a single technique can be associated with several examples of procedures (procedure graphs). When various procedure graphs related to the same technique are successfully matched to a report graph, we generate a collection of GARs that share the same technique ID. RAF-AG will assess these GARs to identify the most representative one for the technique.

   To achieve this, we begin by calculating the weighted arithmetic mean of the GAR distribution for a specific technique. The mean value is computed using Equation 4.9 when considering n GARs. The contribution of each GAR_i is determined by its alignment score, represented as w_i. Additionally, $value_i$ specifies the alignment location of GAR_i.

$$weighted\_mean = \frac{\sum_1^n value_i \cdot w_i}{\sum_1^n w_i} \tag{4.9}$$

   Once the weighted mean value is calculated, the GAR with the alignment location closest to the mean will be deemed the representative of that technique. Then, we use the graph alignment score of the representative GAR as the technique's confidence score.

   When several GARs exhibit identical distances to the mean, we assess their rank score to identify the top one; this is accomplished by dividing its alignment score by the standard deviation of the `orderID` values. The GAR that achieves the highest rank score will represent the technique. This approach prioritizes the GAR that demonstrates less dispersion of information compared to the others.

2. **Representative technique to label attack step** In cases where multiple techniques are found at the same location, we can select the one that has the highest alignment score to serve as the representative. Nevertheless, we have also introduced the hyperparameter k to permit a maximum of k techniques to symbolize an attack step. This hyperparameter k is defined by the user (with a default setting of 1). The introduction of k can provide advantages to users in real-world scenarios. For instance, when examining new CTI reports concerning emerging attack campaigns, users can opt for a higher k to expect a greater variety of potential techniques (as long as they possess sufficient resources to carry out the recommended mitigation).

4.3 Evaluation

In this section, we assess RAF-AG from various perspectives. First, we investigate the attack path employed by RAF-AG in a case study of the Frankenstein Campaign (see Section 4.3.1). Next, we analyze RAF-AG's performance using precision, recall, and F1 score metrics in Section 4.3.2. We explore the correlation between the resulting attack path and the graph alignment threshold to identify the ideal value for report evaluation. Finally, we examine the cost associated with cybersecurity expertise for RAF-AG during both the implementation and maintenance phases. We present these discussions in Appendix A.2 and A.3.1, respectively.

4.3.1 Frankenstein Campaign Case Study

In 2019, cybersecurity specialists reported an attack campaign referred to as "Frankenstein" [47]. The campaign earned its name from the iconic monster because the attackers utilized various unrelated open-source tools as weapons. This campaign has become a standard example for report analysis studies in cybersecurity. Therefore, for illustrative purposes, we provide the attack path output from RAF-AG for this campaign and detail the information used to identify each phase of the attack.

In Figure 4.5, we have presented the pertinent details from the CTI report "Frankenstein Campaign," [47], which describes the sequence of cybersecurity incidents during the campaign in chronological order. An expert not affiliated with this research has manually examined this campaign text to establish a ground truth attack path. The resultant attack path generated by RAF-AG is also included. Each attack step is marked with a related MITRE ATT&CK technique. When the information in the report is generic, we utilize tactic names like "TA0007 Discovery" to represent a tactical objective. Various colors are employed in the text displayed in the figure to emphasize how the information in the report aligns with the identified techniques.

Concerning the output attack path from RAF-AG, a graph alignment threshold of 0.87 has been utilized. This threshold was selected after analyzing the performance of RAF-AG across different threshold values, which is elaborated upon in Appendix A.2.

As we can see from Figure 4.5, the RAF-AG output has two main benefits:

1. **Sequential order**: The attack path produced by the RAF-AG largely preserves the sequence of information given in the text, with minor inaccuracies.

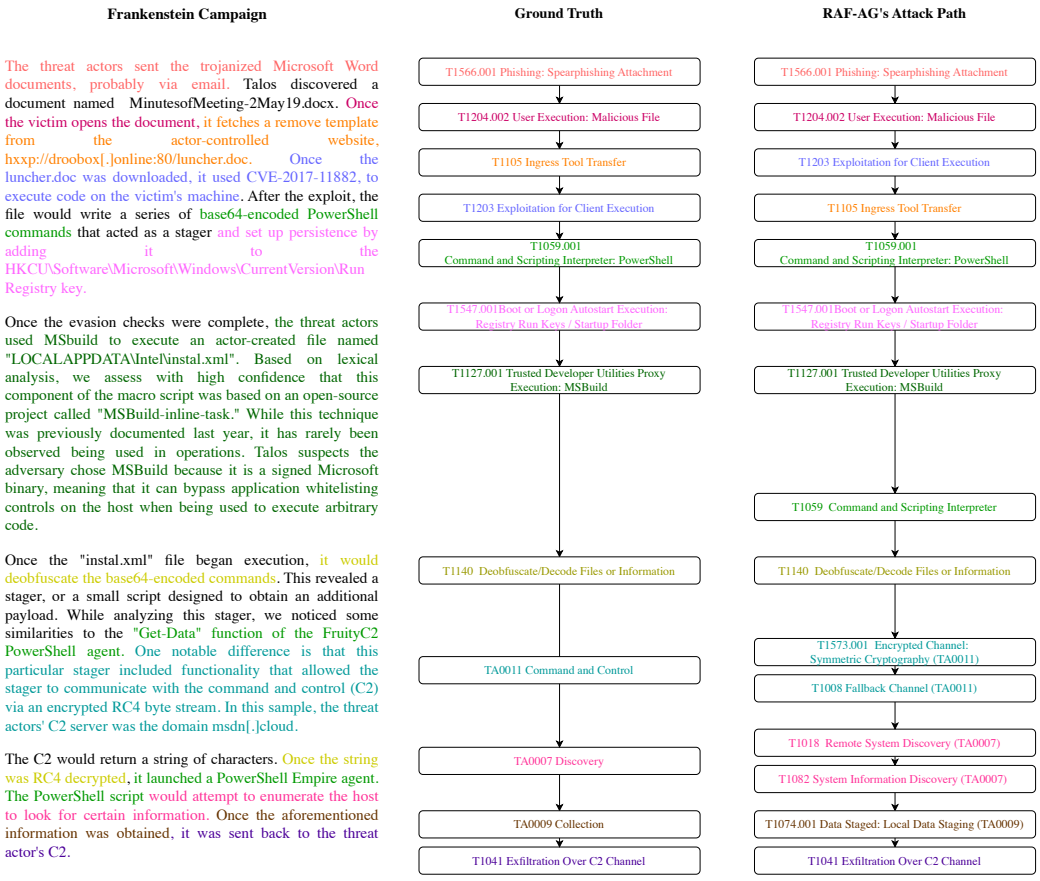


Figure 4.5: The generated attack path for the "Frankenstein Campaign" report by RAF-AG. The relationship between the placement of an attack step description in the document and the associated ATT&CK technique is demonstrated by text color. The sequential order of attack steps in the attack paths is shown from top to bottom.

2. **Sub-techniques**: RAF-AG is capable of identifying the attack path through sub-techniques like T1566.001, which refers to "Phishing: Spear Phishing Attachment." This enables a more effective response to incidents, since different mitigations may be needed for different sub-techniques.

The comprehensive attack path illustrated in Figure 4.6 indicates that various techniques require a specific blend of phrases and verbs for proper identification. This knowledge is explicitly encoded within the procedural examples of the techniques. By utilizing fuzzy graph alignment, RAF-AG can leverage mutated combinations of information to identify the technique. This is the case when the sequence of phrases and verbs in the procedural example does not adhere to the chronological sequence of information in the report.

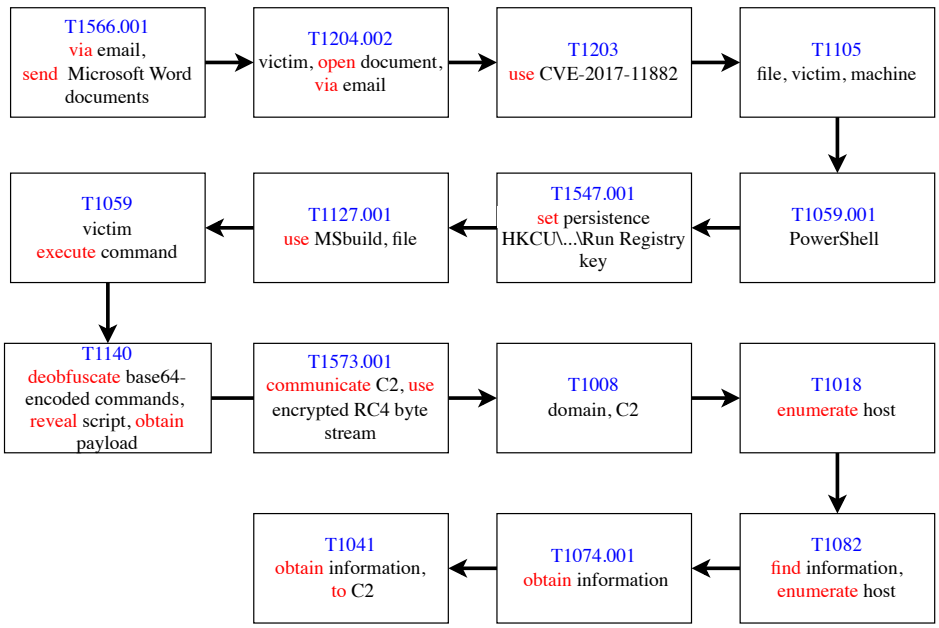


Figure 4.6: The Frankenstein campaign's attack steps generated by RAF-AG. Each box is an attack step. The blue color indicates the ATT&CK technique ID. Verbs and phrases (cybersecurity artifacts) are presented in red and black, respectively.

4.3.2 Attack Path Evaluation

To evaluate the performance of RAF-AG presented in this chapter, we set up ground truth data of 30 CTI reports from various websites/blogs and similar

research in cybersecurity:

- AttacKG paper [6]: Frankenstein Campaign, Firefox BITS Micro APT, Cobalt Campaign, DustySky Campaign, APT41 Campaign
- Talos Intelligence [48]: Konni malware, Smoking Guns - Smoke Loader, ObliqueRAT trojan, NavRAT trojan, Kimsuky APT, Emotet malware
- Bitdefender [49]: Md_client backdoor
- Malwarebytes [50]: Auto Stealer
- CISA [51]: Celas Trade Pro (Windows), Celas Trade Pro (macOS), JMT Trading (Windows), JMT Trading (macOS), Crypto Union (macOS)
- Threatpost [52]: Operation Sharpshooter
- The DFIR Report [53]: AHK keylogger, BazarLoader, BumbleBee malware
- ESET Research [54]: Operation Spalax, Operation Wocao
- McAfee Labs [55]: Operation HoneyBee
- Mandiant [56]: UNC5221 APT, SUGARDUMP - credential stealer, SUGARUSH backdoor
- BlackBerry Blog [57]: Costaricto Campaign
- Cyber Tzar [58]: Operation SnowMan

Initially, a cybersecurity expert who did not participate in this study performed a manual evaluation to establish the ground truth for the gathered reports. Subsequently, the output attack paths from RAF-AG and AttacKG [6] are compared with the established ground truth to compute the performance metrics (precision, recall, and F1 score). These performance metrics are selected because they are commonly used in report analysis research [11, 12, 6]. We select AttacKG for comparison because it has the capability to analyze CTI reports for ATT&CK technique IDs.

For this assessment, we examine the attack steps of the generated attack path and the ground-truth attack path to calculate the metrics. True Positive (TP) instances refer to techniques present in the ground truth that are also included in the output attack path. False Positive (FP) instances are techniques listed in the attack path but absent from the ground truth, while

False Negative (FN) instances are techniques found in the ground truth that the framework failed to identify.

We assess the performance of RAF-AG in comparison to AttacKG [6], which is a comparable framework designed for the automatic extraction of malicious techniques within CTI reports. The researchers behind AttacKG manually developed a private NER dataset to facilitate the entity extraction function. Initially, AttacKG examines MITRE ATT&CK's procedure examples and thousands of CTI reports, and compiles the results statistically to produce technique templates. Subsequently, these technique templates serve as the informational units when analyzing new reports. In the process of statistically compiling the results to create templates, they can obtain a confidence score of nodes/edges by utilizing the frequencies of entities and relations.

This mechanism prioritizes common procedural patterns (frequent entities and relationships) over less common ones within each technique template. To analyze a particular report, AttacKG performs graph alignment between the constructed graph for the report and each of the technique templates. However, the report graph is segmented into clusters of connected nodes. During the alignment process, the technique template can only look for information within each cluster to identify the technique.

The main distinctions between RAF-AG and AttacKG are as follows:

1. RAF-AG is not reliant on a manually created NER dataset during its information retrieval phase. Instead, it leverages existing data and methods to achieve comparable outcomes (such as entities and labels) from the input text. The manual creation of an NER dataset of AttacKG requires considerable expertise. However, the limited size of the NER dataset and the small model fine-tuned with this dataset restrict the generalizability of AttacKG. As a result, its performance may decrease when dealing with reports that fall outside the scope of the dataset.

2. RAF-AG eliminates the need for extensive processing of numerous CTI reports beforehand to create technique templates, as is necessary with AttacKG. Each time AttacKG requires an update, the creation of technique templates must be redone, which involves considerable effort to maintain the framework.

3. During the graph alignment, RAF-AG considers the whole report graph, without any boundaries for subgraph searching. This approach is more suitable for complex scenarios, where accurately defining search boundaries is challenging.

RAF-AG runs with its default settings to generate the attack paths, which are $k = 1$ and a graph alignment threshold of 0.87. For the results of AttacKG, we utilize its most recent source code [59] (incorporating the NER model and technique templates) with the default hyperparameters. The assessment was carried out on an Apple MacBook Pro with an M1 processor (3.2 GHz, eight cores) and 16 GB of RAM.

Table 4.6 shows the performance evaluation for RAF-AG and AttacKG. It can be observed that RAF-AG generally outperforms AttacKG, as follows:

1. RAF-AG outperforms AttacKG significantly in terms of precision, achieving scores of 0.717 and 0.337, respectively. The atomic operator that has a strong influence on both RAF-AG and AttacKG is string comparison. However, RAF-AG's string comparison utilizes Google's Universal Sentence Encoder [46] to calculate cosine similarity between texts, while AttacKG relies on the Levenshtein ratio. Although the Levenshtein ratio is applicable for string comparison, it falls short in capturing the semantic similarity or dissimilarity between phrases.

   Additionally, AttacKG creates technique templates by statically gathering procedure graphs. This approach gives precedence to frequently occurring procedural patterns in each template. Conversely, RAF-AG considers each procedure example as a distinct pattern of the technique. Consequently, RAF-AG results in a lower number of FP cases than AttacKG.

2. RAF-AG achieves a higher recall value (0.722) compared to AttacKG (0.535). AttacKG restricts the graph alignment within each connected component of the graph. When a technique needs information from multiple distinct graph components, the likelihood of recognition failure increases, resulting in a higher false negative rate. In RAF-AG, during the alignment of the graph concerning a particular report graph, the entire report graph is examined to detect a technique.

3. To highlight the RAF-AG performance concerning the F1 score (which represents the harmonic mean of precision and recall), we have also featured it in the table. Overall, RAF-AG achieves a satisfactory average F1 score (0.708), surpassing the score for AttacKG (0.393).

In terms of time efficiency, RAF-AG averages 25.5 seconds to analyze a report. The average time required for text pre-processing is 2.3 seconds. The time taken for event extraction is 0.74 seconds, while entity labeling takes 0.543 seconds. The time cost for graph building is 0.00023 seconds, and the

graph alignment process takes 13.3 seconds. Additionally, other time costs, including I/O operations, amount to 4.88 seconds. And finally, the process to generate attack paths requires 3.8 seconds.

We present a breakdown of runtime for each CTI report in Figure 4.7. Each report exhibits varying times for each module, with the majority of costs occurring during the graph alignment process. This is expected, as the graph alignment algorithm is executed repeatedly for every procedure graph to analyze the target report graphs. The presented time cost demonstrates RAF-AG's ability to effectively manage a significant number of generated reports each day. In contrast, AttacKG has an average report processing time of 18.56 seconds. The time cost required by AttacKG is lower than that of RAF-AG because:

- In AttacKG's graph alignment, the search boundaries are limited to each connected component of the report graph, whereas RAF-AG considers the entire graph of the target report. Consequently, RAF-AG can generate more sub-graphs from the report graph, but it takes more time to analyze them.

- To analyze a CTI report, AttacKG uses technique templates as the information units instead of procedure graphs, as done in the case of RAF-AG. Since the number of technique templates is about ten times smaller than the number of procedure graphs, AttacKG's approach can significantly reduce the time required for graph alignment. However, to generate these technique templates, AttacKG must analyze thousands of CTI reports in advance.

Table 4.6: The performance comparison between RAF-AG and AttacKG using 30 CTI reports.

CTI report	RAF-AG						AttacKG					
	TP	FP	FN	Precision	Recall	F1	TP	FP	FN	Precision	Recall	F1
AHK keylogger	9	4	3	0.692	0.75	0.72	9	7	3	0.562	0.75	0.643
APT41 Campaign	4	2	2	0.667	0.667	0.667	4	11	2	0.267	0.667	0.381
Auto stealer	6	4	1	0.6	0.857	0.706	5	11	2	0.312	0.714	0.434
BazarLoader	10	5	3	0.667	0.769	0.714	8	6	5	0.571	0.615	0.592
BumbleBee malware	6	1	1	0.857	0.857	0.857	4	7	3	0.364	0.571	0.445
Celas Trade Pro (macOS)	4	2	3	0.667	0.571	0.615	3	10	5	0.231	0.375	0.286
Celas Trade Pro (Windows)	6	2	2	0.75	0.75	0.75	3	5	4	0.375	0.429	0.4
Cobalt Campaign	6	2	2	0.75	0.75	0.75	5	9	3	0.357	0.625	0.454
Costaricto Campaign	6	1	3	0.857	0.667	0.75	2	1	7	0.667	0.222	0.333
Crypto Union (macOS)	4	4	2	0.5	0.667	0.572	2	13	4	0.133	0.333	0.19
DustySky Campaign	3	3	2	0.5	0.6	0.545	4	15	1	0.211	0.8	0.334
Emotet malware	5	2	1	0.714	0.833	0.769	3	9	3	0.25	0.5	0.333
Firefox BITS Micro APT	2	0	3	1	0.4	0.571	2	6	3	0.25	0.4	0.308
Frankenstein Campaign	12	0	0	1	1	1	8	5	4	0.615	0.667	0.64
JMT Trading (Windows)	4	3	2	0.571	0.667	0.615	3	10	3	0.231	0.5	0.316
JMT Trading (macOS)	3	3	3	0.5	0.5	0.5	3	11	3	0.214	0.5	0.3
Kimsuky APT	7	2	1	0.778	0.875	0.824	2	7	6	0.222	0.25	0.235
Konni malware	3	2	1	0.6	0.75	0.667	2	13	2	0.133	0.5	0.21
Md_client backdoor	5	0	1	1	0.833	0.909	2	5	4	0.286	0.333	0.308
NavRAT trojan	6	4	4	0.6	0.6	0.6	7	8	3	0.467	0.7	0.56
ObliqueRAT trojan	4	1	2	0.8	0.667	0.727	4	9	2	0.308	0.667	0.421
Operation Sharpshooter	4	1	0	0.8	1	0.889	2	8	2	0.2	0.5	0.286

Operation SnowMan	6	1	3	0.857	0.667	0.75	6	7	3	0.462	0.667	0.546
Operation Spalax	3	4	2	0.429	0.6	0.5	3	8	2	0.273	0.6	0.375
Operation Wocao	9	3	1	0.75	0.9	0.818	5	9	5	0.357	0.5	0.417
SUGARDUMP - credential stealer	5	1	2	0.833	0.714	0.769	2	4	5	0.333	0.286	0.308
SUGARUSH backdoor	5	3	2	0.625	0.714	0.667	3	4	4	0.429	0.429	0.429
Smoking Guns - Smoke Loader	2	6	2	0.25	0.5	0.333	4	9	0	0.308	1	0.471
SolarWinds Compromise	10	1	1	0.909	0.909	0.909	5	9	6	0.357	0.455	0.4
UNC5221 APT	5	0	3	1	0.625	0.769	4	7	4	0.364	0.5	0.421
Average				0.717	0.722	0.708				0.337	0.535	0.393

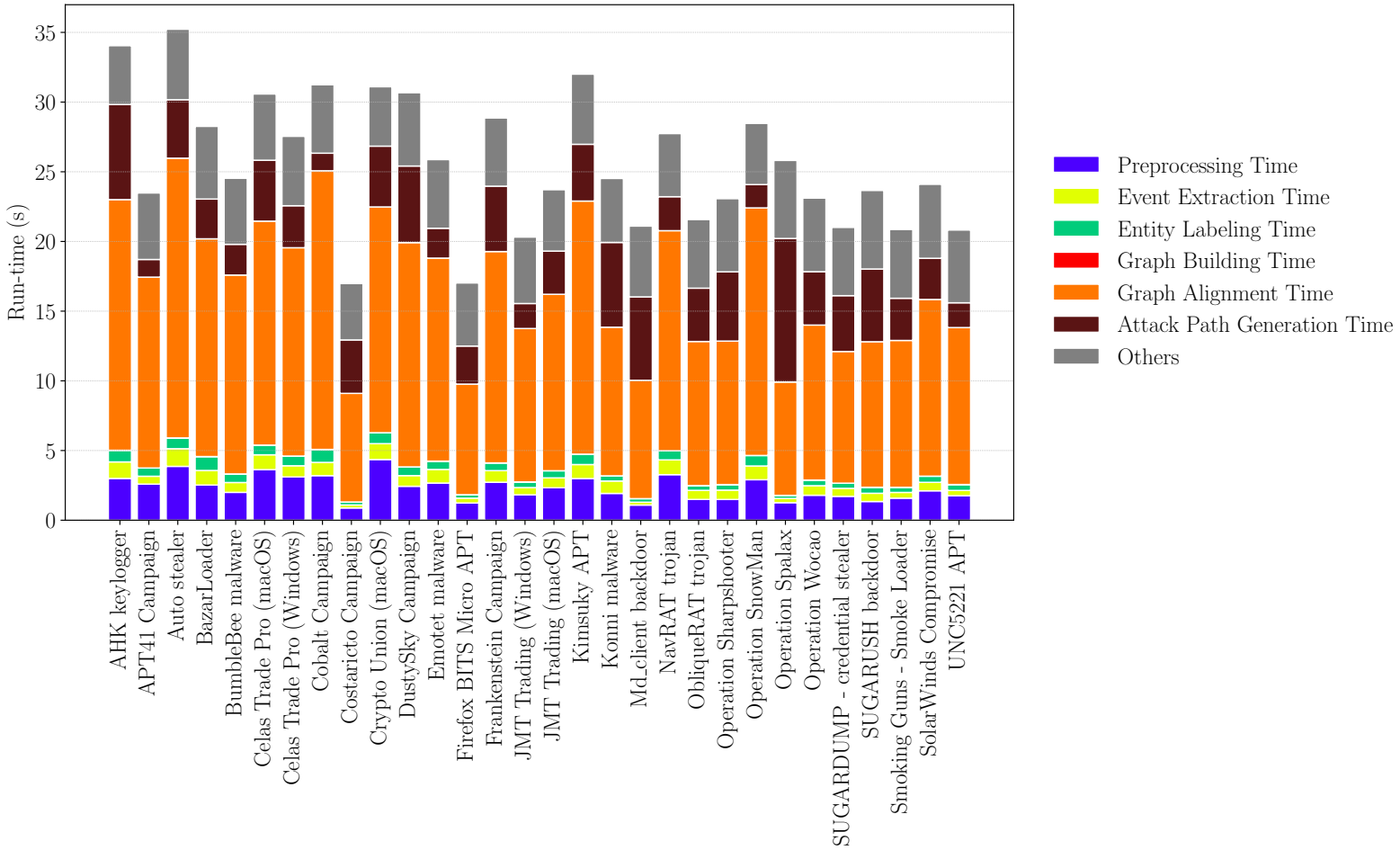


Figure 4.7: Time cost breakdown for running analysis of CTI reports.

4.4 Summary

In this chapter, we introduced RAF-AG, a framework that takes CTI reports as inputs and outputs high-quality attack paths. The attack steps include information about the corresponding MITRE ATT&CK techniques and sub-techniques. We maintain the order of the attack steps to reflect the chronological order of cybersecurity events in the report. Following our evaluation of reports related to 30 CTI reports from diverse sources, we concluded that RAF-AG surpasses the capabilities of a comparable report analysis framework, AttacKG. The average values for precision, recall, and F1 for RAF-AG are superior to those for AttacKG, with figures of 0.717, 0.722, and 0.708 in contrast to 0.337, 0.535, and 0.393, respectively.

Furthermore, in line with the main theme of this thesis, we employ a Weak Supervision-based approach to tackle the problem of insufficient labeled data. The use of high-performance NLP tools, such as SpaCy and the Google Universal Sentence Encoder, enables our framework to address the annotated data insufficiency problem more effectively.

For future work, we intend to tackle certain limitations identified. First, RAF-AG presumes that the steps of an attack in a CTI report are listed in a chronological sequence. Second, the coordination of attack steps described in Section 4.2.3.2 restricts each technique to appearing only once in the attack pathway. Exploring new methods that account for non-linear attack paths and multiple occurrences of the same techniques could enhance the framework for practical application. Furthermore, we will consider integrating RAF-AG into cybersecurity systems such as Security Information and Event Management (SIEM) and incorporating LLMs into the framework.

Discussion As potential additional applications of the framework discussed in this chapter, we highlight that researchers might use RAF-AG to investigate the causal relationships between techniques. This is particularly valuable for automating the threat modeling process. The capability to forecast the most likely subsequent attack techniques after some have been identified is also advantageous for developing secure network design systems. Furthermore, RAF-AG can be employed to generate the initial training dataset necessary to fine-tune these LLMs for tasks related to CTI report analysis. This approach will be described in more detail in Chapter 7.

Recently, large transformer-based models comprising billions of parameters, referred to as LLMs, have demonstrated an impressive ability to generalize in text processing. Therefore, LLMs can handle noisy input text to derive useful information without extensive text normalization, as seen in RAF-AG. However, we have not incorporated LLMs into RAF-AG yet due to various

drawbacks, such as the necessity for high-quality fine-tuning datasets, significant computing expenditures, and issues with non-determinism. Specifically, it is essential to custom fine-tune these models, particularly those with fewer than 30 billion parameters, to achieve the required capability. Creating a fine-tuning dataset for LLMs from scratch is a challenging and error-prone task that demands substantial expert input. Additionally, the GPU resource requirements for working with LLMs can be a major concern. Furthermore, the inherent non-deterministic nature of LLMs can lead to varying outputs for the same input. Considering the usage of LLMs in cybersecurity, we utilize LLMs in Chapter 5 for a different downstream task, namely policy generation. From this, a framework for domain-adaptive pre-training of LLMs (CyLLM-DAP) into cybersecurity is also introduced in Chapter 6. We then utilize both RAF-AG and CyLLM-DAP for report analysis based on open-source LLMs (see Chapter 7).

Chapter 5

Commercial LLM-based ABAC Policy Generation

This chapter introduces a framework for generating Attribute-Based Access Control (ABAC) policies with the help of LLMs.

5.1 Problem Definition

With its adaptability and detail-oriented nature, Attribute-Based Access Control (ABAC) serves as an appropriate access control framework for intricate and ever-changing IT environments. A conventional ABAC deployment starts with the identification of system attributes and the formulation of access control policies. Numerous studies [15, 16, 60] on ABAC seek to automate these time-consuming and error-prone processes using computer-driven models. Training computer models for policy creation from textual data in a supervised manner requires annotating thousands of sentences, posing a significant challenge. Additionally, applying a pre-trained model to new systems often demands re-training with fresh data, which may not always be readily available.

As already mentioned in the discussion of Chapter 4, LLMs are models with strong language capacity, presenting new opportunities to tackle the issue of data scarcity. Their generalizability also allows us to get rid of cumbersome text normalization. This chapter proposes a solution based on LLMs for generating fine-grained ABAC policies for IT networks. Our method addresses challenges related to LLMs, such as insufficient context and length restrictions. The method employs several LLMs in a mixture-of-agents mechanism to examine the ABAC situation from various viewpoints. We integrate multi-turn interactions and retrieval-augmented generation (RAG)

to create and furnish the necessary context for prompting LLMs. For our assessment, we perform experiments within an ICS network, ensuring that the ABAC policies conform to specific security standards. We investigate the possibility of directly using policies produced by LLMs in the process of making access control decisions. Utilizing ground truth data, we develop an optimization module that adjusts the priority values of these policies, leading to an outstanding F1 score of 0.994. This demonstrates that LLMs can effectively produce fine-grained ABAC policies for actual IT networks.

5.2 Proposed Approach

This section describes the proposed methodology for LLM-based fine-grained ABAC policy generation as shown in Figure 5.1, with five main components:

1. **Knowledge Base Construction**: This component constructs a knowledge base from initial data to support the following tasks.

2. **Prompt Construction**: This component aims to create a task-specific prompt to provide LLM with sufficient context.

3. **Attribute Refinement**: This component aims to refine attributes from a list of initial attributes.

4. **Policy Generation**: This component utilizes multiple LLMs as generators to generate policies aligned with security guidelines.

5. **Priority Optimization**: This optional component aims to optimize the policy priority values generated by LLMs for policy conflict resolution.

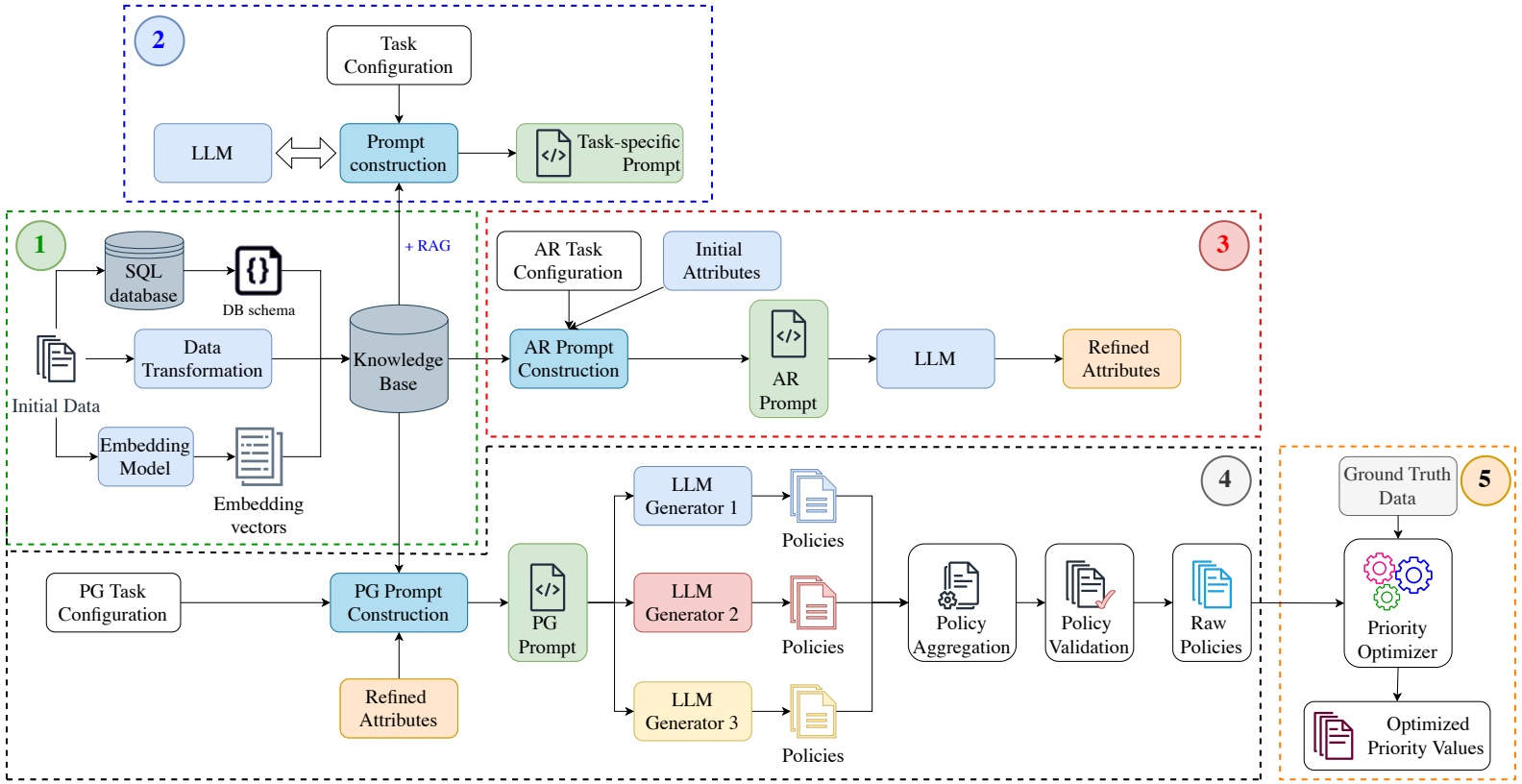


Figure 5.1: System architecture with main components: 1. Knowledge Base Construction, 2. Prompt Construction, 3. Attribute Refinement (AR), 4. Policy Generation (PG) and 5. Priority Optimization.

5.2.1 Knowledge Base Construction

In order to equip LLMs with the necessary context for addressing complex tasks, we established a knowledge base for the collection and organization of data for later analysis. Table 5.1 illustrates an example of a knowledge base utilized for policy generation in the ICS network. Each data item consists of three key components: name, description, and content. Various functions have been created to convert the original data (provided by the user) into a format suitable for LLMs that contains valuable information for future tasks, including:

Table 5.1: Main data elements in the knowledge base for ICS network and their relationship with specific tasks.

Name	**Data Type**	**Attribute Refinement**	**Policy Generation**
Introduction of ICS concept	text	✓	✓
A brief introduction of NIST SP 800-82 r2	text	✓	✓
ABAC's related concepts	text	✓	✓
NIST SP 800-82 r2 specific guidelines	list (JSON file)		✓
The ICS network overview	text	✓	✓
System information (devices, protocols, etc.)	dataframe (CSV files)	✓	✓
System initial attributes	list (JSON file)	✓	
Task description and template, rules	text	✓	✓
Task few-shot examples (for few-shot learning)	text	✓	✓
Database schema	text	✓	
Refined attributes	list (JSON file)		✓

1. We divide the security guideline document (such as the NIST SP 800-82 r2) into smaller sections and paragraphs to prevent issues related to context length limits and to minimize the task's complexity.

2. System information is converted into an SQL database, followed by the creation of a database schema. The database schema is used as input for the prompt in place of the system information (see Section 5.2.3).

3. An example list of access requests is created based on the system information. Each access request comprises a series of attributes along with their designated values, which can be referenced in Section 5.2.4.

4. For each data item, we create embedding vectors for the names and descriptions to support later RAG.

5.2.2 Prompt Construction

As illustrated in the blue dashed box of Figure 5.1, constructing the prompt necessitates a task configuration and involves interaction with both the knowledge base and the LLM to create a structured, task-specific prompt in a multi-turn format. Figure 5.2 depicts that the prompt begins with a system message designed to leverage the persona pattern [61], instructing the LLM to function as an expert in ICS and ABAC. The main body of the prompt consists of several chat turns; each turn can either be a knowledge-recalling prompt (aimed at retrieving frequently encountered knowledge types, such as ICS and ABAC) or a knowledge-injecting prompt (focused on incorporating new knowledge). The prompt concludes with a message that triggers the task, signaling the LLM to commence its work. Moreover, a specialized prompt (utilizing the flipped interaction pattern [61]) allows the LLM to proactively seek out new knowledge. We employ a similarity search technique to retrieve this knowledge from the database in a manner similar to RAG, facilitating its appropriate integration into the chat session.

5.2.3 Attribute Refinement

The workflow example for attribute refinement is illustrated in the red dashed box of Figure 5.1. The necessary data for this process (refer to Table 5.1) is included in a task-specific configuration and forwarded to the prompt construction phase. We begin with a fundamental list of attributes that provides minimal information, such as names and descriptions. Next, we utilize an LLM to refine these attributes. To avoid exceeding the context length limit, we intentionally do not use the system's detailed information and instead use the database schema (as discussed in Section 5.2.1). This step requires the LLMs to improve the attributes by generating more useful information and SQL SELECT commands (as depicted in Figure 5.3). Subsequently, we execute the SQL command to retrieve the valid values associated with the attribute from the SQL database. The outcome will be a set of refined attributes ready for subsequent analysis.

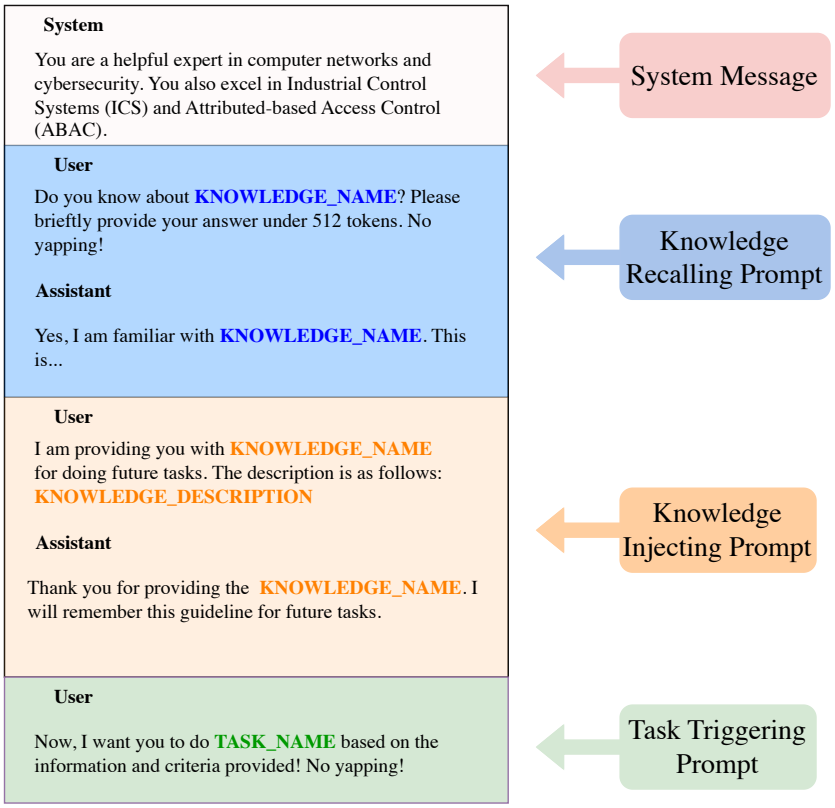


Figure 5.2: Main structure of a chat session.

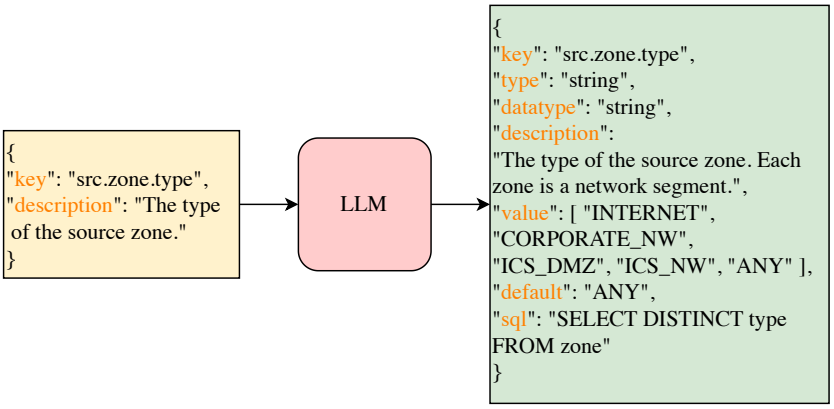


Figure 5.3: Input and output for attribute refinement.

5.2.4 Policy Generation

A policy generation workflow can be observed in the black dashed box of Figure 5.1. The key information for this policy generation procedure is the security guideline along with the system attributes, which have been further refined through attribute refinement. Like the attribute refinement process, all necessary information for this function (refer to Table 5.1) is initially compiled into a specific task configuration before it is sent to prompt construction to formulate an appropriate prompt.

Due to the complexity of this task, we create a list of generation guidelines to assist LLMs in producing ABAC policies in a specified format (e.g., Python function with docstrings). LLMs are also tasked with generating the priority values for the policies based on the guidelines and their reasoning. This proactive approach is intended since we plan to implement a priority-based algorithm to resolve policy conflicts. The example policy generated, which adheres to the specified criteria and rules, is illustrated in Figure 5.4. As shown in this figure, the policies are formatted as Python functions, where each condition either prohibits or permits access. The policy function defaults to return "None" in situations where there is insufficient information to make a decision. These policies can be directly imported into a Python runtime environment to enhance access control decision-making.

```
def SP800_82_51_Rule_No0_1(effect, policy, dictdat):
    """
    **Fact**: Network segmentation and segregation is one of the
most effective architectural concepts to protect ICS.
    **Reasoning**: This policy function checks if the source and
destination zones are different and if the service is essential for
cross-domain communication. If not, deny access.
    **Condition 1**: If `service.essential` is False and the source
and destination zones are different, deny access.
    **Default**: None
    **External Knowledge**: None
    **New Attribute Requirement**: None
    **Attributes**: `service.essential`, `src.zone.type`,
`dst.zone.type`
    **Potential Error**: None
    **Priority Value**: 70
    """
    if (dictdat.get("service.essential") == 0 and
        dictdat.get("src.zone.type") != dictdat.get("dst.zone.type")):
        return "deny"
    return None
```

Figure 5.4: An example of the generated policy.

Mixture of Agents The mixture-of-agents approach [62] enables several LLMs to collaborate in addressing a complex task. Drawing inspiration from this concept, we have created a similar solution that integrates multiple LLMs into the policy generation process, which is illustrated in Algorithm 2. The function `PROMPT_CONSTRUCTION()` is designed to create a prompt specifically for the policy generation task (refer to Section 5.2.2). Each LLM generator is then tasked with producing fine-grained policies. Once the generators complete their tasks, the outcomes are collected and sent to a subsequent aggregation phase.

Algorithm 2 Policy Generation

```
1: procedure GeneratePolicies(T)
2:     G ← list of LLM generators
3:     t ← threshold value
4:     F ← {} /*empty frequency dictionary*/
5:     P ← [] /*empty policy list*/
6:     S ← prompt_construction(T)
7:     for each generator in G do
8:         Prompt the generator with S
9:         Extract the policies from the response
10:        Deduplicate the policies
11:        for each policy p in generated policies do
12:            if p is in F then
13:                Increase count for p by 1
14:            else
15:                Add p into F with count 1
16:            end if
17:        end for
18:    end for
19:    for each policy p in F do
20:        if (p.count/len(G)) ≥ t then
21:            Add p to P
22:        end if
23:    end for
24:    return P
25: end procedure
```

Policy Aggregation This module is designed to integrate policies generated by various LLM generators into one consolidated list. In the initial

mixture-of-agents approach [62], the aggregator that synthesizes responses from different LLM generators is also an LLM itself. In our method, we employ a deterministic aggregator that uses a major voting mechanism (as outlined in lines 12 to 25 of Algorithm 2) to decide on the output policies based on the responses from the generators. We keep a frequency dictionary to count the votes from the generators concerning a particular policy. When a new policy is introduced, its frequency starts at one and will increment by one whenever a generator creates a similar policy. We utilize an equivalent operator to eliminate duplicate policies and assess policy similarity (as indicated in line 10 and line 13 of the algorithm, respectively).

Policy Equivalence Since the policies are formatted as Python functions, we initially assess the similarity of their abstract syntax tree (AST) by utilizing the Python library called `code_diff` [63]. Furthermore, we evaluate the outcomes of both policies with respect to the list of possible access requests. Two policies are deemed equivalent if their outcomes (such as allow, deny, None) are the same for every access request in the list. This characteristic is maintained by assigning a threshold value of 1.

Policy Validation As illustrated in Figure 5.4, every output policy is represented as a Python function accompanied by a docstring. The docstring gives us valuable information to verify the policy. We develop several functions to assess these generated policies in both deterministic and non-deterministic ways, as detailed below.

1. Deterministic validation: We utilize the Bandit [64] library to detect common security vulnerabilities in the produced code.

2. Non-deterministic validation: We utilize LLMs to analyze the relationships among various components of each policy to identify inconsistencies and mistakes.

5.2.5 Priority Optimization

Utilizing fine-grained ABAC policies generated by LLMs for access control can result in conflicts between policies. To resolve these conflicts, we implement a priority-based combining algorithm due to its explainability, flexibility, and scalability. The process for optimizing priorities is depicted within the orange dashed box in Figure 5.1. We presume access to ground truth data to refine the priority values of the generated policies, which is formulated as a continuous mathematical problem to determine the best solution among various options.

Solution Space We define the solution space as $S \subseteq \mathbb{R}^n$. Each solution, formulated as $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ where $0 \leq s_i \leq 100$ for $i = 1, 2, \ldots, n$, is a list of n priority values where each value is a real number from 0 to 100. Here s_i is the priority value for the policy p_i in a list of policies $\mathbf{p} = (p_1, p_2, \ldots, p_n)$.

Objective Function The goal of priority optimization is represented by the objective function $f : S \to \mathbb{R}$, where the function $f(s)$ yields a real number when provided with a solution $s \in S$. Specifically, the objective function $f(s)$ outputs the F1 score by comparing the access control decisions derived from the policies with priority values (using a priority-based combining algorithm) against the ground truth decisions (refer to Section 5.3.1).

We define the objective function as a black-box function, meaning that while we can observe or measure the output corresponding to a specific input, the direct relationship between inputs and outputs is either unclear or too complex to accurately model. This paper examines the performance of priority optimization by utilizing the optimizers available in Nevergrad library [65], such as CMA, DE, TBPSA, and others.

Priority Optimization Problem Using the above definitions, we can mathematically present our priority optimization problem as in Equation (5.1). We have a list of generated policies denoted as $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. The goal of the priority optimization is to determine a list of priority values $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ that maximizes access control performance. Each priority value s_i with $0 \leq s_i \leq 100$ corresponds to a specific policy p_i. The access control performance is measured using the F1 score, which evaluates how the decisions made by the ABAC policies with these priority values align with the expected decisions.

$$\begin{aligned} \text{Maximize} \quad & f(\mathbf{s}) \\ \text{Subject to} \quad & 0 \leq s_i \leq 100, \quad i = 1, 2, \ldots, n \\ \text{with} \quad & \mathbf{s} = (s_1, s_2, \ldots, s_n) \end{aligned} \tag{5.1}$$

5.3 Experimental Evaluation

This section describes the experiments carried out to evaluate the LLM-based policy generation method using an ICS network as our running example.

5.3.1 Ground Truth for the Running Example

For the typical ICS network depicted in Figure 5.5, we produce a list of 26616 access requests by generating valid combinations of system attributes with their corresponding valid values. To establish the access control decisions for the generated access requests, we work alongside cybersecurity experts who have in-depth knowledge of both the ICS environment and ABAC to identify 17 guidelines from the NIST guidelines. These guidelines are then used to manually generate ABAC policies. To enhance these expert-generated policies, the experts also formulate qualitative intentions—a new concept introduced in the QI-ABAC paper [20]. Both the expert-generated policies and qualitative intentions are utilized to develop a neural network-based classifier. This classifier can produce "allow" or "deny" decisions for incoming access requests. Ultimately, the decisions rendered by the trained classifier are regarded as ground truth. Consequently, there are 18815 "allow" decisions and 7,801 "deny" decisions within the ground truth dataset.

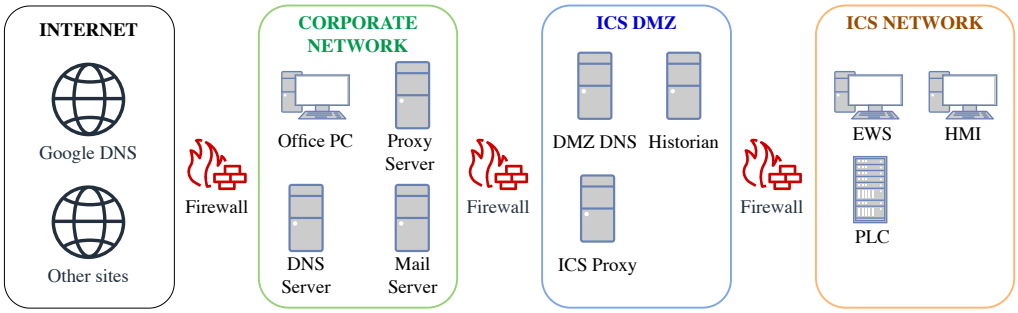


Figure 5.5: Typical ICS network with main network segments and devices that is used as a running example in our evaluation.

5.3.2 Preliminary Evaluation

In this section, we assess the practicality of employing all fine-grained ABAC policies created by LLMs for making decisions at a Policy Decision Point (PDP). A total of 181 policies have been generated by LLMs.

For each access request in the ground truth data, we record the decisions made by each policy from the generated list. Since there may be conflicting decisions among the policies, we apply various policy-combining algorithms (**Deny-overrides**, **Allow-overrides**, **Priority-based**, and **Weight-based**) to reconcile these decisions and produce a final decision ("allow" or "deny") for each access request. This step is repeated for all access requests in the ground truth data. It is important to note that the priority values generated

by the LLMs are used as weights in the weight-based combining algorithm. We then compare the output decisions with the expected decisions (ground truth decisions) to calculate the F1 score. We refer to this F1 score as the access control performance of the generated policies. In this context, we derive the metrics through binary classification, treating "allow" decisions as positive.

The results presented in Table 5.2 demonstrate that the overall performance is unsatisfactory. While the allow-overrides algorithm outperforms the other algorithms with an F1 score of 0.826, the high number of positive cases in the ground truth data raises doubts about the correctness of these output metrics.

Table 5.2: Preliminary performance of LLM-generated policies in access control decision-making.

Combining Algorithm	**TP**	**FP**	**TN**	**FN**	**Precision**	**Recall**	**F1**
Deny-overrides	0	0	7801	18815	0	0	0.0
Allow-overrides	18729	7801	0	86	0.706	0.995	0.826
Priority-based	127	0	7801	18688	1	0.007	0.013
Weight-based	16	0	7801	18799	1	0.001	0.002

Our evaluation highlights multiple factors contributing to these unsatisfactory outcomes:

1. ABAC policies are subject to conflicts. The problem becomes more severe when the number of generated policies is very large. In our evaluation, the LLMs can generate 181 ABAC policies.

2. The security guidelines are generic which suggest various security measures (such as whitelisting and blacklisting) applicable to various systems. In a particular scenario, these guidelines can cause conflicts.

3. The NIST security guidelines are inherently more restrictive, favoring "deny" policies instead of "allow" ones. However, in the current evaluation, the ground truth data contains a large number of access requests that should be allowed.

4. To accommodate the limitations in context length, we have divided the security guideline document into smaller sections. LLMs can only address one guideline or section at a time when generating ABAC policies. LLMs do not have a comprehensive view, which hinders their ability to generate correct policy priority values.

5.3.3 Priority Optimization Evaluation

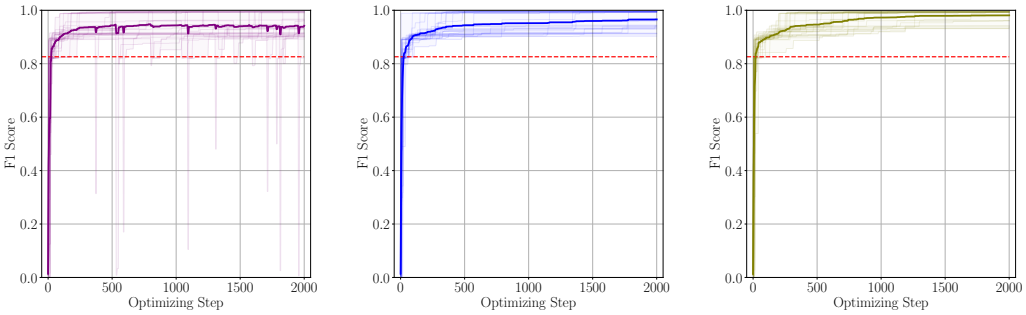
In this section, we assess the effectiveness of optimizers in refining the policy priority values. The ground truth data is split into 80% for training and 20% for testing, utilizing stratified sampling to improve the generalizability of the solution derived from this optimization process. This method guarantees that the distribution of positive and negative samples is preserved in both datasets. Each algorithm is executed 30 times with 2000 optimization steps during each run. For every run, the training and testing data are regenerated. The baseline score is the highest F1 score obtained from the allow-overrides algorithm (refer to Table 5.2) in the preliminary evaluation.

Optimization Results for Testing Data Figure 5.6 shows the optimization process results for six observed algorithms, including CMA, CMandAS3, DE, NGOpt, RandomSearch, and TBPSA. From our observation of this figure, we can conclude various points:

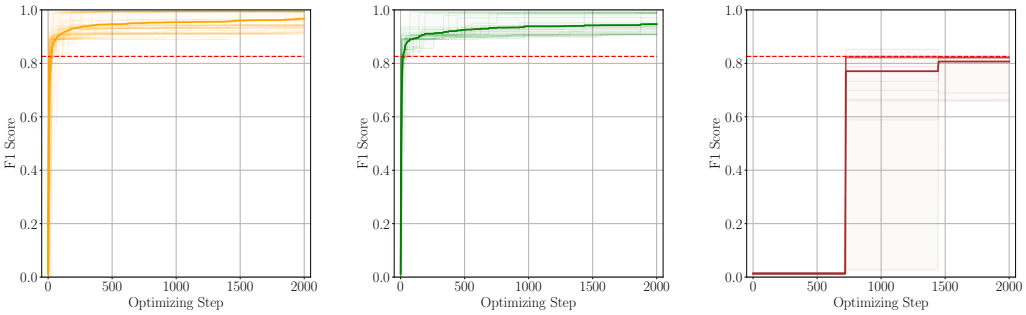
1. Overall, the optimization algorithms achieve F1 scores higher than the baseline, indicating that optimizing priority is feasible when ground truth data is available. Moreover, the performance of each algorithm varies across different runs.

2. When assessing the mean performance of the algorithms across eight distinct charts, it is clear that **TBPSA** produces the least favorable results, while **DE** attains the best performance. Significantly, **CMandAS3, DE, NGOpt**, and **Random Search** show an increasing trend in F1 scores as the number of optimization steps rises. Furthermore, **CMA** exhibits inconsistent performance in F1 scores during optimization.

Overall Comparison We evaluate the optimized solutions produced by the optimization process when all ground truth access requests are used. The optimal solution for each algorithm is the list of priority values that obtains the best F1 score during the optimization process. This evaluation is presented in Table 5.3.



(a) CMA (b) CMandAS3 (c) DE



(d) NGOpt (e) Random Search (f) TBPSA

Figure 5.6: Optimization results with respect to eight algorithms when 80 % ground truth data is used for training. Each chart illustrates the testing F1 scores across 30 independent runs (in lighter lines) and the mean score (in darkest lines). The horizontal red dashed line denotes the baseline score.

The priority values obtained from different optimizers are considerably more effective than those generated by the LLM, as indicated in Table 5.3. Remarkably, several optimizers, including **CMA, CMandAS3, DE**, and **NGOpt**, can reach the highest F1 score of 0.994. However, **NGOpt** shows slightly better performance, achieving the highest Recall value of 0.989. Currently, our system defaults to **NGOpt** as the optimization algorithm for optimizing the priority values of policies.

Table 5.3: Overall comparison between the performance of the LLM-generated and optimized priority values for access control decision-making. The bold values are the best results among the methods that use the priority-based combining algorithms, which are marked in gray.

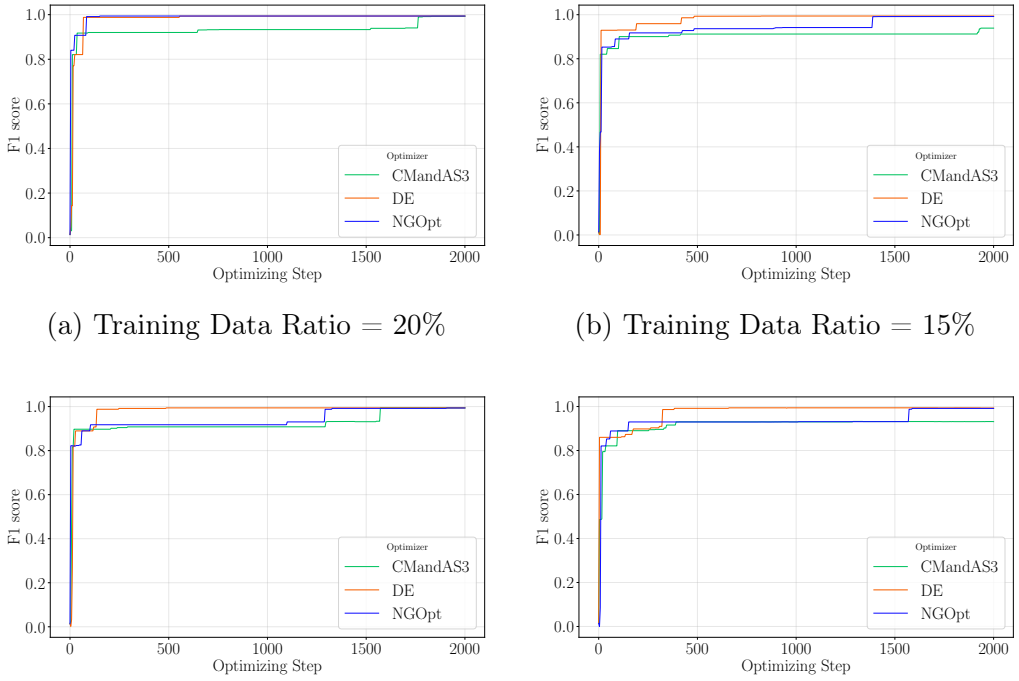
	TP	**FP**	**TN**	**FN**	**Precision**	**Recall**	**F1**
LLM-based priority	127	0	7801	18688	1	0.007	0.013
CMA	18597	0	7801	218	1	0.988	**0.994**
CMandAS3	18597	0	7801	218	1	0.988	**0.994**
DE	18587	0	7801	228	1	0.988	**0.994**
NGOpt	**18609**	0	7801	**206**	1	**0.989**	**0.994**
Random Search	18504	0	7801	311	1	0.983	0.992
TBPSA	18462	6150	1651	353	0.75	0.981	0.85

Priority Optimization with Limited Ground Truth Data In this experiment, we evaluate the effectiveness of priority optimization when access to ground truth data is limited. To simulate a situation with limited ground truth data, we use a small proportion of the available data as the training set for the optimizers. This also means that a large portion of the ground truth data is used as the test set. We apply a stratified strategy to divide the data. At every step of the optimization process, we record the F1 score calculated against the test data. Only the top three optimizers from the previous experiment were included in this experiment: CMandAS3, DE, and NGOpt.

As illustrated in the Figure 5.7, in extreme scenarios, such as when the training data ratio is only 5% , DE (orange line) and NGOpt (blue line) optimizers have successfully found an optimized solution. This is evidenced by the very high F1 scores (near 1.0) when evaluated with the test set. Looking across the charts, we can see that as the amount of training data increases, the optimization process shows faster progress. Furthermore, the

optimizer CMandAS3 (denoted with green line) achieves good optimization results when the training data ratio is set to 10% or higher.

When computer resources are not a limiting factor for conducting priority optimization, we believe that utilizing a small portion (even 5%) of the ground truth data for training is sufficient for achieving high performance. This highlights the practicality of our approach in real-world scenarios, where experts only need to provide a limited number of ground truth examples to optimize the priority values for LLM-generated ABAC policies.



(a) Training Data Ratio = 20%

(b) Training Data Ratio = 15%

(c) Training Data Ratio = 10%

(d) Training Data Ratio = 5 %

Figure 5.7: Optimization results with limited ground truth data. Each chart demonstrates the testing F1 scores with a specific training data ratio for three optimizers: CMandAS3, DE, and NGOpt.

5.4 Summary

This chapter presented a new methodology based on LLMs for creating fine-grained ABAC policies while tackling significant challenges associated with LLMs, such as insufficient context and length restrictions. The strategy integrates multiple elements, including data management and transformation,

prompt creation with RAG-like knowledge incorporation and multi-turn templates, attribute refinement, mixture-of-agents policy generation, and optimization of priorities.

We employed a typical ICS network as a case study to create 181 fine-grained ABAC policies. We highlighted several reasons why the direct implementation of these policies in decision-making processes can lead to unfavorable outcomes. Our tests with different optimization algorithms revealed that adjusting the priority values significantly boosts the effectiveness of the policies generated, achieving an F1 score of 0.994.

Optimizing priorities can improve the decision-making process for access control in policies generated by LLMs. However, its effectiveness is limited by the reliance on ground truth data. We conducted experiments to evaluate the current approach in an environment with limited ground truth data. The results reveal that we can achieve a good F1 score even when using just 5% of the available ground truth data for optimization.

Discussion In the methodology presented in this chapter, the need for text normalization is avoided thanks to the generalizability of LLMs. Additionally, only a limited number of examples are necessary for the LLMs to understand the tasks. However, the use of commercial LLMs still presents some challenges. First, a significant amount of data must be transmitted to an external server. In practical scenarios, it is crucial to avoid exposing sensitive system information to the outside environment. Furthermore, this data could potentially be used to train the models in the future, making it vulnerable to hackers who might exploit prompt engineering techniques. While the cost of using APIs is manageable during the development of this application (under 1000$), other applications that make thousands of API calls each day should consider the long-term costs involved. Moreover, as users become increasingly accustomed to LLMs, the demand for greater customization of the models emerges.

The policy generation presented in this chapter is designed mainly for the ICS network. However, the approach can be configured to work with other types of IT networks. To implement the policy generation solution for a specific IT network, we need to utilize security guidelines that are appropriate for that specific IT environment. Furthermore, since the generated policies are designed to operate directly within a Python environment for access control, it is essential to gather system information, including details about network devices and network structures.

Chapter 6

Framework for Cybersecurity-Adaptive Pre-training of LLMs

This chapter presents CyLLM-DAP, a framework for the cybersecurity-adaptive pre-training of Large Language Models (LLMs).

6.1 Problem Introduction

As highlighted in the discussion of the previous chapter, the utilization of commercial Large Language Models (LLMs) may face challenges such as data transmission to third-party servers, high long-term cost and lack of model customization. Recently, high-tech companies (e.g., Meta, Mistral) have published highly capable open-source LLMs that are competitive with commercial LLMs. With a high level of customization and data transparency, developing private LLMs based on these open-source models has become a new trend.

In the field of cybersecurity, the majority of LLM applications depend on custom fine-tuning or post-training techniques, such as prompt engineering. Although domain-adaptive pre-training (DAP) has been shown to enhance the model's effectiveness in specialized areas, its adoption in cybersecurity is limited due to the complex implementation requirements. This chapter presents CyLLM-DAP, a framework aimed at streamlining the process of domain specialization for LLMs in cybersecurity by simplifying the stages of data collection, preprocessing, and pre-training in low-resource environments.

We illustrate how CyLLM-DAP can be used to gather and process data, as well as create cybersecurity-specific LLMs (CyLLMs) based on cutting-edge

open-source models (Llama 3 and Mistral v0.3). Llama and Mistral models were chosen for developing the CyLLMs because they are created by well-known AI companies, Meta and Mistral. These models are also recognized within the research community, with a large number of active users.

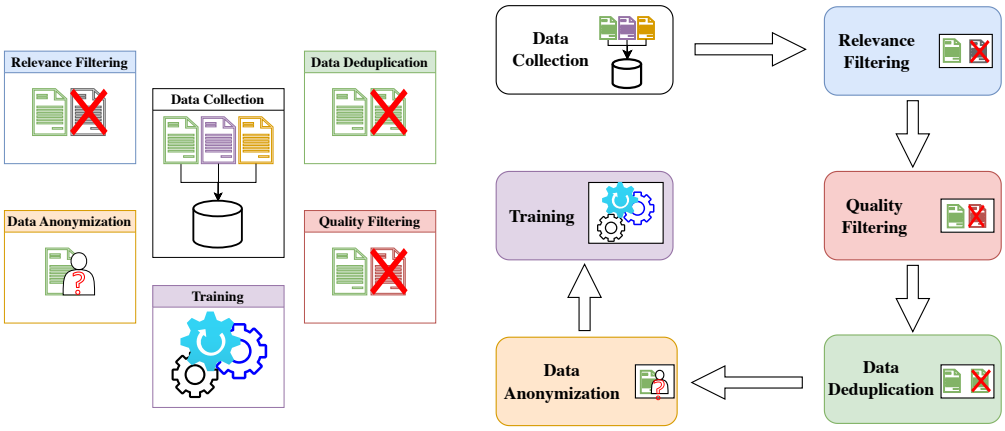
The benefits of DAP are validated through two experiments focusing on text classification and Q&A tasks. Our evaluation findings indicate that, when contrasted with general base or instruct models, integrating cybersecurity knowledge into the LLMs results in improved performance in every fine-tuning epoch for the text classification task and yields a performance boost of up to 4.75% for the Q&A task (comparable to DAP in other fields). The framework, the developed CyLLMs, and the data are made publicly accessible for use in cybersecurity applications.

6.2 The Architecture of Cybersecurity-Adaptive Pre-training Framework for LLMs

This section presents the general architecture of CyLLM-DAP. As shown in Figure 6.1, the framework contains six main components to support the domain-adaptive process in cybersecurity, including:

1. **Data Collection**: To collect data from different sources.
2. **Relevance Filtering**: To filter out irrelevant documents from the dataset.
3. **Quality Filtering**: To ensure the quality of the data, various methods can be utilized to filter out bad documents from the dataset.
4. **Data Anonymization**: To protect individuals' private and sensitive information.
5. **Data Deduplication**: To ensure the uniqueness of each document.
6. **Training**: To provide training scripts for the DAP process.

The framework allows for extensive customization through the principles of object-oriented programming. Users have the option to select a pre-defined workflow or develop a custom workflow of components tailored to their needs. Furthermore, they can also create their own components by utilizing the inheritance mechanism. It's important to acknowledge that not every part of the framework is developed from scratch by us. Instead, we leverage the valuable work done by other researchers and developers to develop some of the components.



(a) Framework components (b) Example workflow

Figure 6.1: CyLLM-DAP's components and example workflow during the practical use of the framework.

6.2.1 Data Collection

To guarantee the generalizability of LLMs, it is essential to gather training data from various sources. The variety of the data introduces the model to diverse language patterns, promoting fairness in the linguistic abilities of the models across different situations. Different data sources are currently supported by CyLLM-DAP, including:

1. Web data: Data obtained from web pages is known as web data, which is the most prevalent and repetitive type of data. The Common Crawl dataset [66] consists of data collected from the internet through crawling. This dataset, managed by the Common Crawl organization, has undergone regular crawls since 2007. As of now, Common Crawl represents the largest dataset, containing hundreds of TiB of information gathered from billions of web pages. When dealing with this kind of dataset, users have the option of working with either the HTML format (WARC format) or the plain text version (WET format) derived from these web pages.

   Our framework is designed to support data collection in both of these formats. However, the text extraction method initially used by Common Crawl to generate WET data is not ideal, as noted in [67]. Instead, we utilize the `trafilatura` Python library [68] to retrieve high-quality text from the raw HTML data (WARC format).

2. Academic papers: Academic papers represent another valuable source

of data, offering high-quality text produced by researchers. At present, S2OCR [69] stands as the largest and most established dataset of English academic papers that is regularly maintained. All the papers included in S2OCR have been converted from various data formats (such as PDF and LaTeX source code) into a more user-friendly format. In addition to metadata like corpus ID, the content of the papers is labeled to facilitate the easy extraction of titles, abstracts, and paragraphs. Currently, CyLLM-DAP exclusively supports this dataset for the collection of academic papers.

3. Hugging Face hub: Hugging Face (HF) [70] is a well-known ML framework that includes open-source libraries and tools for creating and implementing advanced natural language processing (NLP) models. Additionally, the HF hub serves as a platform for researchers and developers to share or access a large variety of developed models and datasets. Consequently, the hub is a vital source of data in terms of redundancy, diversity, and quality. In CyLLM-DAP, we presently facilitate data collection from Wikipedia and RedPajama [71]. Users can easily modify the framework for other datasets available on the HF hub.

4. Books: Books represent a valuable source of information that should be utilized during the pre-training of LLMs. Although CyLLM-DAP now facilitates the downloading of books from the internet, users need to be mindful of copyright issues when acquiring them.

5. Code: Pre-training LLMs on code data is beneficial for subsequent tasks related to programming. Code data can be sourced from online platforms that host code or from question-and-answer sites (such as GitHub and StackOverflow).

CyLLM-DAP utilizes multiple collectors to facilitate the gathering of data from the specified sources and convert it into text-based formats. It is important to note that the data collectors in CyLLM-DAP are designed to operate in efficient modes to prevent the need for downloading all raw data onto local storage. To achieve this, the collectors are equipped with basic relevance filters (as discussed in Section 6.2.2.1) to eliminate irrelevant documents in real-time.

6.2.2 Data Filtering

In this part, we examine the two primary components of CyLLM-DAP related to data filtering: relevance filtering and quality filtering. Typically, a

data filter receives a document as input and produces a boolean value that indicates whether the document satisfies certain established criteria. Data filters may be utilized as components of the data collectors for initial filtering or function as independent modules within the workflow.

6.2.2.1 Relevance Filtering

When organizing data for the specialization of LLMs in a specific domain, it is essential to assess the relevancy of data, like documents, to the intended knowledge area. Based on the characteristics of the target domain, various strategies can be effectively applied. For the purpose of relevance filtering within the cybersecurity field, as noted previously, the following aspects should be taken into account:

1. Vague domain boundary: The field of cybersecurity lacks a distinct separation from other areas. Computer systems are utilized across various domains to enhance their operations with higher accuracy. However, these computer-based solutions introduce certain risks and cybersecurity challenges. To effectively tackle cybersecurity problems in such systems, it is essential to possess knowledge that spans multiple domains. For example, to prevent online fraud in internet banking, one must integrate expertise in both cybersecurity and finance.

2. Broad and diverse: Cybersecurity includes multiple sub-domains, such as hardware security, software security, data security, and code security. The level of specificity within these sub-domains differs, with some being broad, like CTI, and others requiring more specialized or niche expertise. For instance, Generative AI Security represents a distinct area within cybersecurity, focusing on security concerns related to generative AI tools and solutions, including LLMs.

In CyLLM-DAP, there are two main methods used for relevance filtering, as follows.

Keyword-based Relevance Filtering The keyword-based method starts with a compilation of cybersecurity keywords, regex and URL patterns. CyLLM-DAP currently utilizes a collection of 500 cybersecurity terms (for instance, data security, data safety) and regex patterns (such as CVE). For URL, we compile a list of 300 cybersecurity news sites and blogs (e.g., www.scmagazine.com). Please note that these lists are regularly updated.

The keywords and patterns utilized in CyLLM-DAP are enclosed in filters, each employing distinct techniques for evaluating different components

(like URLs, text, and titles) of the document for relevance filtering. A significant advantage of keyword-based filtering is its speed and suitability for multiprocessing. Each filter uses minimal memory to execute the function. Therefore, we suggest using these keyword-based filters as the initial method for managing large data sources.

Model-based Relevance Filtering In model-based filtering, language models are utilized to assess how relevant a document is to the field of cybersecurity. To categorize documents into either cybersecurity or non-cybersecurity, an encoder-only model, which has already been pre-trained in the cybersecurity context, is fine-tuned using a labeled dataset. Following the fine-tuning process, the model is capable of assigning a cybersecurity relevance score to an input document. By applying a threshold of 0.5, any document that receives a cybersecurity relevance score below 0.5 will be excluded from the dataset.

6.2.2.2 Quality Filtering

In LLM research, the quality of data is a critical factor that influences the performance of the model. Numerous quality metrics have been established across various LLM projects based on observations of real data. These quality signals and metrics are primarily derived from analyzing the statistics of the text, such as the ratio of sentences that begin with bullet points compared to other sentences.

In CyLLM-DAP, drawing inspiration from other LLM research [71], we have created functions for quality metrics and rules. The metric functions accept a document as input and provide corresponding metric scores as output. Moreover, the rules in CyLLM-DAP are implemented as filters, applying specific thresholds to the quality metrics of a document to assess whether it is considered good. For instance, the aforementioned ratio must not exceed 0.7 to classify a document as good.

Although these metrics have demonstrated their effectiveness, they are not universally applicable and should be chosen carefully based on the specific tasks at hand. In addition to the list of default metrics and rules tailored for the cybersecurity domain, LLM developers have the flexibility to create their own functions to address their specific requirements.

6.2.3 Data Deduplication

Duplicate entries within training datasets can lead to overfitting and create a misleading impression of enhanced performance during training. Recent

studies, as noted in [72], highlight the significance of eliminating duplicates, emphasizing its ability to improve model performance. To remove duplicate data generally, we must first identify duplicates through similarity searching and eliminate them.

In an exhaustive similarity search, each document pair in the dataset is assessed for similarity. This method incurs a significant time cost and is impractical for managing large datasets. In CyLLM-DAP, we employ an efficient similarity search technique known as Locality Sensitive Hashing (LSH) [73]. Essentially, LSH aims to reduce the data dimensions while preserving local distances among data points. The result of this approach is groups of similar documents, called buckets. After this, only the longest document from each group is kept, with the others eliminated from the dataset.

When handling smaller datasets, the entire deduplication procedure can be executed in the computer's RAM. However, during the execution of LSH, a substantial amount of intermediate data (like dense hash signatures) is produced. To handle large dataset, CyLLM-DAP divides the deduplication process into smaller phases and stores any intermediary data on local storage.

6.2.4 Data Anonymization

Data anonymization is a method of sanitizing data to protect an individual's privacy or sensitive information. This approach involves identifying and either removing or altering personally identifiable information (PII) from the text prior to inputting it into LLMs. Consequently, LLMs do not retain personal information, thereby reducing the risk of it being accessed by cyber attackers during the inference process. CyLLM-DAP employs the presidio anonymizer [74] created by Microsoft to develop its anonymization functions. Currently, it addresses four types of PII, which include email addresses, URLs, phone numbers, and credit card information. Additional types of PII will be incorporated in the future.

6.2.5 Training

In CyLLM-DAP, we develop various scripts that users can directly utilize for domain specialization. These scripts primarily leverage the HuggingFace library and its code samples. Since HuggingFace offers top-tier libraries and tools for ML and LLM models, utilizing their examples guarantees a strong yet straightforward method for LLM development. The scripts adhere to two prevalent LLM pre-training strategies, taking into account the available computing resources. One strategy involves full pre-training, where all model

parameters are adjusted throughout the pre-training phase. This method demands a significant amount of GPU RAM and carries the risk of catastrophic forgetting of prior knowledge. The alternative is to only partially update the model's parameters, which is better suited for environments with limited computing resources.

6.3 CyLLMs Creation using CyLLM-DAP

Utilizing CyLLM-DAP, we develop cybersecurity-specific foundation LLMs, referred to as CyLLMs. The two versions of CyLLM, namely CyLlama3 and CyMistral, are constructed on the foundations of two corresponding open-source LLMs: Llama-8B-v3 and Mistral-7B-v0.3. We begin by outlining the process of data preparation. Next, we carry out the DAP to adapt the foundation models with cybersecurity knowledge. Both of these tasks utilize CyLLM-DAP. Finally, we assess the effects of domain-adaptive pre-training through experiments conducted on two different downstream tasks. The entire procedure is illustrated in Figure 6.2, where the data preparation and DAP processes are elaborated upon in Appendix B.1 and Appendix B.2, respectively.

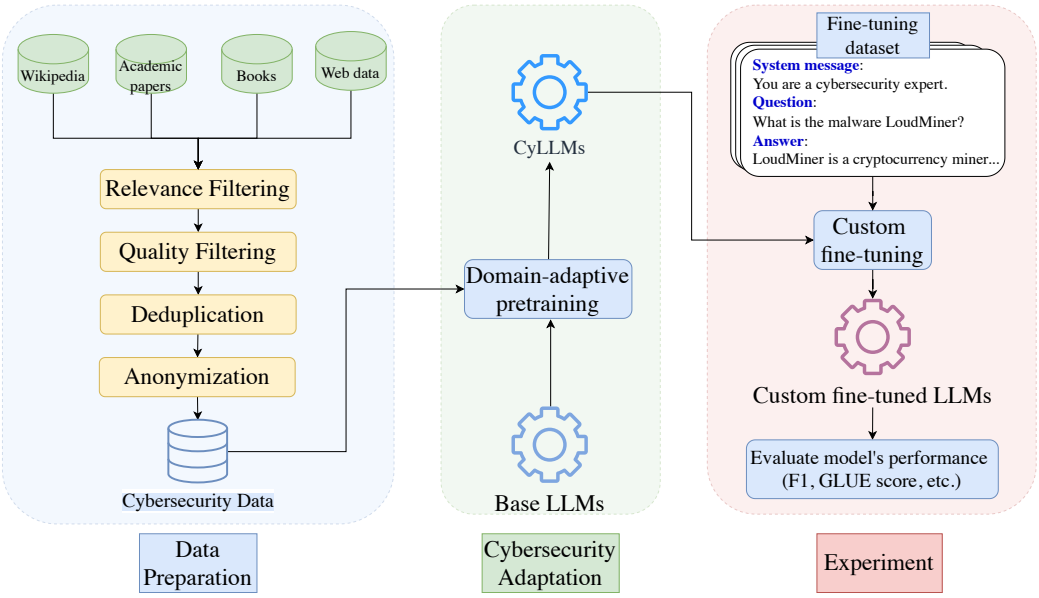


Figure 6.2: Development process of CyLLMs using CyLLM-DAP.

6.4 Evaluation

In this part, we utilize two downstream tasks to assess the impact of CyLLMs (produced through CyLLM-DAP) in different cybersecurity applications, especially those involving specialized datasets. There are several key points to consider:

1. In this assessment, we are not concentrating on developing multi-purpose instruct LLMs that can perform well in all cybersecurity tasks. We carry out two experiments in which appropriate models are individually fine-tuned.

2. The assessment aims to validate the effectiveness of these CyLLMs on private and specialized datasets. This scenario is a typical application for open-source LLMs in the cybersecurity domain.

As shown in Table 6.1, three groups of LLMs are related to the experiments. Each group contains corresponding models from the Llama 3 and Mistral v0.3 LLM families.

Table 6.1: The list of LLMs involved in the experiments and the type of training method already applied to them.

Model Group	**General Pre-training**	**Domain-adaptive Pre-training**	**General Instruction Fine-tuning**
Group **B**: - **B**aseMistral - **B**aseLlama3	✓		
Group **I**: - **I**nstructMistral - **I**nstructLlama3	✓		✓
Group **C**: - **C**yMistral - **C**yLlama3	✓	✓	

- Group B are the base (foundation) Llama-3-8B (BaseLlama3) and Mistral-7B-v0.3 (BaseMistral) models. Base models are developed via general pre-training to inject general knowledge into these models.

- Group I are Llama-3-8B-Instruct (InstructLlama3) and Mistral-7B-Instruct-v0.3 (InstructMistral) models. These models have undergone both general pre-training and general instruction fine-tuning. They have the capacity to answer questions in the general domain of knowledge.

- Group C comprises cybersecurity-specific LLMs (CyLLMs), namely CyLlama3 and CyMistral. These are developed based on BaseLlama3 and BaseMistral via domain-adaptive pre-training, respectively. These models have both general knowledge and cybersecurity knowledge.

To verify the efficacy of the DAP, every model in groups B, I, and C undergoes task-specific instruction fine-tuning to address the target tasks. Ultimately, during the inference phase to evaluate the performance of the relevant LLMs, the models in each group have already undergone different training pipelines:

- **Group B**: general pre-training → task-specific instruction fine-tuning
- **Group I**: general pre-training → general instruction fine-tuning → task-specific instruction fine-tuning
- **Group C**: general pre-training → domain-adaptive pre-training → task-specific instruction fine-tuning

There are other pipelines that we can utilize in this evaluation. For example, pipeline **(a)** general pre-training → general instruction fine-tuning → domain-adaptive pre-training → task-specific instruction fine-tuning; and pipeline **(b)** general pre-training → domain-adaptive pre-training → general instruction fine-tuning → task-specific instruction fine-tuning. However, there are several reasons why we did not follow these pipelines:

- We did not follow the pipeline **(a)** because domain-adaptive pre-training should be implemented on base LLMs. Similar to general pre-training, DAP will provide the LLM with input text in its original form, using extensive unlabeled text. In contrast, general instruction fine-tuning will utilize a supervised dataset focused on question-and-answer formats. This supervised data will be formatted using a specific template before being fed into the LLM.

  If we continue to pre-train an instruction-tuned LLM on a large amount of unlabeled text using DAP, we may diminish or eliminate its ability to follow instructions since we are no longer focusing on "instruction-following." This could result in a "forgetting" of the instruction-following behaviors that were learned earlier, a phenomenon referred to as catastrophic forgetting.

- We did not utilize pipeline **(b)** since we lack access to the general instruction dataset that was employed to develop InstructLlama or

Instruct Mistral from their foundational models. Neither Meta nor Mistral has released their instruction datasets. Although there are attempts to recreate these datasets, it remains on a smaller scale and only functions as an approximation.

While we could implement a smaller instruction dataset for pipeline **(b)**, this would ultimately compromise the assessment of the dataset's quality. It is clear that this smaller dataset would not match the quality of the original datasets that transformed the base models into general instruction models.

In the experiments presented next, we utilize 90% of the dataset for training and 10% for testing. Following the fine-tuning phase, we assess and compare their performances using various metrics. During the inference phase, we prompt the LLM only once for each question in the test set. Given the consistent use of inference parameters and the fine-tuning process conducted, we believe that the non-determinism of the LLM does not significantly impact the fairness of the evaluation. The specifics of each experiment are outlined in the subsequent sections.

6.4.1 Task 1: Text Classification

This experiment in text classification focuses on evaluating the capability of LLMs to generate short responses for categorizing cybersecurity texts into various classes. For this study, we have compiled a dataset containing 145,000 samples. Each sample comprises text pertaining to a specific cyberattack technique, accompanied by its corresponding MITRE ATT&CK technique ID. This undertaking involves multi-class classification with 628 ATT&CK technique IDs serving as the categories. Overall, the process of dataset creation unfolds in two phases:

1. Stage 1: We utilize the automatic report analysis framework, namely RAF-AG (previously discussed in Chapter 4), to obtain cyberattack paths from CTI reports. For each attack path, the attacking technique ID of an attack step, along with its correlated text, is gathered.

2. Stage 2: Data augmentation is done with OpenAI GPT 3.5 model, by which an input text is paraphrased into various text patterns.

We employ the Alpaca prompt template [75] to convert the data into a single prompt used for fine-tuning. Since general instruction fine-tuned models (group I models) were initially fine-tuned with specific prompt templates,

continually fine-tuning them with an Alpaca prompt template may ruin their capability. Therefore, only Group B and C's models are included in this experiment. Following the custom fine-tuning, the models should identify the most likely technique ID for the given input text.

For fine-tuning, a single A100 40 GB machine is utilized, using LORA (with a rank of 64 and alpha of 128) [76] and a learning rate of $1e-4$. 90% dataset is designated for fine-tuning, with the remaining portion set aside for evaluation. Model performance is recorded at the conclusion of each epoch over a ten-epoch period. During the evaluation stage, we collect responses from the LLM for all samples in the evaluation set. We then extract technique IDs from these responses and compare them with the ground truth to compute the F1 score.

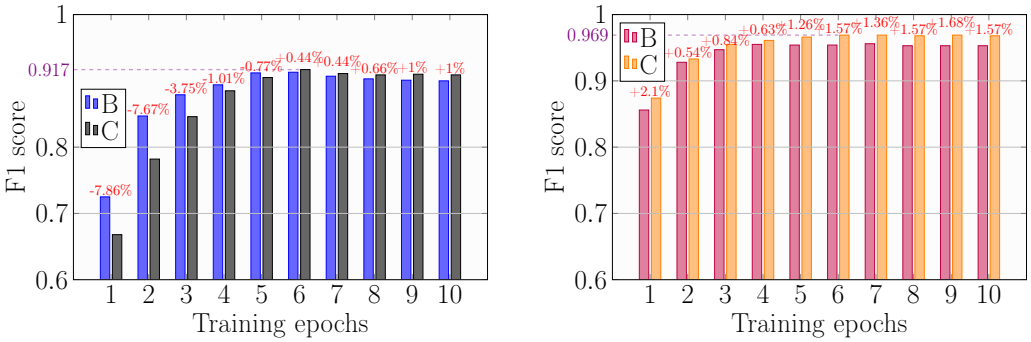
As illustrated in Figure 6.3, incorporating cybersecurity knowledge into the LLMs (group C models) leads to notable performance enhancements when compared to the baseline foundation models (group B models) (see values marked in red). This effect is consistently evident throughout each epoch of fine-tuning with the Mistral v0.3 models (displayed on the right). In contrast, the impact of cybersecurity knowledge shows variability across epochs for the Llama 3 models, becoming more pronounced when the number of epochs exceeds six (≥ 6). Additionally, models from the Mistral v0.3 series (depicted in the right chart) outperform others, achieving a maximum F1 score of 0.969, compared to 0.917 for the Llama 3 series (shown in the left chart).

6.4.2 Task 2: Question & Answer

In this study, we create a simple Q&A chatbot designed to effectively respond to user questions related to CTI, including attacking techniques, malware, and attackers. The goal of the study is to assess the ability of LLMs to produce lengthy responses while ensuring they closely match the expected answers. This is especially important in scenarios where the chatbot must deliver precise and thorough information without any inaccuracies.

We utilize a dataset containing 22,000 examples of conversations, which include questions along with their expected responses. The dataset also includes a system message instructing the LLM to act as a cybersecurity specialist throughout the conversation. An example of the data can be seen in Table 6.2.

All the models listed in Table 6.1 participate in this study. They are subsequently fine-tuned using the dataset to develop their Q&A capabilities. It is important to note that we use the default chat templates that were originally developed by the companies responsible for each model family. In



(a) Llama 3 series

(b) Mistral v0.3 series

Figure 6.3: The performance of group C models (CyLLMs) and group B models (Base models) measured via the F1 score for text classification. The performance differences in percentage between group C models and group B models are shown in red. Horizontal violet dashed lines and numbers show the maximum F1 scores in each chart. The left-hand side chart shows the Llama 3 LLM series, and the right-hand side chart shows the Mistral v0.3 LLM series.

Table 6.2: The task-specific dataset used in the experiment with examples.

Related Task	**Example**
Text classification (145,000 samples)	- **System message**: You are a cybersecurity expert. Below is an instruction that describes a task in the cybersecurity domain, paired with an input that provides further context. Write a response that appropriately completes the request. - **Instruction**: You are given a text description of a procedure example. Identify the MITRE ATT&CK technique used. - **Input**: Siloscape leverages a sophisticated form of API call concealment... - **Output**: Obfuscated Files or Information T1027
Question & Answering (22,000 samples)	- **System message**: You are a helpful cybersecurity expert. - **Question**: What is the malware LoudMiner? - **Answer**: LoudMiner is a cryptocurrency miner that uses virtualization software to siphon system resources...

assessing the text produced by the fine-tuned LLMs, we consider the following metrics:

1. The **GLEU (Google BLEU) score** [77] is frequently utilized to assess the effectiveness of translation systems by evaluating the resemblance between machine-generated translations and the correct results.

2. The **BERTScore** [45] utilizes contextual embeddings produced by the BERT [9] model. This metric is commonly employed to evaluate the similarity between texts and has been shown to yield results that align closely with human assessments.

3. We use a commercial LLM (GPT-4o) (**LLM-as-a-judge**) to assess the produced response based on Understandability (clarity of the answer), Relevance (sufficiency and appropriateness of the information), Naturalness (human-like quality of the response), and Hallucination (presentation of incorrect information).

Table 6.3 presents the performance of various models concerning Q&A fine-tuning across six key metrics: GLEU score, BERTScore, Understandability, Naturalness, Relevance, and Hallucination. The word count is included as a sub-metric primarily for reference. From this table, we can see that:

- Overall, integrating cybersecurity expertise into the models improves their effectiveness compared to those that rely solely on general knowledge. Among the Mistral v0.3 series, CyMistral is the leading model across all six evaluation metrics, while CyLlama3 excels beyond other models in 4 out of 6 metrics within the Llama 3 group. Additionally, the Llama series shows a 4.75% GLEU score increase for CyLlama3 compared to BaseLlama3. Mistral series achieves increases of 3.07% in GLEU score, and a 2.85% BERTScore between CyMistral and Instruct-Mistral (the second-best model). These performance improvements are on par with those achieved by incorporating specialized knowledge in various other fields (e.g., medical [31]).

- When comparing model families, Mistral models tend to surpass their Llama 3 counterparts, especially regarding the GLEU score. The top Mistral model, CyMistral, attains a GLEU score of 0.805, while the leading Llama3 model, CyLlama3, records a score of 0.596. This difference in performance can be explained by the tendency of Llama3 models to produce longer responses in comparison to Mistral models, as shown in the word count column. The disparity in length between

the generated text and the expected text has a considerable effect on the calculation of the GLEU score, leading to a reduced output score for Llama 3 models.

- It is evident from the Table 6.3 that adding cybersecurity knowledge does not lead to a notable enhancement in the Understandability and Naturalness metrics. This is mainly due to the fact that all models used are built on foundation models that have been pre-trained with high-quality text, enabling them to produce coherent and natural language. Furthermore, the Llama 3 models are not as effective as the Mistral models in capturing all the essential information found in the expected answer, as shown by the Relevance metric. Regarding the Hallucination metric, the Llama 3 models often produce additional information that could be incorrect, resulting in a high Hallucination score as evaluated by the LLM.

Table 6.3: The performance of the created LLMs for the Q&A task. The underlined scores show the best models within the same family (same background color). The bold scores show the best model for specific metrics among all of the involved models. Unlike other metrics, models with a lower hallucination score are better. Metrics: G (GLEU score), B (BERTScore), U (Understandability), N (Naturalness), R (Relevance), H (Hallucination), AL (Average Length).

Model	G	B	U	N	R	H	AL
BaseLlama3	0.569	0.718	0.846	0.77	0.618	0.565	115.615
BaseMistral	0.778	0.91	0.957	0.937	0.835	0.119	64.699
InstructLlama3	0.565	0.707	0.828	0.747	0.603	0.592	116.474
InstructMistral	0.781	0.911	0.954	0.934	0.828	0.118	64.01
CyLlama3	0.596	0.717	0.854	0.762	0.63	0.563	111.943
CyMistral	**0.805**	**0.937**	**0.959**	**0.939**	**0.84**	**0.103**	64.022

Typically, the evaluations made by LLMs are helpful and reliable, as their results correspond with other deterministic metrics, such as BERTScore and GLEU score. Additionally, it can be deduced that DAP is less effective for Llama 3 models compared to Mistral models. This suggests that further considerations are necessary when employing Llama 3 models.

6.5 Summary

In this chapter, we present CyLLM-DAP, a framework designed to support key tasks in the cybersecurity specialization process for open-source LLMs. This framework is comprised of modules that can be used in their default states or tailored to collect data and ensure data quality prior to engaging in DAP.

We demonstrated how CyLLM-DAP can be employed in order to develop cybersecurity-specific LLMs. How these CyLLMs can be used in tasks such as text classification and Q&A is also presented. Additionally, similar to DAP in other fields, specializing the LLM for cybersecurity can lead to significant performance enhancements, up to 4.75% (for the Q&A task), when compared to the general base and instruct models.

The framework, models, and cybersecurity data are made publicly accessible to promote the adoption of LLMs in the realm of cybersecurity.

Chapter 7

Cybersecurity-specific LLM-based CTI Report Analysis

This chapter presents a methodology to combine the previously discussed approaches for solving cybersecurity downstream tasks. We also introduce an LLM-based CTI report analysis as an example of this methodology.

7.1 Problem Introduction

Developing cybersecurity applications often requires data annotation by experts to gather sufficient data for model training. Given the rapidly evolving nature of the cybersecurity field, addressing the problem of data insufficiency is critical, as new systems typically require freshly annotated data. Additionally, data can quickly become obsolete. In the previous chapter, we introduced methodologies to tackle data scarcity with advancements in NLP and DL, such as Weak Supervision and LLMs. In Chapter 4, we utilized Weak Supervision to avoid starting the data annotation process from scratch when developing a CTI report analysis. We also replace the cybersecurity relationships with grammatical relationships to extract cybersecurity events from text. In Chapter 5, we also leveraged LLMs to harness their generalizability for generating ABAC policies. This allows us to solve the policy generation task without text normalization or using many annotated examples. Furthermore, we developed the CyLLM-DAP framework (Chapter 6) to support the cybersecurity specialization of open-source LLMs, which can be applied to various downstream tasks in cybersecurity.

In this chapter, we present a methodology that integrates these previous approaches to develop cybersecurity applications in scenarios where annotated data insufficiency arises. This methodology consists of four main steps,

including one optional step, as follows:

1. **Data Generation**: Data can first be collected from public CTI frameworks, such as MITRE ATT&CK, or from private data sources. Next, we can utilize RAF-AG as a weak supervision source to generate the first stage of data annotation, which includes entities, relationships, and secondary labels, such as attack techniques. We can either use this weakly annotated data directly for training or further augment it with a commercial LLM.

2. **Cybersecurity Specialization (Optional)**: As mentioned, domain-adaptive pretraining can enhance model performance in specific domains. We could directly utilize CyLLM-DAP with customized settings to create cybersecurity-specific LLMs customized for particular needs. However, this step can be skipped since we can directly use published CyLLMs for problem-solving.

3. **Model Fine-tuning**: Once the data is prepared, it is essential to fine-tune the model to facilitate the problem-solving capacity. The published cybersecurity-specific LLMs (such as CyLlama3 and CyMistral mentioned in Chapter 6) are already pre-trained with cybersecurity knowledge and ready to fine-tune for a specific task.

4. **Model Deployment**: After the model is fine-tuned, it can be used to address specific tasks or integrated into a larger system to tackle more complex challenges.

We introduce how to apply this methodology to analyze CTI reports in the remainder of this chapter. We collect and enrich CTI knowledge from the MITRE ATT&CK framework to construct the training dataset. Following a fine-tuning phase, the CyLLM becomes skilled at identifying the technique ID from the provided text. We break down CTI reports into smaller, action-oriented segments and use the LLM technique recognizer to spot malicious behaviors. Our initial evaluation indicates that this approach can achieve an F1 score of 0.716, showing performance similar to another report analysis framework, RAF-AG, which reached a 0.784 F1 score.

7.2 CTI Report Analysis Based on Cybersecurity-specific Large Language Models

In this section, we outline the design and implementation of the LLM-based report analysis. The main architecture can be seen in Figure 7.1. This

method consists of two primary elements: the creation of the LLM-based technique recognizer and the analysis of reports. Each of these elements will be elaborated on in the subsequent sections.

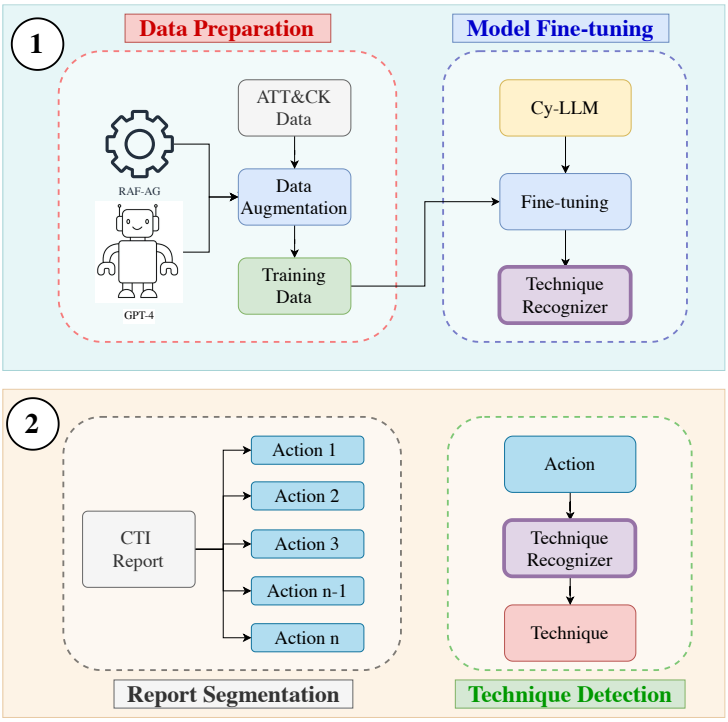


Figure 7.1: Overall architecture with two main components: ① Development of LLM-based technique recognizer; ② Report analysis.

7.2.1 Development of LLM-based Technique Recognizer

In this section, we present our approach to developing a technique recognizer that is based on an open-source LLM. There are two sub-components:

1. **Data Preparation**: We collect the data from MITRE ATT&CK and use GPT-4 [3] and RAF-AG [78] for data augmentation.

2. **Model Fine-tuning**: A fine-tuning procedure is carried out to customize a CyLLM (see Chapter 6) for the given task. This process leads to the development of a technique recognizer based on LLM technology.

7.2.1.1 Data Preparation

This component, presented within a red dashed box in Figure 7.1, aims to prepare the training data for fine-tuning. As stated in Section 2.1, the

MITRE ATT&CK framework encompasses a vast array of CTI data, which is structured into specific tactics and techniques. Each technique features a set of brief descriptions known as procedure examples, which outline how attackers might employ the technique in particular situations. The procedures and corresponding techniques within the ATT&CK framework account for the majority of our training data.

To improve the variety of language patterns, we utilize RAF-AG [78] to assess realistic CTI reports and extract texts that include both technique and tactic information. As a result, we could collect many cybersecurity-related texts with the MITRE ATT&CK tactics/techniques as their labels. It is essential to recognize that sometimes the text may be overly generic, preventing RAF-AG from providing specific technique information. In such cases, it defaults to tactic information instead. As a result, there are relatively few samples in our training data labeled with tactic information rather than technique.

Following the collection of procedure examples, we conducted data augmentation using GPT-4, a leading commercial language model. The purpose of this data augmentation is to increase the variety of our dataset, allowing it to contain a wider range of language patterns found in different CTI reports. To achieve this, we present each procedure example to the language model to obtain paraphrased versions of the input. The outcome is a rich compilation of procedure examples that closely resemble those created by cybersecurity experts in their online blog entries. A straightforward illustration of this process can be seen in Figure 7.2.

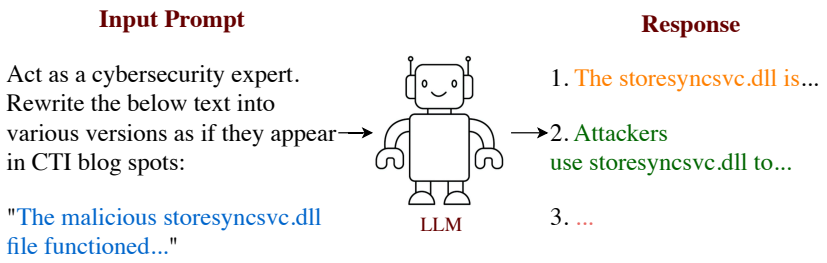


Figure 7.2: Input and output example of the data augmentation.

After completing the collection and augmentation process, we obtain a training dataset that consists of 154,000 samples. This dataset is available on the Hugging Face (HF) hub [70], a platform specifically designed for the sharing of assets and models among professionals in ML. A data viewer from the HF site is shown in Figure 7.3. As illustrated in the figure, every data sample includes a system message, a task instruction, a procedure example

(input), and technique information (output).

system string · classes	instruction string · classes	input string · lengths	output string · classes
You are a … 100%	You are gi… 100%	256⟷320 13%	Exploit Pu… 0.4%
You are a cybersecurity expert. Below is an instruction that describes a task in the cybersecurity domain, paired with an input that provides further context. Write a response that appropriately completes the request.	You are given a text description of a procedure example. Identify the MITRE ATT&CK technique or tactic used.	On March 5, 2020, cybersecurity expert Steven Seeley disclosed a zero-day vulnerability in Zoho ManageEngine Desktop Central, affecting versions before 10.0.474, identified as CVE-2020-10189. He also released proof-of-concept code demonstrating the remote code execution flaw.	Exploit Public-Facing Application - T1190
You are a cybersecurity expert. Below is an…	You are given a text description of a procedure…	Researcher Steven Seeley announced a critical zero-day flaw in Zoho…	Exploit Public-Facing Application - T1190
You are a cybersecurity expert. Below is an…	You are given a text description of a procedure…	On March 5, 2020, a zero-day remote code execution vulnerability was…	Exploit Public-Facing Application - T1190

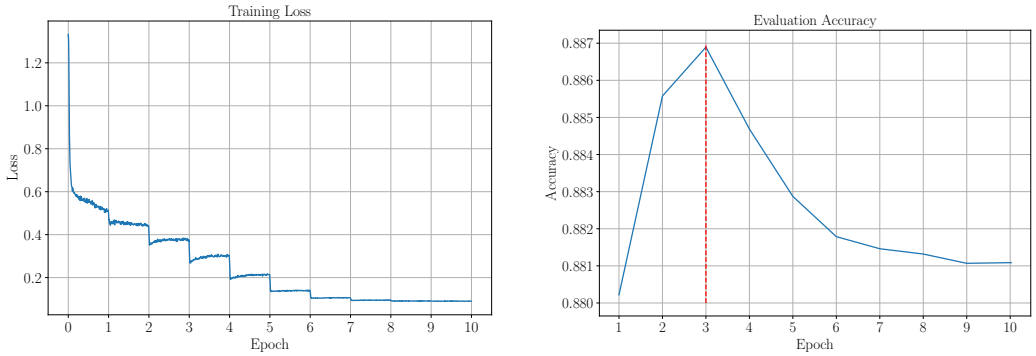
Figure 7.3: An overview of training data.

7.2.1.2 Model Fine-tuning

To minimize hallucinations in LLMs when addressing new downstream tasks, it is essential to fine-tune the model. The primary workflow of the fine-tuning procedure in our method is depicted in the blue dashed box of Figure 7.1. Generally, fine-tuning consists of modifying part or all of the model's parameters to improve its capacity to generate the expected outputs. As discussed in Section 2.5, PEFT methods, particularly LoRA, have proven effective in partially fine-tuning LLMs while still ensuring competitive performance. In our methodology, we configure the rank and alpha, which are two critical LoRA hyperparameters, to be high (64 and 128, respectively) to enable a greater number of trainable parameters. Moreover, we opt to fine-tune an open-source cybersecurity-specific LLM (based on Mistral 7B v0.3 [29]) for our task. It is important to note that this open-source LLM has already been refined for cybersecurity through a continual pre-training process using cyber textual data using the CyLLM-DAP framework (see Chapter 6)

During the fine-tuning process, the LLM is provided with examples of procedures along with their related technique/tactic information (e.g., names and IDs). Each sample in the training dataset is converted into a single text sequence utilizing the alpaca format [75]. This step is crucial for making the data compatible with the LLM. Upon completing fine-tuning, we expect that the LLM will accurately generate the correct technique/tactic name and ID in response to specific input text. This task resembles text classification involving a significantly larger number of classes. The outcome of this component is a fine-tuned LLM, referred to as the technique recognizer, which is prepared for use in further analysis.

It is crucial to highlight that our training dataset consists solely of positive examples, as each procedure must relate to at least one technique. This factor should be considered during the report analysis. We address this issue in Section 7.2.2.2. In Figure 7.4, we illustrate the fine-tuning procedure of this research. The fine-tuning is conducted using an Nvidia A100 GPU with 40 GB of memory over ten epochs. Our findings from the figure suggest that a limited number of training epochs is optimal for enhancing the LLM's performance. Although the model demonstrates progress throughout the training phase, it starts to overfit after three epochs, which is indicated by a decrease in evaluation accuracy. As a result, we decided to stop training after the third epoch and use the output language model from that point as the technique recognizer.



(a) Training curve (b) Evaluation curve

Figure 7.4: Fine-tuning process with PEFT.

7.2.2 Report Analysis

This section explains our method for analyzing a CTI report. We employ the technique recognizer created from the earlier stage for report analysis. This element consists of two smaller sub-components, outlined below.

1. **Report Segmentation**: We first split a CTI report into smaller pieces of information, where each of them is an action.

2. **Technique Detection**: We then utilize the technique recognizer to detect the malicious technique described in each action.

7.2.2.1 Report Segmentation

This component (illustrated in the black dashed box of Figure 7.1) aims to break down a lengthy report into smaller pieces that facilitate technique recognition. While a straightforward approach might simply involve dividing the report into sentences, the complexity of certain sentences can result in the relevance of multiple attacking techniques. As a result, we further divide each sentence into smaller, action-focused fragments. Each fragment will include at least one VERB and one OBJECT, with any extra information being optional.

We utilized a Python library named SpaCy [79] for this purpose. First, SpaCy analyzes the sentences in the report by constructing dependency trees. We then traverse through each tree to determine verbs and their associated objects. Using these verbs, we extract the text that captures the actions found within the sentence. A straightforward illustration of this procedure can be seen in Figure 7.5. This approach enables us to create a list of action-centric text from the report, where each element contains sufficient information for technique recognition.

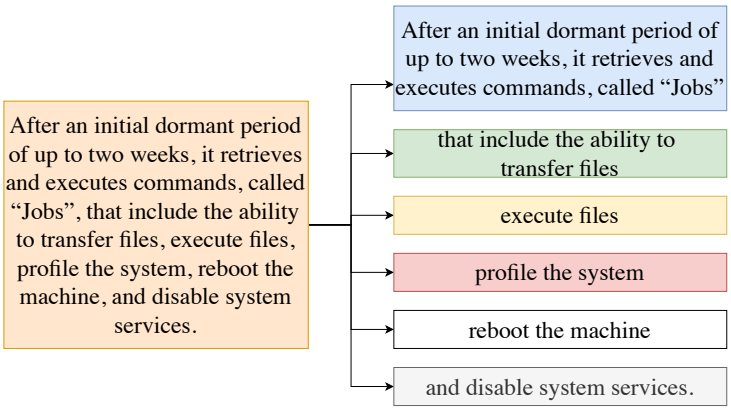


Figure 7.5: An example of the report segmentation.

7.2.2.2 Technique Detection

Following the fine-tuning process, the LLM-based technique recognizer can detect malicious techniques and tactics within the input text. Once the report is divided into segments, each piece of information becomes ready for technique detection (illustrated by the green dashed box in Figure 7.1). Each fragment of text is fed into the LLM technique recognizer to identify any techniques it may include.

This research examines two separate categories of text: positive and negative. Positive texts convey information about malicious cyberattack techniques and tactics, while negative texts do not. Importantly, the LLM recognizer is also capable of generating technical information for all input texts, including those classified as negative. Our findings indicate that when the LLM encounters new input patterns that were not part of its fine-tuning data, it can hallucinate by producing inaccurate technique information. In this scenario, since the LLM has not been exposed to negative training data during the fine-tuning, any negative content found in the CTI report is likely to elicit such responses.

We detect this hallucination by noting the discrepancies in the responses generated by the LLM when a high temperature parameter is applied. A higher temperature promotes the LLM to generate varied and imaginative outputs. This variability implies that the hallucinated responses can differ with each prompt, allowing us to identify cases where the LLM demonstrates uncertainty in its answers. In our methodology, we define a value n, and we perform technique detection on the input text n times while using a high temperature value of 1.0. We then compute how many times each specific technique is recognized. Ultimately, only those techniques that are detected at least half of the time are considered correct. If none of the output techniques satisfy this rule for the input text, we classify the text as a negative case (with no technique information found). This process is illustrated in Figure 7.6.

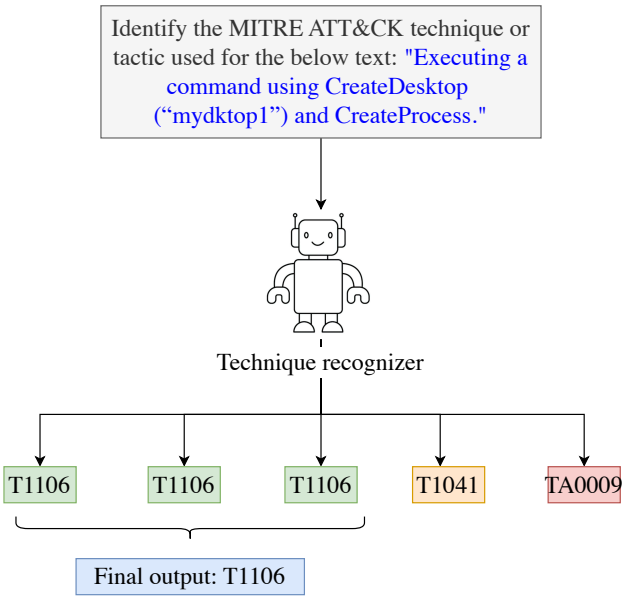


Figure 7.6: Technique/tactic detection with five attempts; the final output is T1106.

7.3 Evaluation

In this section, we present our preliminary evaluation of the methodology for analyzing reports using LLMs. In this section, we present our preliminary evaluation of the approach. Firstly, we show the actual output from our system for a typical CTI report as a case study. After that, we evaluate the performance metrics with respect to five CTI reports.

7.3.1 Case Study of a CTI Report

We use a CTI report on the Md_client backdoor as a case study to illustrate the actual analysis outcomes of the proposed methodology in Chapter 7. The Md_client tool is proprietary software created by a Chinese Advanced Persistent Threat (APT) group, which targets countries in Southeast Asia as part of its operational efforts. The information provided here is derived from a detailed whitepaper published by Bitdefender [49].

We first divide the mentioned CTI report into smaller sections and carry out the technique detection using our LLM-based technique recognizer. For each segment of text, we perform 30 attempts. We record the output technique/tactic for each trial to determine the final technique/tactic linked to the input text. To be considered valid, the resulting technique or tactic must

be the one that occurs most frequently among the 30 attempts and must appear more than 15 times. Due to these criteria, there are occasions where the output technique or tactic for a specific text is noted as None. The initial outcomes of this report analysis are presented in Figure 7.7.

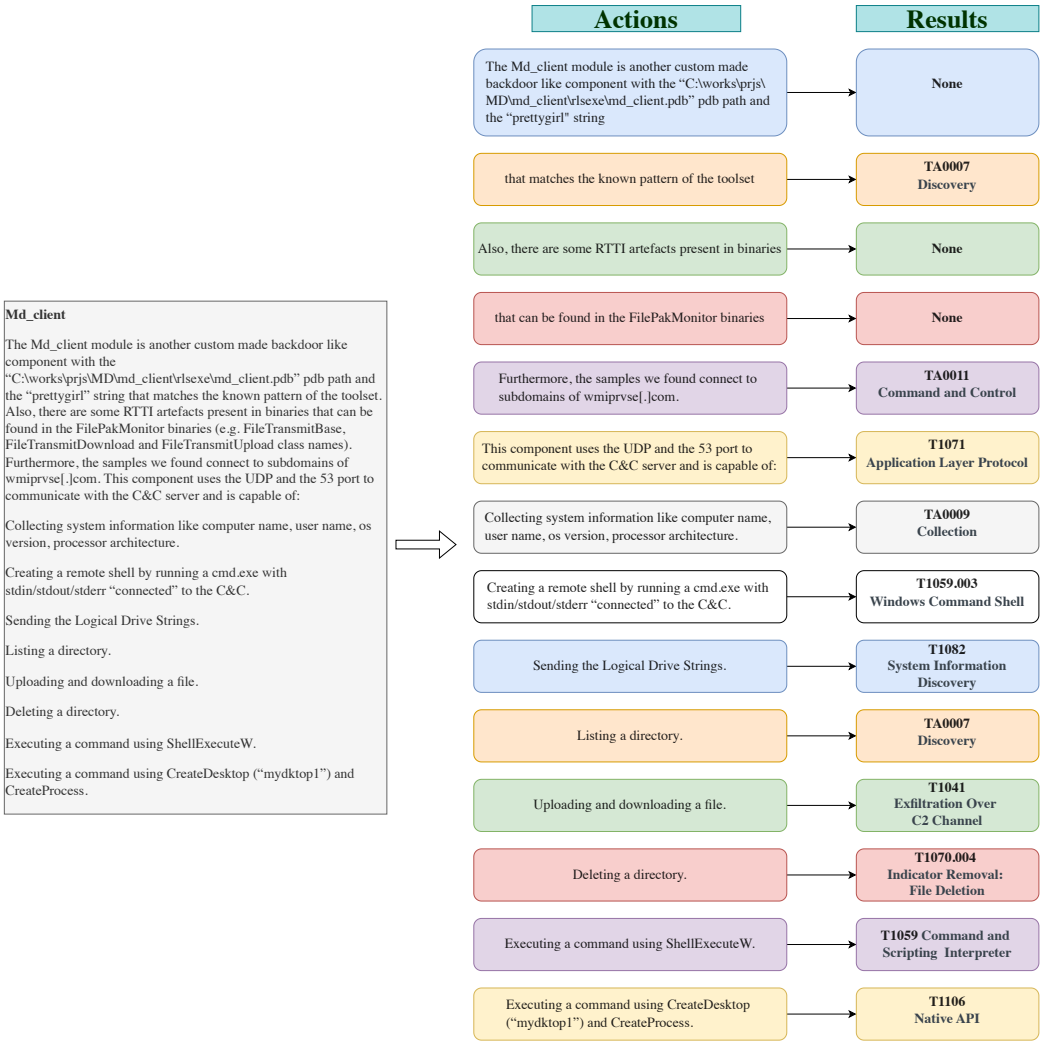


Figure 7.7: An example of the output analysis results for Md_client backdoor. The label **None** for Results means that there is no cyberattack technique information associated with the text.

After completing the report segmentation, it becomes clear from Figure 7.7 that a chronological list of actions arises, reflecting the order in which the text appears in the report. This ordered arrangement represents the common writing style favored by cybersecurity bloggers. However, it is crucial

to recognize that more intricate scenarios can occur where the techniques discussed in the report do not follow this sequential structure. For now, our primary emphasis will be on this common pattern to facilitate the recognition of the attacking techniques in the CTI report.

7.3.2 Evaluation Results

In this evaluation, we analyze the metrics associated with five CTI reports from different sources [48, 49, 51] using the ground-truth data supplied by RAF-AG [78] (see Chapter 4). We evaluate the predictions produced by our method against the ground truth data to compute the F1 score. In this scenario, True Positives (TP) are identified as cases that appear in both the prediction set and the ground truth. False Positives (FP) are cases that are found in the prediction set but not in the ground truth, while False Negatives (FN) are instances that exist in the ground truth but are missing from the prediction set.

As shown in Table 7.1, our approach achieves an F1 score of 0.716, which is quite close to the 0.784 obtained by RAF-AG. Importantly, our method exhibits a higher Recall of 0.887 compared to RAF-AG's Recall of 0.768, mainly due to a lower occurrence of False Negative (FN) cases. However, the notably high count of False Positives (FP) in our method leads to a lower Precision value (0.615) compared to the 0.802 reported by RAF-AG.

Table 7.1: Performance comparison of RAF-AG and the CyLLM-based approach presented in this chapter. We evaluate the performance of 5 CTI reports.

CTI report	**RAF-AG**						**CyLLM-based approach**					
	TP	**FP**	**FN**	**Precision**	**Recall**	**F1**	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1**
Frankenstein Campaign	12	0	0	1	1	1	10	3	2	0.769	0.833	0.8
JMT Trading (macOS)	3	3	3	0.5	0.5	0.5	6	7	0	0.462	1	0.632
Md_client backdoor	5	0	1	1	0.833	0.909	6	4	0	0.6	1	0.75
NavRAT trojan	6	4	4	0.6	0.6	0.6	6	9	4	0.4	0.6	0.48
SolarWinds Compromise	10	1	1	0.909	0.909	0.909	11	2	0	0.846	1	0.917
Average	7.2	1.6	1.8	0.802	0.768	**0.784**	7.8	5	1.2	0.615	0.887	**0.716**

In our examination of the results, we find that the high occurrence of FP cases largely results from the LLM technique recognizer's tendency to hallucinate, which stems from the absence of negative examples in the training dataset. This observation indicates that our current strategies for reducing false positives are not sufficiently effective. Consequently, it is essential for future research efforts to focus on tackling this significant issue.

In addition to the previously mentioned observations, we can conclude that the methods' performance fluctuates considerably across various reports. When the information in a report is clearly aligned with the MITRE ATT&CK framework, there is a higher likelihood that the LLM will accurately identify the corresponding technique. However, intricate techniques often require information from multiple actions rather than depending on a single action. In such cases, the LLM may find it challenging to correctly identify the technique. To overcome this obstacle, we need to implement a more sophisticated mechanism that enables the LLM to simultaneously consider multiple actions. For instance, we could consolidate a series of continuous actions into a single sequence and present that to the LLM for technique identification.

7.4 Summary

This chapter introduces a methodology for developing cybersecurity applications, where annotated data is insufficient. We then presented a report analysis approach as an example of this methodology. We utilized the data from the MITRE ATT&CK framework and RAF-AG's report analysis to establish the initial training data, then perform data augmentation to improve the diversity of language patterns. Subsequently, we conducted a fine-tuning process on the LLM using this augmented data to develop an LLM-based technique recognizer. This technique recognizer is then utilized to analyze CTI reports, which have been segmented into action-centric fragments.

In our evaluation, our approach achieved an F1 score of 0.716, which is comparable to the score of 0.784 reported by RAF-AG. However, the high false positive rate, resulting from training solely with positive examples, led to a lower precision value for our approach. This limitation is a significant drawback of this study.

In our future work, we plan to tackle the mentioned limitation to improve the performance of our study. Currently, we employ a 1-gram method, whereby each action is processed by the recognizer individually. We intend to implement an n-gram analysis, which will allow for the recognition of text containing consecutive actions.

Chapter 8

Conclusion

8.1 Summary

In this thesis, we presented a list of applications designed to support experts in cybersecurity tasks. The thesis started with a survey. From this survey, we recognized the challenges involved in implementing computer-based systems to solve cybersecurity problems. The methodologies used to develop these applications aim to address issues such as the insufficiency and rapid obsolescence of training data, which often requires significant involvement from experts to label them. The thesis successfully proposed these methodologies and instantiated them into practical applications, which have been published in the research community.

1. The first application is the RAF-AG framework, designed for analyzing CTI reports, with the goal of enhancing the information-sharing process among cybersecurity professionals. In developing this framework, we leverage recent advancements in NLP and DL, such as Weak Supervision and public text analysis tools, to avoid labeling the data from scratch. In evaluation, RAF-AG outperformed AttacKG, a similar report analysis framework, in precision, recall, and F1 scores, achieving values of 0.717, 0.722, and 0.708 compared to AttacKG's scores of 0.337, 0.535, and 0.393, respectively.

2. Generating ABAC policies to manage access in IT systems is a cumbersome expertise task. The second application in this thesis supports the work of cybersecurity experts in generating policies for ICS environments. This application utilizes commercial LLMs to leverage their strong generalizability, which helps avoid the need for text normalization. Additionally, it employs few-shot learning to reduce the large

amount of annotated data typically required. Our experiment with a typical ICS network showed that the generated policies with optimized priority values (used in policy conflict resolution) could obtain a high F1 score of 0.994.

3. To address potential issues with commercial LLMs, we developed the CyLLM-DAP framework to draw the research community's attention to open-source LLMs and domain-adaptive pre-training. CyLLM-DAP streamlines the implementation process for specializing open-source LLMs in cybersecurity. We developed cybersecurity-specific LLMs, namely CyLLMs, using the CyLLM-DAP framework. In our experiment, we observed that domain-adaptive pre-training in cybersecurity leads to performance improvements in line with those seen in other domains. For instance, in the Q&A task, CyLLMs demonstrated a performance gain of up to 4.75% compared to general models.

4. Inspired by RAF-AG and Cy-LLM, we developed a methodology to combine previous achievements into a methodology for cybersecurity problem-solving. We instantiated this methodology in an innovative report analysis. This methodology eliminates the need for text normalization, as required in RAF-AG, and does not require manual annotations by experts. Furthermore, it delivers performance comparable to that of RAF-AG.

Moreover, our dissertation successfully met the three objectives mentioned at the end of Section 1.1, as follows:

- We studied the feasibility of leveraging NLP and DL advancements for solving annotated data scarcity when developing cybersecurity applications. Based on this, we successfully developed methodologies and applications for various downstream tasks. In those presented applications, we could (1) utilize the available annotated data from other research with Weak Supervision, (2) replace cybersecurity relationships with grammatical ones with Sentence Dependency Tree, (3) utilize LLM's generalizability and few-shot learning to solve tasks with a small amount of annotated examples, and (4) implement transfer learning by pre-injecting cybersecurity knowledge into LLMs to be reused in various tasks.

- The cybersecurity applications developed throughout this thesis were published for the benefit of the research community. For instance, we developed RAF-AG for CTI report analysis and an access control policy

generator to produce ABAC policies for ICS networks. Our evaluations show that these applications generate high-quality outputs and deliver performance comparable to traditional approaches.

- We developed and published CyLLM-DAP—a cybersecurity-adaptive pre-training framework for LLMs—as well as CyLLMs created with it—cybersecurity-specific LLMs. Experts and users can use CyLLM-DAP to build their own private CyLLMs or directly utilize our published CyLLMs to address cybersecurity tasks. Additionally, we demonstrated how these methodologies and applications can be integrated into a cohesive framework for solving cybersecurity problems.

8.2 Future work

Potential future work directions for this thesis include:

1. In addition to the attack paths generated by RAF-AG, we can configure the framework to produce labeled entities, including cybersecurity phrases, relationships, and specific attack steps. These outputs can be valuable for future research in the field of cybersecurity, serving as a weak supervision data source for upcoming studies.

2. So far, we focused on a methodology for generating policies based on LLMs for ICS networks. Our approach relies on ground truth data to optimize the generated policies. We aim to develop a more generic policy generation solution that leverages commercial LLMs and can be adapted to work with any new system in the future. Additionally, we intend to reduce the reliance on extensive ground truth data, allowing for a smaller amount of labeled data to be utilized in the optimization process for the generated policies.

3. Although CyLLM-DAP and CyLLMs provide valuable features for the research community, their model sizes remain relatively small, with fewer than 10 billion parameters. There is potential for integrating additional techniques into CyLLM-DAP, and the development of larger, cybersecurity-specific models is also possible. By collecting larger and more diverse cybersecurity datasets, we can ensure that these models stay up-to-date with emerging knowledge in the field. Additionally, developing a multi-purpose instruction fine-tuned CyLLM that can support various cybersecurity downstream tasks simultaneously is worth considering for future work.

Appendix A

RAF-AG's Supplementary Materials

A.1 Text Normalization in RAF-AG

This section of the Appendix outlines the key text normalization functions in RAF-AG. These functions transform raw text into normalized output, which is generated by applying a series of normalization steps before the text is sent to subsequent analysis modules.

Unicode Fixing RAF-AG uses available open-source NLP tools (e.g., SpaCy) for English text analysis. The presence of some Unicode characters may interrupt the normal workflow of these tools, such as failing to properly tokenize the text into correct words/tokens due to the non-breaking space Unicode character "\xa0". To resolve this, we first transform Unicode characters into their equivalent ASCII representations. In case of unsuccessful transformation, we remove the Unicode characters. Once the Unicode is corrected, there might be a risk of losing some information. In the analysis process of the RAF-AG report, the fundamental action involves comparing phrases from the report with data from its knowledge base (refer to Section 4.2.2.3). Since the Unicode correction function is uniformly applied to CTI reports and all other text-based materials (such as procedure examples) within RAF-AG, the effect of information loss is reduced.

Text Simplification English contains phrases that convey similar meanings, yet their grammatical structures vary. Consequently, this can result in notable distinctions in the generated dependency trees of the sentences, even when the words or phrases remain unchanged. For example, although "We

must update the system" and "We have to update the system" carry comparable meanings, their dependency trees are different. In the former, the primary verb is "update," while in the latter, it is "have." This necessitates the use of two distinct grammar rules to address such dependency trees.

A set of grammar rules is initially defined and used in a tree traversal algorithm to extract cybersecurity relations from the sentence dependency tree. A limited set of grammar rules can not cover all grammatical cases. We decide to simplify the text to improve the system's performance in two ways: (1) reduce the complexity of the dependency tree and time complexity of tree traversal, and (2) reduce the variations of these dependency trees to use a small number of grammar rules effectively.

To achieve this, we begin by creating a basic dictionary in which complex terms (for instance, "have to") serve as keys, and their simpler forms (such as "must") are the corresponding values. Based on this dictionary, we perform a string search to replace instances of complex keys with their corresponding values, simplifying the text.

Subject Ellipsis Handling The phenomenon of subject ellipsis, where sentences lack a main subject, is frequently encountered in English writing. In our analysis of CTI reports, we found that such sentences commonly appear in bullet points. These bullet points are usually preceded by trigger sentences that include specific phrases and symbols (for example, the expression "as follows" and/or a colon, ":"). In RAF-AG, cybersecurity events are derived from the input text, where all three components (Subject, Verb, Object) are identified for each event to be accurately established. When subject ellipsis occurs, the missing subject element may prevent the extraction of cybersecurity events from these sentences. Satvat et al. [7] suggest that subject ellipsis should be taken into account during cybersecurity text analysis. They also introduced a dependable method for addressing this issue. To address the mentioned issue of subject ellipsis, RAF-AG follows the below steps:

1. Detect subject ellipsis sentences in the document. They are sentences that come with a main verb but no main subject.

2. Determine the trigger sentence. The trigger of a subject ellipsis sentence is the nearest preceding sentence in the text with pre-defined patterns (e.g., "as follows", an ending colon).

3. Recognize the potential subjects and objects from the trigger sentence. The distances between them and the target sentence are also calculated. The subject or object that is closest in distance is kept as the

appropriate subject for the sentence with subject ellipsis. According to our observations, the appropriate subject is typically the pronoun "it" in the trigger sentence, which implies malicious entities (such as attackers or malware).

4. The appropriate subject is prepended to the subject ellipsis sentence. Subsequently, the sentence's main verb is modified accordingly.

Although this approach enables RAF-AG to extract events from these sentences, it may inadvertently alter the context of the subject ellipsis sentence. In certain instances, this could lead to the generation of new cybersecurity events. Given the infrequent occurrence of subject ellipsis and the various methods of similarity measurement utilized in RAF-AG (including node similarity, edge similarity, verb similarity, and graph similarity), the risk of producing nonexistent cybersecurity events is negligible compared to the advantages offered by this function.

IoC Replacement An Indicator of Compromise (IoC) refers to a type of forensic information, such as an IP address or domain name, that provides valuable insights for identifying malicious activities. However, more intricate IoCs (like registry paths or directories) can diminish the effectiveness of NLP tools. When text that includes these IoCs is processed by NLP tools, the tokenization step separates them into smaller elements prior to analysis, resulting in a sentence dependency tree that is overly complex. RAF-AG conducts the IoC replacement for the following reasons:

1. We want to extract the IoCs during the replacement for later analysis.

2. We want to expedite the parsing process of NLP tools. The complex IoC is replaced with placeholders. Thus, NLP tools treat it as a single entity to run more efficiently.

3. We aim to simplify the sentence dependency tree, leading to a faster performance of the tree traversal algorithm.

To achieve this goal, we initially create a set of regular expression (regex) patterns designed to identify these IoCs within cybersecurity-related text. In RAF-AG, we focus on six primary types of IoCs, which include file and directory paths, registry and DLL paths, email addresses, websites, IP addresses, and vulnerability IDs. Following this, we substitute these IoCs (for example, "abc@mail.com") with appropriate placeholders (for instance, EMAIL). Generally, the grammatical function of most IoCs in a sentence is that of nouns.

These placeholders are presented in uppercase (such as EMAIL, REGISTRY) to be treated as proper nouns by NLP tools. This approach guarantees that the grammatical function of these IoCs remains intact, preserving the primary architecture of the sentence dependency tree. Once parsing is complete, we restore the original IoCs by performing a reverse replacement. This system facilitates the effective subsequent analysis of those IoCs in RAF-AG.

However, for a specific set of regex patterns, RAF-AG is unable to address all instances of IoC representation within the text. While unrecognized IOCs may be split into several tokens, NLP tools such as SpaCy [2] could categorize them as noun phrases. Based on this insight, RAF-AG also attempts to extract entire noun phrases and employs keywords (for example, "HKLM" for registry paths) to detect these IoCs. This extraction occurs during the entity labeling phase (see Section 4.2.1.3). In the most unfavorable scenario, the unrecognized IoCs are disregarded in further analysis.

A.2 Graph Alignment Threshold in RAF-AG

In RAF-AG, the graph alignment threshold serves as the crucial hyperparameter. It regulates the population size of the GARs during the coordination phase of the attack step (refer to Section 4.2.3.2). A lower graph alignment threshold can ease the coordination, resulting in a greater number of false positives (FPs). Conversely, a higher threshold may lead to an increase in false negatives (FNs), as it could eliminate potential techniques.

To identify the best threshold for maximizing the F1 score, we examined the performance of RAF-AG at multiple threshold levels between 0.8 and 1.0. As illustrated in Figure A.1, the threshold of 0.87 yields the highest F1 score, which is why we chose it as the optimal threshold.

From Figure A.1, we can see that the precision curve based on the alignment threshold is not monotonic. This outcome is anticipated due to the operational mechanism of RAF-AG. In our approach, every technique is characterized by a collection of GARs sharing the same technique ID. A false positive technique will only be excluded from the final attack path if all GARs related to that technique are also excluded. While increasing the threshold might result in the removal of more GARs, it doesn't guarantee the successful elimination of false positive techniques, which may lead to no improvement in the precision value.

Furthermore, the precision curve is unable to achieve a value of 1.0, even when an extremely high graph alignment threshold is applied. When RAF-AG employs a threshold at its maximum (for instance, 1.0), it may produce an attack path that is empty (lacking any TP techniques) due to the com-

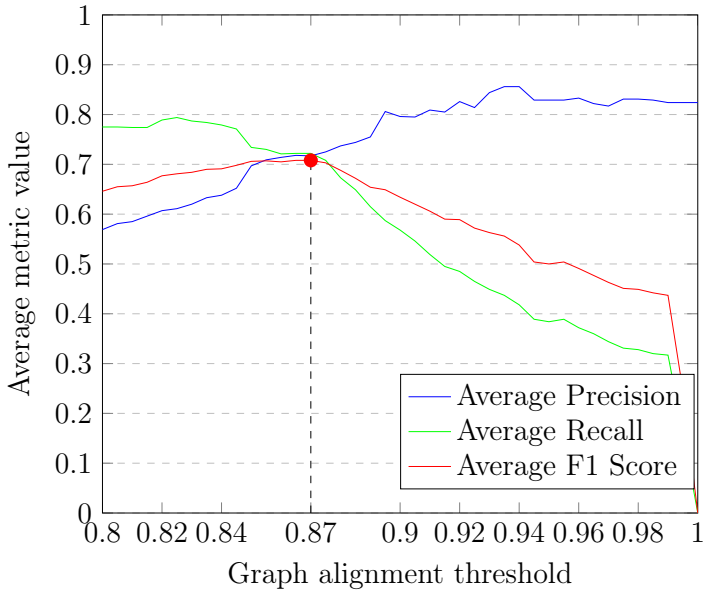


Figure A.1: The determination of the graph alignment threshold across a set of 30 CTI reports is based on average precision, recall, and the F1 score. When the graph alignment threshold is set at 0.87, the F1 score reaches its maximum, as represented by the large red dot.

plexity of some CTI reports. Consequently, the average precision value in our assessment did not attain a value of 1.0.

According to this discussion, we recommend that RAF-AG users set the graph alignment threshold to 0.87 as a standard value. Nevertheless, as they become more familiar with the framework, they may adjust this value to better suit their particular needs and context.

A.3 Cybersecurity Expertise Cost

This section discusses the cost of cybersecurity expertise in the implementation and maintenance of RAF-AG.

A.3.1 Expert Involvement in Implementation Stage

In addition to creating attack paths through report analysis, the RAF-AG seeks to minimize the need for expert participation in the framework's implementation. Table A.1 outlines the estimated levels of expertise necessary for executing the functions of RAF-AG. As indicated, the function that demands

the highest level of cybersecurity-specific expertise is Entity Labeling.

In traditional supervised DL, specialists manually label thousands of text sentences to create training data for DL models. In RAF-AG, we utilize Weak Supervision-based entity labeling to leverage existing resources from other cybersecurity studies, including IoC patterns and keywords. Nevertheless, cybersecurity experts need to be involved to ensure accurate entity labeling.

Table A.1: The level of cybersecurity knowledge necessary for implementing RAF-AG's functionalities.

No.	Functionality	Level	Description
1	Text Normalization	Low	Generic work: - The use of Spacy framework for text analysis - The knowledge to work with English grammar Expertise work: - Acceptance testing
2	Entity Labeling	Medium	Generic work: - The Snorkel framework's utilization for entity labeling Expertise work: - Survey conduction regarding Weak Supervision sources - Determination of cybersecurity categories - Classification of the collected keywords and patterns
3	Event Extraction	Low	Generic work: - Use of SpaCy and grammar rules for event extraction Expertise work: - Grammar rule generation
4	Graph Building and Alignment	Low to Medium	Generic work: - Function implementation Expertise work: - Knowledge related to verbs, strong verbs, soft verbs - Verb classification - Setting the initial hyperparameters
5	Attack Path Generation	Low	Generic work: - Function implementation Expertise work: - Heuristic rule generation

A.3.2 Expert Involvement in Maintenance Stage

Through an examination of relevant studies in the field of cybersecurity, we found that incorporating the newest cybersecurity knowledge into the framework is a challenging task. For instance, the ontology of threat actions discussed in [5] and the threat models presented in [8] necessitate considerable manual effort. In the case of AttacKG [6], updating the knowledge base involves analyzing thousands of current CTI reports to generate the technique templates.

To incorporate the latest cybersecurity knowledge, RAF-AG does not require substantial expert involvement, as follows:

- Updating the ATT&CK knowledge base: This can be done by just replacing the STIX file for the most current version, since it includes all the necessary information.
- Updating the list of procedure graphs: RAF-AG is capable of automatically retrieving a list of procedure examples from the ATT&CK database file and subsequently reanalyzing them to generate its internal database of procedure graphs.
- Updating the entity labeling: The entity labeling procedure in RAF-AG relies on a collection of keywords, unique phrases, and regex patterns. When significant changes occur in the cybersecurity domain, experts may be required to contribute new sources for emerging systems/software, along with patterns to identify the new types of IoCs.
- Updating the NLP library: Given the significant advancements in English-related NLP research, it should be fairly straightforward to enhance the NLP tools for improved analysis of English text.

Appendix B

Cybersecurity-specific LLM Development

B.1 Data Preparation

The training data utilized in the specialization of LLM is gathered through CyLLM-DAP with the standard settings. Table B.1 presents the sources from which we have collected data. The data gathering procedure was carried out over a span of six months (from January to June 2024) using four high-speed internet-connected computing nodes. As indicated in Table B.1, no code data is included, as our focus is not on addressing code-related downstream tasks. We intend to incorporate code data for tasks related to coding in our future projects.

Table B.1: Cybersecurity text data for domain-adaptive pre-training.

Data sources	**Size**	**Original Sources**
Wikipedia	∼500 MB	Wikipedia dataset on HuggingFace hub
Academic papers	∼3 GB	S2OCR
Books	∼200 MB	Online book libraries
Web data	∼26 GB	Common Crawl RedPajama
Total	∼30 GB	

Using CyLLM-DAP, various data filters are applied to the dataset to ensure its quality and relevance. These include:

1. Language filtering: In this dataset, only text written in the English language is kept.

2. Relevance filtering: During the data collection, the preliminary relevance filters (with keywords and patterns) are applied. Model-based filtering is also used as a separate module. Regarding academic papers, we use the API search function provided by the S2OCR authors to obtain pertinent documents related to a list of cybersecurity keywords.

3. Quality filtering: We utilize the default metrics and rules implemented by CyLLM-DAP for the cybersecurity domain. Note that we also remove toxic documents from the dataset during this stage.

4. Deduplication: Duplicated documents are also removed from the dataset by CyLLM-DAP.

5. Anonymization: The default anonymization function of CyLLM-DAP is used.

B.2 Domain-adaptive Pre-training

As noted in Section 6.2.2.1, the field of cybersecurity intersects with other areas of knowledge without distinct boundaries. It is essential that language models specializing in the cybersecurity field do not lose their knowledge from other domains. For this reason, we refrain from fully training the model where all parameters are updated. Instead, we limit the number of parameters to be modified during the continual pre-training using LORA (Low-rank Adaptation) [76]. LORA is a widely recognized technique for working with LLMs in low-resource environments. It operates by freezing the model's weights and adding smaller trainable parameter matrices, ensuring that only a limited number of weights are adjusted to incorporate new information.

By freezing most model parameters during the domain-adaptive pre-training phase, we mitigate the risk of catastrophic forgetting. This helps to retain the model's pre-existing knowledge pertinent to other fields. Two critical hyperparameters, rank and alpha, govern the size of the trainable matrix in LORA training. We establish high values for these primary hyperparameters (64 and 128, respectively) to efficiently incorporate new knowledge into the models.

The domain-adaptive pre-training is conducted on H100 GPU computers, taking 123 hours for training CyLlama3 and 135 hours for CyMistral for a single epoch. Our methodology employs a context length of 1024 and a batch

size of 64 without applying quantization to the model. Quantization refers to a strategy intended to decrease GPU memory usage during LLM pre-training by transforming high-precision values into lower precision. Although this method can significantly cut down on GPU memory requirements, it may also adversely affect the quality of the pre-training phase. Given that our models fit within the GPU memory during training with LORA, we choose not to implement quantization to preserve the quality of the pre-training process.

The outcome of the domain-adaptive pre-training is two foundation LLMs tailored specifically for cybersecurity, referred to as CyLLMs. The CyLLMs are available in two different versions, each based on its specific foundational model. Specifically, CyLlama3 is based on the cybersecurity-focused version of Llama-8B-v3, while CyMistral3 is derived from Mistral-7B-v0.3. In the next section, we will conduct experiments on these CyLLMs to evaluate their effectiveness on downstream tasks.

Bibliography

[1] Blake E. Strom et al. *Mitre Att&ck: Design and Philosophy*. 2018.

[2] *spaCy: Industrial-Strength Natural Language Processing in Python*. Explosion AI, 2024. URL: https://github.com/explosion/spaCy.

[3] OpenAI et al. *GPT-4 Technical Report*. Version 6. 2023. DOI: 10.48550/ARXIV.2303.08774. URL: https://arxiv.org/abs/2303.08774. Pre-published.

[4] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] Ghaith Husari et al. "TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources". In: *Proceedings of the 33rd Annual Computer Security Applications Conference*. Orlando FL USA: ACM, Dec. 4, 2017, pp. 103–115. ISBN: 978-1-4503-5345-8. DOI: 10.1145/3134600.3134646.

[6] Zhenyuan Li et al. "AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports". In: *Computer Security – ESORICS 2022*. Ed. by Vijayalakshmi Atluri et al. Vol. 13554. Cham: Springer International Publishing, 2022, pp. 589–609. ISBN: 978-3-031-17139-0 978-3-031-17140-6. DOI: 10.1007/978-3-031-17140-6_29.

[7] Kiavash Satvat, Rigel Gjomemo, and V.N. Venkatakrishnan. "Extractor: Extracting Attack Behavior from Threat Reports". In: 2021 IEEE European Symposium on Security and Privacy (EuroS&P). Vienna, Austria: IEEE, Sept. 2021, pp. 598–615. ISBN: 978-1-66541-491-3. DOI: 10.1109/EuroSP51992.2021.00046.

[8] Wenjun Xiong et al. "Cyber Security Threat Modeling Based on the MITRE Enterprise ATT&CK Matrix". In: *Software and Systems Modeling* 21.1 (Feb. 2022), pp. 157–177. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-021-00898-7.

[9] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[10] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. Version 1. 2019. DOI: 10.48550/ARXIV.1907.11692.

[11] JunJun Chen et al. "Automatically Identifying Sentences with Attack Behavior from Cyber Threat Intelligence Reports". In: *2023 8th International Conference on Data Science in Cyberspace (DSC)*. Hefei, China: IEEE, Aug. 18, 2023, pp. 491–498. ISBN: 9798350331035. DOI: 10.1109/DSC59305.2023.00077.

[12] Md Tanvirul Alam et al. "Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI". In: *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. Hong Kong China: ACM, Oct. 16, 2023, pp. 92–108. ISBN: 9798400707650. DOI: 10.1145/3607199.3607208.

[13] Kashan Ahmed, Syed Khaldoon Khurshid, and Sadaf Hina. "CyberEntRel: Joint Extraction of Cyber Entities and Relations Using Deep Learning". In: *Computers & Security* 136 (Jan. 2024), p. 103579. ISSN: 01674048. DOI: 10.1016/j.cose.2023.103579.

[14] Marie-Catherine Marneffe et al. "Universal Stanford Dependencies: A Cross-Linguistic Typology". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 4585–4592.

[15] Masoud Narouei, Hamed Khanpour, Hassan Takabi, et al. "Towards a Top-down Policy Engineering Framework for Attribute-based Access Control". In: *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. Indianapolis Indiana USA: ACM, 2017, pp. 103–114. ISBN: 978-1-4503-4702-0. DOI: 10.1145/3078861.3078874. URL: https://dl.acm.org/doi/10.1145/3078861.3078874.

[16] John Heaps et al. “Access Control Policy Generation from User Stories Using Machine Learning”. In: *Data and Applications Security and Privacy XXXV*. Vol. 12840. Cham: Springer International Publishing, 2021, pp. 171–188. ISBN: 978-3-030-81241-6 978-3-030-81242-3. DOI: 10.1007/978-3-030-81242-3_10. URL: https://link.springer.com/10.1007/978-3-030-81242-3_10.

[17] Manar Alohaly, Hassan Takabi, and Eduardo Blanco. “Towards an Automated Extraction of ABAC Constraints from Natural Language Policies”. In: *ICT Systems Security and Privacy Protection*. Vol. 562. Cham: Springer International Publishing, 2019, pp. 105–119. ISBN: 978-3-030-22311-3 978-3-030-22312-0. DOI: 10.1007/978-3-030-22312-0_8. URL: https://link.springer.com/10.1007/978-3-030-22312-0_8.

[18] Carlos Cotrini, Thilo Weghorn, and David Basin. “Mining ABAC Rules from Sparse Logs”. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. London: IEEE, Apr. 2018, pp. 31–46. ISBN: 978-1-5386-4228-3. DOI: 10.1109/EuroSP.2018.00011. URL: https://ieeexplore.ieee.org/document/8406589/.

[19] Maryem Ait El Hadj et al. “ABAC Rule Reduction via Similarity Computation”. In: *Networked Systems*. Vol. 10299. Cham: Springer International Publishing, 2017, pp. 86–100. ISBN: 978-3-319-59646-4 978-3-319-59647-1. DOI: 10.1007/978-3-319-59647-1_7. URL: http://link.springer.com/10.1007/978-3-319-59647-1_7.

[20] Shohei Mitani et al. “Qualitative Intention-aware Attribute-based Access Control Policy Refinement”. In: *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies*. Trento Italy: ACM, 2023, pp. 201–208. ISBN: 9798400701733. DOI: 10.1145/3589608.3593841. URL: https://dl.acm.org/doi/10.1145/3589608.3593841.

[21] Ashish Vaswani et al. “Attention Is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Vol. 30. Curran Associates, Inc., 2017, pp. 6000–6010.

[22] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training*. 2018.

[23] Francesco Greco, Giuseppe Desolda, and Luca Viganò. “Supporting the Design of Phishing Education, Training and Awareness Interventions: An LLM-based Approach”. In: *2nd International Workshop on Cyber-Security Education for Industry and Academia*. 2024.

[24] Shaswata Mitra et al. *LOCALINTEL: Generating Organizational Threat Intelligence from Global and Local Cyber Knowledge*. 2024. arXiv: 2401.10036 [cs]. URL: http://arxiv.org/abs/2401.10036. Pre-published.

[25] Mohamed Amine Ferrag et al. *SecureFalcon: Are We There Yet in Automated Software Vulnerability Detection with LLMs?* May 29, 2024. arXiv: 2307.06616 [cs]. URL: http://arxiv.org/abs/2307.06616. Pre-published.

[26] Pei Yan et al. *Prompt Engineering-assisted Malware Dynamic Analysis Using GPT-4*. Dec. 13, 2023. arXiv: 2312.08317 [cs]. URL: http://arxiv.org/abs/2312.08317. Pre-published.

[27] Shams Tarek et al. *SoCureLLM: An LLM-driven Approach for Large-Scale System-on-Chip Security Verification and Policy Generation*. 2024.

[28] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. Version 2. 2023. DOI: 10.48550/ARXIV.2307.09288. URL: https://arxiv.org/abs/2307.09288. Pre-published.

[29] Albert Q. Jiang et al. *Mistral 7B*. Oct. 10, 2023. arXiv: 2310.06825 [cs]. URL: http://arxiv.org/abs/2310.06825. Pre-published.

[30] Chen Ling et al. *Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey*. Version 7. 2023. DOI: 10.48550/ARXIV.2305.18703. URL: https://arxiv.org/abs/2305.18703. Pre-published.

[31] Chaoyi Wu et al. *PMC-LLaMA: Towards Building Open-source Language Models for Medicine*. Aug. 25, 2023. arXiv: 2304.14454 [cs]. URL: http://arxiv.org/abs/2304.14454. Pre-published.

[32] Quanjun Zhang et al. "Pre-Trained Model-Based Automated Software Vulnerability Repair: How Far Are We?" In: *IEEE Transactions on Dependable and Secure Computing* (2023), pp. 1–18. ISSN: 1545-5971, 1941-0018, 2160-9209. DOI: 10.1109/TDSC.2023.3308897. URL: https://ieeexplore.ieee.org/document/10232867/.

[33] Ehsan Aghaei et al. *SecureBERT: A Domain-Specific Language Model for Cybersecurity*. Version 3. 2023. DOI: 10.48550/ARXIV.2204.02685. URL: https://arxiv.org/abs/2204.02685.

[34] Priyanka Ranade et al. "CyBERT: Contextualized Embeddings for the Cybersecurity Domain". In: *2021 IEEE International Conference on Big Data (Big Data)*. 2021, pp. 3334–3342. DOI: 10.1109/BigData52589.2021.9671824.

[35] Markus Bayer et al. "CySecBERT: A Domain-Adapted Language Model for the Cybersecurity Domain". In: *ACM Trans. Priv. Secur.* 27.2 (Apr. 2024). ISSN: 2471-2566. DOI: 10.1145/3652594. URL: https://doi.org/10.1145/3652594.

[36] Jie Zhang et al. *When LLMs Meet Cybersecurity: A Systematic Literature Review*. Version 1. 2024. DOI: 10.48550/ARXIV.2405.03644. URL: https://arxiv.org/abs/2405.03644. Pre-published.

[37] Nan Jiang et al. *Nova: Generative Language Models for Assembly Code with Hierarchical Attention and Contrastive Learning*. June 24, 2024. arXiv: 2311.13721 [cs]. URL: http://arxiv.org/abs/2311.13721. Pre-published.

[38] Thomas L. Nielsen et al. "The CAPEC Database". In: *Journal of Chemical & Engineering Data* 46.5 (Sept. 1, 2001), pp. 1041–1044. ISSN: 0021-9568, 1520-5134. DOI: 10.1021/je000244z.

[39] Slav Petrov, Dipanjan Das, and Ryan McDonald. "A Universal Part-of-Speech Tagset". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 2089–2096.

[40] Alexander J. Ratner et al. "Snorkel: Fast Training Set Generation for Information Extraction". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. Chicago Illinois USA: ACM, May 9, 2017, pp. 1683–1686. ISBN: 978-1-4503-4197-4. DOI: 10.1145/3035918.3056442.

[41] Max Bachmann. *Levenshtein Ratio*. RapidFuzz, 2024. URL: https://github.com/rapidfuzz/Levenshtein.

[42] Xuren Wang et al. "APTNER: A Specific Dataset for NER Missions in Cyber Threat Intelligence Field". In: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD). Hangzhou, China: IEEE, May 4, 2022, pp. 1233–1238. ISBN: 978-1-66540-527-0. DOI: 10.1109/CSCWD54268.2022.9776031.

[43] Armin Buescher. *IoC Parser*. 2024. URL: https://github.com/armbues/ioc%20parser.

[44] Hudson Richard. *Coreferee*. Version 1.4.1. 2024. URL: https://github.com/richardpaulhudson/coreferee.

[45] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". In: *International Conference on Learning Representations*. 2020. DOI: 10.48550/arXiv.1904.09675.

[46] Daniel Cer et al. "Universal Sentence Encoder for English". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029.

[47] Nick Biasini. *Frankenstein Campaign*. It's alive: Threat actors cobble together open-source pieces into monstrous Frankenstein campaign, 2019. URL: https://blog.talosintelligence.com/frankenstein-campaign/.

[48] Talos Intelligence. *Cisco Talos Intelligence's Blog*. 2024. URL: https://blog.talosintelligence.com/.

[49] *Bitdefender*. 2024. URL: https://www.bitdefender.com/.

[50] *Malwarebytes*. malwarebytes, 2024. URL: https://www.malwarebytes.com/.

[51] *CISA News and Events*. 2024. URL: https://www.cisa.gov/news-events.

[52] *Threatpost*. Threatpost, 2024. URL: https://threatpost.com/.

[53] *The DFIR Report*. thedfirreport, 2024. URL: https://thedfirreport.com/.

[54] *ESET Research*. ESET, 2024. URL: https://www.welivesecurity.com/.

[55] *McAfee Labs*. McAfee, 2024. URL: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/.

[56] *Mandiant*. Mandiant, 2024. URL: https://www.mandiant.com/.

[57] *BlackBerry Blog*. BlackBerry, 2024. URL: https://blogs.blackberry.com/.

[58] *Cyber Tzar*. Cyber Tzar, 2024. URL: https://planet.cybertzar.com/.

[59] LI Zhenyuan. *Knowledge-enhanced-Attack-Graph*. 2024. URL: https://github.com/li-zhenyuan/Knowledge-enhanced-Attack-Graph.

[60] Manar Alohaly, Hassan Takabi, and Eduardo Blanco. "Automated Extraction of Attributes from Natural Language Attribute-Based Access Control (ABAC) Policies". In: *Cybersecurity* 2.1 (2019), p. 2. ISSN: 2523-3246. DOI: 10.1186/s42400-018-0019-2. URL: https://cybersecurity.springeropen.com/articles/10.1186/s42400-018-0019-2.

[61] Jules White et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. Version 1. 2023. DOI: 10.48550/ARXIV.2302.11382. URL: https://arxiv.org/abs/2302.11382.

[62] Junlin Wang, Jue Wang, et al. *Mixture-of-Agents Enhances Large Language Model Capabilities*. June 7, 2024. arXiv: 2406.04692 [cs]. URL: http://arxiv.org/abs/2406.04692. Pre-published.

[63] Alice Smith and Bob Johnson. *code_diff: Fast AST Based Code Differencing in Python*. https://github.com/username/code_diff. Version 1.0. 2024.

[64] OpenStack Security Group (OSSG). *Bandit: Security Analyzer for Python Code*. https://github.com/PyCQA/bandit. Version 1.7.10. OpenStack Foundation, 2024.

[65] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. https://GitHub.com/FacebookResearch/Nevergrad. 2018.

[66] Jay M. Patel. "Introduction to Common Crawl Datasets". In: *Getting Structured Data from the Internet*. Berkeley, CA: Apress, 2020, pp. 277–324. ISBN: 978-1-4842-6575-8 978-1-4842-6576-5. DOI: 10.1007/978-1-4842-6576-5_6. URL: http://link.springer.com/10.1007/978-1-4842-6576-5_6.

[67] Jeffrey Li et al. *DataComp-LM: In Search of the next Generation of Training Sets for Language Models*. June 20, 2024. arXiv: 2406.11794 [cs]. URL: http://arxiv.org/abs/2406.11794. Pre-published.

[68] Adrien Barbaresi. "Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction". In: *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2021, pp. 122–131. URL: https://aclanthology.org/2021.acl-demo.15.

[69] Kyle Lo et al. "S2ORC: The Semantic Scholar Open Research Corpus". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, pp. 4969–4983. DOI: 10.18653/v1/2020.acl-main.447. URL: https://www.aclweb.org/anthology/2020.acl-main.447.

[70] Clément Delangue. *Hugging Face hub*. https://huggingface.co/. Hugging Face, Inc., 2024. URL: https://huggingface.co/.

[71] TogetherAI. *RedPajama: an Open Dataset for Training Large Language Models*. 2023. URL: https://github.com/togethercomputer/RedPajama-Data.

[72] Katherine Lee et al. "Deduplicating Training Data Makes Language Models Better". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 8424–8445. DOI: 10.18653/v1/2022.acl-long.577. URL: https://aclanthology.org/2022.acl-long.577.

[73] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. "Fast locality-sensitive hashing". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2011, pp. 1073–1081.

[74] Microsoft. *Microsoft Presidio - Data Protection and Anonymization SDK*. https://microsoft.github.io/presidio/. 2024. URL: https://github.com/microsoft/presidio.

[75] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. 2023.

[76] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. Version 2. 2021. DOI: 10.48550/ARXIV.2106.09685. URL: https://arxiv.org/abs/2106.09685. Pre-published.

[77] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[78] Khang Mai et al. "RAF-AG: Report Analysis Framework for Attack Path Generation". In: *Computers & Security* 148 (Jan. 2025), p. 104125. ISSN: 01674048. DOI: 10.1016/j.cose.2024.104125. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404824004309.

[79] *Industrial-strength Natural Language Processing (NLP) in Python*. URL: https://github.com/explosion/spaCy.

Publications

Journal Papers

[1] Khang Mai et al. "RAF-AG: Report Analysis Framework for Attack Path Generation". In: *Computers & Security* 148 (Jan. 2025), p. 104125. ISSN: 01674048. DOI: 10.1016/j.cose.2024.104125. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404824004309.

Conference Papers (Peer Reviewed)

[2] Khang Mai et al. "LLM-based Fine-grained ABAC Policy Generation". In: *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*. INSTICC. Porto city, Portugal: SciTePress, 2025, pp. 204–212. ISBN: 978-989-758-735-1. DOI: 10.5220/0013225500003899.

[3] Khang Mai, Razvan Beuran, and Naoya Inoue. "CyLLM-DAP: Cybersecurity Domain-Adaptive Pre-training Framework of Large Language Models". In: *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*. INSTICC. Porto city, Portugal: ICISSP2025, 2025, pp. 24–35. ISBN: 978-989-758-735-1. DOI: 10.5220/0013094800003899.

Conference Papers (Non Peer Reviewed)

[4] Khang Mai et al. "Attack Path Extraction via Semi-Automatic Analysis of Cyber Threat Reports". In: *SCIS2024*. 2024 Symposium on Cryptography and Information Security. Nagasaki City, Japan: SCIS2024, 2024.

[5] Khang Mai, Naoya Inoue, and Razvan Beuran. "LLM-based Cyber Threat Intelligence Report Analysis". In: *SCIS2025*. Kitakyushu, Fukuoka, Japan: SCIS2025, 2025.

Note: Best Student Paper Award

The paper **[3]** received the Best Student Paper Award.