TECHNIQUERAG: Retrieval Augmented Generation for Adversarial Technique Annotation in Cyber Threat Intelligence Text

Ahmed Lekssays[1], **Utsav Shukla**[2], **Husrev Taha Sencar**[1], **Md Rizwan Parvez**[1]

[1]Qatar Computing Research Institute, Doha, Qatar, [2]Independent Researcher

alekssays@hbku.edu.qa, utsavshuk@gmail.com, hsencar@hbku.edu.qa, mparvez@hbku.edu.qa

Abstract

Accurately identifying adversarial techniques in security texts is critical for effective cyber defense. However, existing methods face a fundamental trade-off: they either rely on generic models with limited domain precision or require resource-intensive pipelines that depend on large labeled datasets and task-specific optimizations—such as custom hard-negative mining and denoising—resources rarely available in specialized domains. We propose **TECHNIQUERAG**, a domain-specific retrieval-augmented generation (RAG) framework that bridges this gap by integrating off-the-shelf retrievers, instruction-tuned LLMs, and minimal text–technique pairs. First, our approach mitigates data scarcity by fine-tuning only the generation component on limited in-domain examples, circumventing resource-intensive retrieval training. Second, although conventional RAG mitigates hallucination by coupling retrieval and generation, its dependence on generic retrievers often introduces noisy candidates, thereby limiting domain-specific precision. To address, we enhance the retrieval quality and domain specificity through a zero-shot LLM re-ranking that explicitly aligns retrieved candidates with adversarial techniques. Experiments on multiple security benchmarks demonstrate that TECHNIQUERAG achieves state-of-the-art performances without extensive task-specific optimizations or labeled data, while comprehensive analysis provides further insights.

Example of text to *(sub-)techniques* annotated pairs

Monero miner scripts are downloaded from TeamTNT's server and piped to "bash" using a SSH session as the "root" user with private key from "/tmp/TeamTNT."

1. T1098.004: SSH Authorized Keys
2. T1195: Supply Chain Compromise
3. T1059.004: Unix Shell
4. T1021.004: Remote Services: SSH
5. T1552.004: Private Keys

Figure 1: Example of MITRE ATT&CK *techniques* and *sub-techniques* highlighted in text with corresponding colored (implicit) indicators. IDs with "." denote *sub-techniques* (e.g., T1098.004).

1 Introduction

Uncovering new adversarial behaviors is critical for strengthening defenses against rapidly evolving cyber threats. These behaviors, defined by the tools, techniques, and procedures used by attackers, reveal how adversaries plan and execute attacks and impact systems and data. By identifying and analyzing the traces or artifacts left behind, security analysts can map low-level actions to higher-level concepts, such as *tactics* (i.e., strategic objectives like "lateral movement"), *techniques* or fine-grained *sub-techniques* (i.e., tactical methods like "Debugger Evasion"), and *procedures* (i.e., implementation details like "using PowerShell for credential dumping"). The findings are shared with other experts through public channels and threat intelligence services via security reports and detailed descriptions, to strengthen defenses, anticipate possible threats, and improve incident response.

The MITRE ATT&CK framework has established itself as the industry standard for categorizing and mapping adversarial behaviors (Corporation, 2022). This framework provides a comprehensive knowledge base of adversarial *tactics, techniques*, and *procedures (TTPs)*, built from real-world threat intelligence and incident response data. It uses a hierarchical matrix structure to systematically organize and classify adversary behaviors. The broad adoption of the ATT&CK framework presents a significant operational challenge: security analysts must manually map ambiguous threat descriptions from incident reports (such as the example shown in Fig 1) to standardized ATT&CK *(sub)-techniques*—a time-intensive process that demands expert knowledge. This manual task has motivated research into automated adversarial *technique* identification, which aims to systemati-

Findings of the Association for Computational Linguistics: ACL 2025, pages 20913–20926
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

cally label text segments with their corresponding ATT&CK *technique* and *sub-technique* IDs.

Prior approaches for *(sub-)technique* annotation adopt two primary paradigms: (1) Multi-class classification that directly map text to *(sub-)technique* IDs (You et al., 2022; for Threat-Informed Defense, 2023), which, while straightforward to implement, struggle with class imbalance and require extensive labeled training data; and (2) Retrieval/ranking approaches that evaluate semantic similarity between the text and *(sub-)techniques*. Early methods like Ladder (Alam et al., 2023) & AttackKG (Li et al., 2022a) introduce basic similarity-based ranking. Text2TTP (Kumarasinghe et al., 2024) advanced this through hierarchical re-ranking with fine-tuned embeddings, while NCE (Nguyen et al., 2024) improved similarity learning using dual-encoder architectures. Recently, IntelEX (Xu et al., 2024) employed LLMs in both retrieval and zero-shot learning settings to assess *(sub-)technique* relevance.

Although promising, the current methods are constrained by a critical trade-off: they either rely on general-purpose models lacking domain expertise or require large-scale labeled datasets and computationally intensive training pipelines. Retrieval approaches, for instance, require extensive hard-negative mining to distinguish fine-grained *(sub-)techniques* while classification models demand curated, balanced training sets—resources rarely available in this specialized domain. Compounding this issue, despite MITRE ATT&CK framework defines over 550 adversarial *(sub-)techniques*, only approximately 10,000 annotated examples are publicly available (Kumarasinghe et al., 2024; Nguyen et al., 2024), severely limiting generalization.

To address these dual challenges, we propose **TECHNIQUERAG**, a domain-specific retrieval-augmented generation (RAG) framework for *(sub-)techniques* annotation task that bridges generic and specialized models while eliminating dependency on resource-intensive pipelines or large labeled data. Unlike conventional approaches, TECHNIQUERAG integrates three key components: (a) off-the-shelf retrievers for candidate extraction (b) instruction-tuned LLMs to re-rank candidate *(sub-)techniques* (c) minimal text-*(sub-)technique* pairs used exclusively for fine-tuning the generator. Our approach mitigates data scarcity by fine-tuning only the generation component on limited in-domain examples while leveraging a novel re-ranking framework that uses generic off-the-shelf LLMs without fine-tuning to explicitly align retrieved candidates with adversarial *(sub-)techniques*, thereby enhancing domain specificity. Fig 2 shows an overview.

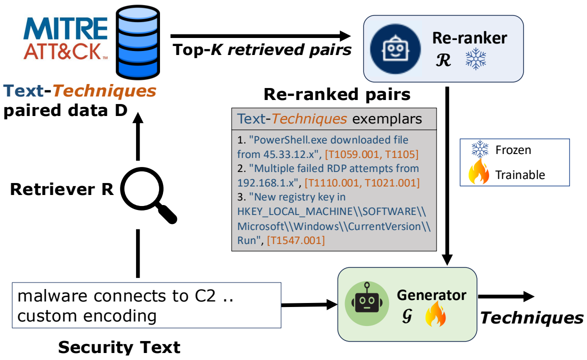

MITRE ATT&CK
Top-K retrieved pairs
Re-ranker
$\mathcal{R}$
Text-Techniques
paired data D
Re-ranked pairs
Text-Techniques exemplars
1. "PowerShell.exe downloaded file from 45.33.12.x", [T1059.001, T1105]
2. "Multiple failed RDP attempts from 192.168.1.x", [T1110.001, T1021.001]
3. "New registry key in HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", [T1547.001]
Frozen
Trainable
Retriever R
malware connects to C2 .. custom encoding
Generator
$\mathcal{G}$
Techniques
Security Text

Figure 2: Overview of TECHNIQUERAG.

While LLMs offer promising capabilities for ranking adversarial *(sub-)techniques*, this task poses challenges beyond their standard pre-training and alignment objectives. Unlike traditional ranking tasks—such as those encountered in question-answering—security technique ranking requires distinguishing among subtly different ATT&CK *(sub-)technique* that may co-occur in a text without explicit indicators (see Fig 1). Consequently, conventional re-ranking frameworks like RankGPT (Sun et al., 2023), though effective for general search, struggle with the nuanced demands of security-specific ranking. To address, our framework prompts LLMs to engage in explicit, step-by-step reasoning about *(sub-)technique* relevance, considering both high-level *techniques* and fine-grained *sub-techniques*. This structured decomposition not only enables more precise ranking but also captures hierarchical relationships among *(sub-)techniques* as in the ATT&CK framework.

In experiments, we evaluate our framework on three benchmarks, addressing both single-label and multi-label prediction tasks for *(sub-)techniques*. Results demonstrate that TECHNIQUERAG significantly outperforms various baseline approaches, including classification-based, retrieval/ranking-based, and hybrid methods. Furthermore, it achieves comparable or superior performance to vanilla RAG approaches, even when utilizing powerful LLMs like GPT-4o.

2 Related Work

Table 1 presents a comparison of the methods proposed for the *(sub-)technique* ID annotation task. These methods can be categorized into three groups based on their characteristics

Text-based Feature Extraction Initial ap-

Method	Problem Formulation	Key Features
TRAM (for Threat-Informed Defense, 2023)	Classification	Utilizes n-gram frequency features
TTPDrill (Husari et al., 2017a)	Matching/Ranking	Employs TF-IDF and BM25 for text retrieval
AttacKG (Li et al., 2022b)	Matching/Ranking	Leverages knowledge graph representations
TIM (You et al., 2022)	Classification	Incorporates textual and lexical features
LADDER (Alam et al., 2023)	Matching/Ranking	Uses sentence encoder embeddings
NCE (Nguyen et al., 2024)	Matching/Ranking	Applies task-adapted dual-encoder embeddings
Text2TTP (Kumarasinghe et al., 2024)	Matching/Ranking	Enhances dual-encoder retrievals with a cross-encoder embedding filter model
IntexEX (Xu et al., 2024)	Hybrid (Retrieval and Evaluation)	Combines sentence/entity-based search with LLM-based evaluation of candidate outputs
TechniqueRAG (Ours)	Retrieval Augmented Generation (RAG)	Integrates any retriever with LLM-based re-ranking and fine-tuned generation

Table 1: Overview of methods proposed for automatically mapping security text to Mitre ATT&CK *(sub-)technique.*. Our proposed TECHNIQUERAG leverages a flexible RAG framework by combining off-the-shelf retrievers, a novel LLM-based re-ranking mechanism, and a fine-tuned generator, distinguishing it from prior approaches.

proaches to *(sub-)technique* identification utilized classical text representations: bag-of-words models utilizing TF-IDF (Legoy et al., 2020; Tsai et al., 2020), n-gram frequencies (Legoy et al., 2020; for Threat-Informed Defense, 2023), and word embeddings (Legoy et al., 2020) as features for multi-class & multi-label classifiers. Later works enhanced representation through syntactic parsing to extract $(subject, verb, object)$ patterns (Husari et al., 2017b) & knowledge graph alignment (Li et al., 2022b) to capture contextual relationships in threat behaviors.

Neural Text Embedding Approaches Transformer-based language models (Reimers and Gurevych, 2019) enabled semantic similarity-based technique identification through neural embeddings. Early approaches used pre-trained encoders to embed threat behaviors, either handling multi-sentence descriptions (You et al., 2022) or specific attack patterns (Alam et al., 2023), evaluating relevance through embedding similarity. (Kumarasinghe et al., 2024) advanced this through a multi-stage architecture combining fine-tuned cross-encoder and dual-encoder models to balance effectiveness and efficiency. (Nguyen et al., 2024) further developed this approach using a dual-encoder architecture with alignment components, leveraging both scratch-trained embeddings and domain-specific models.

LLM Applications The application of LLMs to technique identification has yielded important insights. (Kumarasinghe et al., 2024) found that Normally, LLMs perform poorly compared to fine-tuned smaller models due to hallucination issues. To address, (Xu et al., 2024) introduced a hybrid approach combining zero-shot classification, retrieval, and LLM-based validation while we propose RAG to enhance reliability and reduce errors.

3 Method

We present TECHNIQUERAG, a domain-specific retrieval-augmented generation (RAG) framework for adversarial *technique* (or sub-*technique*) annotation. Unlike conventional approaches that rely on task-specific optimizations and extensive labeled data, TECHNIQUERAG effectively integrates retrievers, instruction-tuned LLMs and small-scale text-*techniques* paired data. We first provide an overview of our approach, followed by details on the key components: (i) retriever and (ii) LLM-based re-ranking (ii) generator fine-tuning. Fig 2 presents an overview of our system.

3.1 Problem Formulation and Overview

Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of security texts (e.g., attack behaviors) and $Y = \{y_1, y_2, \ldots, y_m\}$ denote the set of adversarial *(sub-)techniques* (e.g., MITRE ATT&CK IDs). For a given security text, its annotation is represented as a sequence $\mathbf{Y} = (y_1, y_2, \ldots, y_l)$, where each $y_i \in Y$ and $l \leq m$. We define Y^* as the set of all finite sequences over Y, so that $\mathbf{Y} \in Y^*$. We assume access to a small paired dataset $D = \{(x_1, \mathbf{Y}_1), (x_2, \mathbf{Y}_2), \ldots, (x_n, \mathbf{Y}_n)\}$ of threat descriptions and the corresponding set of ground-truth *(sub-)techniques*. Given an input text $x \in X$, the task is to predict the corresponding *(sub-)techniques* $\mathbf{Y}_x \subseteq Y^*$.

Our framework, TECHNIQUERAG, comprises three modules: (1) a *retriever* R, (2) an LLM-based *re-ranker* $\mathcal{R}$, and (3) a *generator* LLM $\mathcal{G}$. Given an input x^q, the retriever R first retrieves the top-K relevant pairs R_x from the dataset D based on their similarity to the query text. The re-ranker $\mathcal{R}$ processes the retrieved pair of annotated text, R_x, to produce an ordered sequence $\mathcal{R}_x$, which is then augmented with the input sequence x to form the generator context $\mathcal{C}_x = x \oplus \mathcal{R}_x$ where $\oplus$ denotes concatenation. To conform to the context length of the generator $\mathcal{G}$, the user may select $k \leq K$ re-ranked pairs for augmentation. These augmented pairs serve as exemplars that guide the generation process and help to reduce hallucination. Finally, the generator $\mathcal{G}$ produces the target output $\mathbf{Y}_x$ from the augmented input $\mathcal{C}_x$ (See Fig 2). In the following subsections, we provide detailed descriptions of the retriever R, the LLM-based re-ranker $\mathcal{R}$, and the generator $\mathcal{G}$ used in TECHNIQUERAG.

3.2 Retriever R

The retriever module processes a query security text x^q by leveraging a retrieval corpus D_R to fetch most relevant candidate pairs. In our setting, due to the lack of additional data, we employ the paired dataset D both as the retrieval corpus D_R and as the training data for the generator $\mathcal{G}$. To prevent data leakage during its training, we explicitly exclude x^q from D_R, defining it as:

$$D_R = \{(x_i, \mathbf{Y}_i) \mid (x_i, \mathbf{Y}_i) \in D \wedge x_i \neq x^q\}.$$

The retriever R returns the top-K pairs $R_x = \{(x_1^R, \mathbf{Y}_1^R), (x_2^R, \mathbf{Y}_2^R), \ldots, (x_K^R, \mathbf{Y}_K^R)\} \subset D_R$, where each pair $(x_i^R, \mathbf{Y}_i^R)$ corresponds to a security text and its associated *(sub-)techniques* from D_R along with their (lexical/semantic) similarity $sim(x^q, x_i^R) \geq sim(x^q, x_j^R)\ \forall\ j > i$. Any off-the-shelf retriever (e.g., sparse: BM25 or dense: pre-trained sentence embedding model) can be employed as R. While our approach is generic and further domain adaptation of R may improve performance, it is important to note that D only has behavior description and *(sub-)technique* annotation pairings $(x_i, \mathbf{Y}_i)$ without specifying the absolute relevance of x_i with any of the individual *(sub-)techniques* within the sequence of ground truth technique annotations $\mathbf{Y}_i$. As a result, training R solely with heuristic losses, such as in-batch negatives, leads to sub-optimal adaptation. Furthermore, no hard negatives or denoising data are available. Therefore, instead of fine-tuning a retriever, we employ an off-the-shelf retriever as R and enhance it through re-ranking, as detailed below.

3.3 Re-Ranker $\mathcal{R}$

To address data scarcity and enhance domain-specific precision, our re-ranker $\mathcal{R}$ refines the candidate set retrieved by R using an instruction-tuned large language model. Unlike generic prompting frameworks for ranking (e.g., RankGPT (Sun et al., 2023)), which lack domain-specific knowledge, our re-ranker employs a novel prompting framework specifically designed for adversarial technique annotation that addresses three key challenges described below.

Explicit Reasoning for Implicit Mapping: Security texts rarely provide explicit rationales for technique mappings. For example, the text "malware connects to C2 using custom encoding" implies both command-and-control (TA0011) and defense evasion (TA0005) tactics, but doesn't directly state this relationship. $\mathcal{R}$ instructs the LLM to decompose such implicit connections through structured reasoning:

Prompt and Response: Break Down the Query

# Decompose the given security query into distinct attack steps or phases.
# Identify any implied or explicitly mentioned behaviors that indicate specific adversarial *(sub-)techniques*.

Query: "malware connects to C2 .. custom encoding"

Step 1: Identify core behavior → C2 communication with encoding
Step 2: Map to tactics → Command and Control + Defense Evasion
Step 3: Link to techniques → T1071 (C2 Protocol) + T1027 (Obfuscation)

Balanced Consideration of Multiple Techniques: Security activities often involve multiple *techniques* simultaneously. $\mathcal{R}$ prompt ensures comprehensive coverage through parallel evaluation by instructing to explore each possible *technique*.

Prompt and Response: Multiple *Techniques*

# Consider that the query may involve multiple *(sub-)techniques*. (both direct and implied).

Query: "malware connects to C2 .. custom encoding"

Primary Technique: T1071 (Application Layer Protocol)
Rationale: Direct C2 communication behavior
Secondary Technique: T1027 (Obfuscated Files or Information)
Rationale: Custom encoding for evasion

Fine-Grained Relevance w.r.t Sub-Techniques: Certain *techniques* have multiple sub-*techniques* and some do not. Our framework evaluates subtle distinctions among them to inform the overall ranking of different *techniques* (with and without *sub-techniques*). For instance:

> **Prompt and Response: Fine-grained Relevance**
>
> # Map each identified step or behavior to the most appropriate technique or sub-technique (if available for each corresponding *technique*). For each matching technique, explain the connection between the query and the corresponding adversarial behavior.
>
> ---
>
> Query: "PowerShell script encoded in base64 downloads malware from remote server"
>
> ---
>
> Technique Analysis:
> 1. T1059 (Command and Scripting Interpreter)
> - T1059.001 (PowerShell): Direct match for script execution
> - Confidence: High due to explicit PowerShell usage
> 2. T1027 (Obfuscated Files)
> - No sub-techniques apply to basic encoding
> - Confidence: Med. as common obfuscation method
>
> ---
>
> Final Ranking: T1059.001 > T1027
> Rationale: Sub-technique analysis reveals credential access as primary intent with process injection as supporting mechanism

The complete system prompt to guide the LLM through this structured analysis is provided in Appendix E. This hierarchical, reasoning-based approach enables $\mathcal{R}$ to reorder candidates while maintaining alignment with ATT&CK's taxonomy, addressing ambiguities in initial retrieval. To address the challenge of processing large candidate sets within the LLM's context window, we utilize the sliding window mechanism as in Sun et al. (2023).

3.4 Generator $\mathcal{G}$

To adapt the LLM generator $\mathcal{G}$ that produces the final annotations of the adversarial technique $\mathbf{Y}_x$ from an augmented input $\mathcal{C}_x$, we fine-tune it using D as the training set. As discussed in Section 3.1, the augmentation process concatenates the original query x with the re-ranked candidate pairs $\mathcal{R}_x$ (i.e., $x \oplus \mathcal{R}_x$), specifically as following:

$$\begin{aligned}\mathcal{C}_x = x\ [text]\ x_1^{\mathcal{R}}\ [technique]\ \mathbf{Y}_1^{\mathcal{R}}\ [text]\\ x_2^{\mathcal{R}}\ [technique]\ \mathbf{Y}_2^{\mathcal{R}} \dots\ [text]\\ x_k^{\mathcal{R}}\ [technique]\ \mathbf{Y}_k^{\mathcal{R}},\end{aligned}$$

where "[]" denotes separator tokens, x_j and $\mathbf{Y}_j$ are parallel data (e.g., x_j is the security text and $\mathbf{Y}_j = (y_{1,j}, y_{2,j}, \dots y_{m,j})$ is the corresponding *(sub-)technique* sequence.

We train the generator model $\mathcal{G}$ minimizing the negative log-likelihood of the ground-truth *(sub-)technique* annotations $\mathbf{Y}_x = (y_{1,x}, y_{2,x}, \dots, y_{l,x})$ conditioned on the augmented input C_x:

$$\mathcal{L} = -\sum_{(x,\mathbf{Y}_x)\in D}\sum_{i=1}^{l}\log P_{\mathcal{G}}(y_{i,x} \mid \mathcal{C}_x).$$

To mitigate hallucination, $\mathcal{G}$ is constrained to generate outputs from the re-ranked candidate set C_x. This design ensures that the final predicted *(sub-)techniques* are both contextually grounded in x and consistent with the adversarial taxonomy provided by the exemplars in C_x.

4 Experiment Setup

4.1 Data and Implementation

We assess the capability of TECHNIQUERAG to accurately map threat behaviors to *(sub-)technique* IDs. We consider both single-label ($l = 1$) and multi-label ($l > 1$) prediction setups. Following previous works, we evaluate on three publicly available benchmark datasets: Tram (for Threat-Informed Defense, 2023) as a single-label dataset, and the Procedures and Expert datasets (Nguyen et al., 2024) representing single-label and multi-label settings, respectively. We report performances on the test sets of these datasets, training our model on the combined training sets. Rather than developing separate models for *technique* and *sub-technique* prediction, we train a single model for *sub-technique* prediction. This is motivated by the fact that *sub-technique* annotations provide a more fine-grained representation that inherently includes the broader *technique* identifier (e.g., in T1050.001, T1059 is the *technique* and 001 is the *sub-technique*). When evaluating for *technique* prediction, we simply truncate the *sub-technique* component. As the retriever R, we use BM25 with $K = 40$ and $k = 3$. For the frozen re-ranker $\mathcal{R}$, we employ DeepSeek v3 (Liu et al., 2024) (with temperature set to 0) , processing retrieved candidates in batches of 40 with an overlap of 20. The trainable generator $\mathcal{G}$ is implemented using an 8B Ministral Instruct model (MistralAI, 2024). Fine-tuning is performed with LLaMa-Factory using LoRA (Hu et al., 2021), with a learning rate of 10^{-4}, LoRA rank $r = 8$, and $\alpha = 4$. For generation, we use a sampling temperature of 0.7, a top-p value of 0.1, and a context length of 2048 tokens. Our source code, datasets, and models are publicly available on GitHub[1].

[1] https://github.com/qcri/TechniqueRAG

Model	Tram (Single-label)			Procedures (Single-label)			Expert (Multi-label)		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Retrieval-based Methods									
NCE	**90.30**	**78.90**	**84.22**	84.10	80.60	82.31		N/A	
Text2TTP	51.59	21.36	30.22	74.76	74.65	74.70			
BM25	67.86	64.74	66.26	32.54	32.48	32.51			
RankGPT	61.93	58.56	60.20	59.40	59.33	59.37			
Our Re-Ranker	64.69	61.43	63.02	85.46	85.29	85.37			
Generative Models									
GPT-4o	38.28	49.98	43.35	51.42	64.04	57.04	20.91	32.96	25.59
w/ CoT+Ref	43.52	67.20	52.83	51.84	78.47	62.43	38.19	48.67	42.80
DeepSeek v3	43.74	65.69	52.51	50.87	78.13	61.62	40.17	46.89	43.27
w/ CoT+Ref	43.68	66.36	52.68	51.55	75.86	61.39	36.39	49.25	41.85
Ministral 8B	7.68	31.71	12.36	7.07	30.79	11.50	8.47	19.63	11.84
w/ CoT+Ref	14.94	26.21	19.03	16.58	29.29	21.17	16.88	17.17	17.02
IntelEx	60.67	70.71	65.31	61.13	75.07	67.39	48.03	41.88	44.74
RAG Models									
GPT-4 (RAG)	55.50	70.64	62.16	71.34	88.06	78.82	47.49	55.76	**51.30**
DeepSeek v3 (RAG)	54.59	77.36	64.01	66.43	91.57	77.00	40.94	**60.86**	48.95
Ministral 8B (RAG)	51.88	57.61	54.60	61.40	69.81	65.34	43.21	35.23	38.81
TechniqueRAG	76.00	72.14	74.02	**91.11**	**91.06**	**91.09**	**75.16**	37.67	50.19

Table 2: Results in *technique* prediction. CoT+Ref: Chain-of-Thought w/ Reflection. The num of predicted labels are fixed for ranking models while generative models determines at runtime, hence compared in Section 5.3.

4.2 Baselines and Evaluation Metrics

Retrieval/Ranking-only Methods These include state-of-the-art approaches that rely solely on retrieval and re-ranking w/o using generative models. We compare w/ NCE (Nguyen et al., 2024) for contrastive domain-specific learning, Text2TTP (Kumarasinghe et al., 2024), which combines bi-encoder semantic search w/ cross-encoder re-ranking, underlying BM25 retriever baseline, and RankGPT (Sun et al., 2023) re-ranking framework that uses same BM25 retrievals. As NCE is not released we report from (Nguyen et al., 2024).

Generation-based Methods *Direct Generation:* We evaluate against powerful LLMs including GPT-4, DeepSeek V3, and Ministral 8B. For each model, we implement both direct prompting and chain-of-thought approaches with self-reflection (Shinn et al., 2024). *Retrieval-Augmented Generation:* We compare against IntelEX (Xu et al., 2024), a hybrid retrieval and LLM-judge approach. Additionally, we implement retrieval-augmented versions of the above LLMs using text and identical exemplars from our retrieved and re-ranked pairs ($\mathcal{C}_x$).

Evaluation Metrics Following previous works, we evaluate performance on two settings: (i) for single-label *technique* and *sub-technique* prediction task, we use standard **Precision**, **Recall**, and **F1** scores; (ii) for multi-label tasks, we adopt a differentiated evaluation protocol. Our evaluation consists of: (1) End-to-End Evaluation: comparing our model's adaptive label predictions with generative baselines, as both can dynamically determine the optimal number of labels—a capability that retriever-only methods lack; and (2) Ranking Analysis: evaluating our re-ranker against all retriever or ranking methods using standard ranking metrics (Precision, Recall, and F1) at k={1,3}.

5 Results and Analysis

5.1 *Technique*-Level Performance

Table 2 reports the performance of various models on the technique prediction task across three datasets with increasing diversity: Tram (single-label with 198 unique), Procedures (single-label with 488 unique), and Expert (multi-label with 290 unique) *techniques* and *sub-techniques*. Among retrieval-based methods, NCE achieves the highest F1 score on Tram (84.22%), reflecting its strength in a constrained label space. However, as the diversity increases, NCE's performance drops markedly—for example, on Procedures it only reaches an F1 of 82.31%

In contrast, our proposed TechniqueRAG excels consistently. On Procedures, TechniqueRAG attains an F1 score of 91.09%, demonstrating its superior ability to generalize in a high-diversity, single-label setting. Although on the Expert dataset proprietary model GPT-4o (RAG) achieves a marginally higher F1 (51.30% vs. TechniqueRAG's 50.19%), the overall perfor-

Model	Tram (Single-label)			Procedures (Single-label)			Expert (Multi-label)		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Retrieval-based Models									
NCE	**77.00**	65.80	**70.96**	75.70	71.88	73.74		N/A	
Text2TTP	42.62	40.41	41.49	71.08	70.94	71.01			
BM25	48.41	46.56	47.47	24.17	24.17	24.17			
RankGPT	43.03	40.97	41.97	51.19	51.12	51.16			
Our Re-Ranker	50.76	48.45	49.58	84.21	83.98	84.10			
Generative Models									
GPT4o	27.62	36.34	31.38	42.42	55.04	47.91	16.69	24.26	19.77
w/ CoT+Ref	32.33	49.91	39.24	43.90	68.33	53.46	32.24	37.75	34.78
DeepSeek v3	30.97	47.07	37.36	40.98	64.23	50.04	34.60	35.87	35.22
w/ CoT+Ref	33.28	51.47	40.42	41.47	61.63	49.58	30.79	37.14	33.67
Ministral 8B	3.72	21.99	6.37	3.25	17.32	5.47	6.65	14.72	9.17
w/ CoT+Ref	10.96	21.34	14.48	7.17	13.53	9.37	12.74	11.60	12.14
IntelEx	53.09	63.33	57.76	53.07	67.77	59.53	43.55	33.52	37.88
RAG Models									
GPT4o (RAG)	39.29	52.84	45.07	64.11	81.63	71.82	41.77	45.87	**43.73**
DeepSeek v3 (RAG)	39.31	58.54	47.04	59.72	86.47	70.65	35.91	**48.06**	41.11
Ministral 8B (RAG)	34.94	40.86	37.67	53.41	63.75	58.12	32.90	28.24	30.39
TECHNIQUERAG	72.69	**68.74**	70.66	**91.11**	**88.09**	**88.11**	**70.06**	30.21	42.22

Table 3: Performance Comparison for *sub-technique* prediction task (in percentage). Note: CoT+Ref: Chain-of-Thought with Reflection. Retrieval-based methods are not applicable for the multi-label Expert dataset.

mance averaged across the three datasets signifies the effectiveness of our open-source framework. When we compute the average F1 score, TECHNIQUERAG achieves approximately 80.76%, compared to only about 58.11% for GPT-4o (RAG). This substantial improvement underscores that our model is more robust, particularly in handling diverse and complex adversarial scenarios.

5.2 *Sub-Technique*-Level Performance

Table 3 presents the results at the *sub-technique* level. A similar trend is observed. At finer granularity, our method maintains dominance on Procedures (our F1 88.11 vs NCE's 73.74) while matching Tram's performance gap (F1 70.66 vs NCE's 70.96). This again posits our effectiveness for complex and robust threat annotation. The performance gap between our model and other generative and RAG baselines widens further at the *sub-technique* level. While GPT-4o (RAG) achieves a slightly higher F1 score on the Expert dataset (43.73 vs our 42.22), the overall results across all datasets demonstrate that our approach generalizes more effectively to complex, high-diversity environments.

5.3 Single-Label versus Multi-Label Settings

Our experiments in Table 2 and 3 reveal that multi-label prediction poses significant challenges in compare to single-label. For example GPT-4o achieves a F1 score of 76.75 in Procedure while only 19.77 in Expert). While retrieval augmented generation enhances all generative models, gains in open-source LLMs remain low such as using Ministral RAG without our finetuning scores a F1 of 30.39 in Expert. Adapting to the domain TECHNIQUERAG boosts it up to 42.22 tailing the RAG score of GPT-4o's 43.73. Furthermore, in Table 4 we compare all the ranking based models with our re-ranker framework–showing a large margin gains over all. These significant F1 improvements both in single and multi-label setup confirm the effectiveness of our model in real-world scenarios.

5.4 Ablation Study

Enhancement with Our Re-Ranker Our comprehensive evaluation in Table 2, 3 and 4 clearly indicate that our re-ranker not only outperforms all the ranking based methods but also enhance the overall end-to-end performances. In addition to our model, all the generative models (e.g., DeepSeek) in RAG setups using our re-ranked exemplars achieves notable gains over their direct or CoT+Ref inferences.

Gains over Other Fine-Tuning Methods We also validate the effectiveness of our RAG-based domain adaptation methods over zero-shot and CoT+Ref based methods. For zero-shot, we finetune our same Ministral model on the same training data but without exemplars and for CoT+Ref based methods we followed the Alpaca approach (Taori et al., 2023) where for our same train data using DeepSeek v3 as the teacher model with the

Model	Technique Level						Sub-Technique Level					
	@1			@3			@1			@3		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
NCE	**74.5**	23.6	35.9	—	—	48.3	**73.1**	18.2	29.1	—	—	39.9
Text2TTP	53.5	26.1	35.1	37.4	49.1	42.4	49.0	21.3	30.2	34.4	39.7	36.8
BM25	51.6	21.4	30.2	35.5	40.4	37.8	45.9	15.6	23.3	31.0	29.9	30.5
RankGPT	56.7	25.3	34.9	37.4	46.6	41.5	49.7	19.8	28.4	34.8	37.8	36.3
Our Re-Ranker	71.3	**35.3**	**47.2**	**44.6**	**59.9**	**51.1**	66.9	**29.0**	**40.5**	**47.1**	**54.2**	**50.4**

Table 4: Performance of Ranking Methods on Expert Dataset (Multi-Label). '-' refers to results not reported.

CoT+Ref prompt we synthesis a new traindata and then fine-tune the Ministral model. Results in Fig 3 shows our approach achieves the highest gain in both target tasks in all benchmark datasets.

5.5 Qualitative Analysis

Running Example. We provide in Appendix A a concrete example from Expert dataset that shows the predictions of our Re-Ranker and how TECHNIQUERAG generator improved it. We also provide examples of the prompts in RankGPT and ours with detailed responses with our re-ranker LLM DeepSeek V3 in Appendix E.

Error Analysis Analysis reveals few challenges:

Under-prediction. The model often captures primary techniques while missing related techniques in the same attack pattern (e.g., identifying T1055 but missing associated techniques like T1106)

Contextual errors. (i) Confusion between similar techniques within the same tactic family specially for *Command and Scripting Interpreter* techniques (T1059.*) (ii) Missing implicit or contextual techniques not explicitly stated (iii) Difficulty capturing logical relationships between techniques

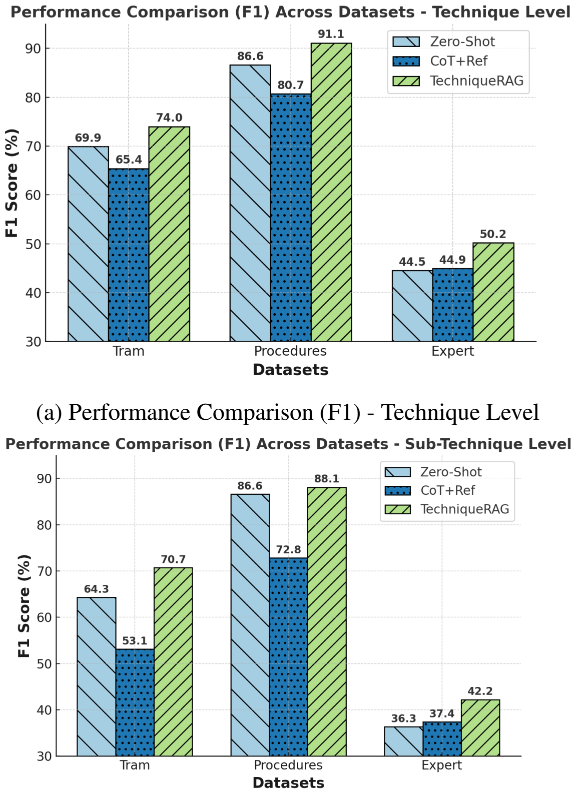
Hierarchical issues. Struggles with parent-child technique relationships and sometimes generates invalid sub-technique IDs

Re-ranker limitations. Missed *techniques* due to ambiguous queries and compound statements, affecting the Generator through propagation

Technique similarity. Challenges in distinguishing between *techniques* with overlapping descriptions and keywords (e.g., phishing-related techniques T1598.003, T1566.002, T1204.001)

Class Imbalance Effects. The severe data imbalance fundamentally impacts model performance - only 47 out of 203 *techniques* (23.2%) have more than 50 training samples. Techniques with abundant data show high precision and recall, while rare *techniques* suffer from both misclassification and under-prediction.

We present detailed analysis in Appendix C .



(a) Performance Comparison (F1) - Technique Level

(b) Performance Comparison (F1) - Sub-Technique Level

Figure 3: F1 scores for different fine-tuning methods.

6 Conclusion

Annotating threat intelligence texts with adversarial *techniques* from the MITRE ATT&CK framework is a manual and time-intensive task that security analysts must perform daily. Its automation requires methods capable of accurately identifying *techniques* and *sub-techniques* across hundreds of possibilities while handling complex security terminology, diverse text formats, and limited labeled data. We introduce TECHNIQUERAG a retrieval-augmented fine-tuning approach designed to tackle these challenges effectively. Our comparative analysis demonstrates that TECHNIQUERAG establishes a new state-of-the-art, outperforming both semantic ranking models and other LLM-based methods in adversarial technique annotation.

7 Limitations

Obtaining large, balanced parallel datasets of threat descriptions and ground truth technique annotations remains a significant challenge due to the reliance on domain expertise for accurate annotation. Although our approach mitigates data scarcity, two key limitations may impact performance:

1. **Limited Technique Coverage:** Coverage of techniques is often insufficient. Even the MITRE ATT&CK knowledge base lacks procedural examples for many techniques and sub-techniques.

2. **Sparse Technique Annotations:** Existing datasets typically contain very few technique annotations per example, with many instances in our data having only a single technique label. During fine-tuning, this bias toward minimal technique labeling limited our method's ability to generalize effectively. To mitigate this, we oversampled examples with multi-label technique annotations. However, our method rarely assigned more than two technique labels per input query, leading to low recall, particularly on the Expert dataset, which consists almost exclusively of multi-label examples.

3. **Annotation Inconsistencies** Some model predictions marked as errors represent valid technical interpretations not included in gold standard annotations. For example, the following sentence: *"SMOKEDHAM was observed using UltraVNC to establish a connection to the IP address and port pair 81.91.177[.]54[:]7234 that has been observed in past UNC2465 intrusions."* had *T1571: Non-Standard Port* as the only ground truth label. However, if we analyze it carefully, we see that the threat actor used UltraVNC, so *T1021.005: Remote Services - VNC* exists in the given description. Our model correctly predicted it, but missed the *T1571: Non-Standard Port*. This highlights challenges in maintaining consistent annotation standards for complex attack patterns.

Acknowledgments

The authors declare the use of AI Assistants in editing the text, fixing the typos, and generating the prompts and figures.

References

Md Tanvirul Alam, Dipkamal Bhusal, Youngja Park, and Nidhi Rastogi. 2023. Looking beyond iocs: Automatically extracting attack patterns from external cti. *Preprint*, arXiv:2211.01753.

MITRE Corporation. 2022. MITRE ATT&CK Enterprise Matrix. `https://attack.mitre.org/versions/v12/matrices/enterprise/`. Accessed: Jan 2, 2024.

Center for Threat-Informed Defense. 2023. Tram: Threat-risk assessment model. Accessed: 2025-01-02.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017a. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd annual computer security applications conference*, pages 103–115.

Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017b. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC '17, page 103–115, New York, NY, USA. Association for Computing Machinery.

Udesh Kumarasinghe, Ahmed Lekssays, Husrev Taha Sencar, Sabri Boughorbel, Charitha Elvitigala, and Preslav Nakov. 2024. Semantic ranking for automated adversarial technique annotation in security text. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 49–62.

Valentine Legoy, Marco Caselli, Christin Seifert, and Andreas Peter. 2020. Automated retrieval of att&ck tactics and techniques for cyber threat reports. *arXiv preprint arXiv:2004.14322*.

Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. 2022a. Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In *Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I*, pages 589–609. Springer.

Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. 2022b. Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I*, page 589–609, Berlin, Heidelberg. Springer-Verlag.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

MistralAI. 2024. Ministral-8b-instruct-2410. Accessed: 2025-02-15.

Tu Nguyen, Nedim Šrndić, and Alexander Neth. 2024. Noise contrastive estimation-based matching framework for low-resource security attack pattern recognition. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 355–373, St. Julian's, Malta. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Chia-En Tsai, Cheng-Lin Yang, and Chong-Kuan Chen. 2020. Cti ant: Hunting for chinese threat intelligence. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1847–1852. IEEE.

Ming Xu, Hongtai Wang, Jiahao Liu, Yun Lin, Chenyang Xu Yingshi Liu, Hoon Wei Lim, and Jin Song Dong. 2024. Intelex: A llm-driven attack-level threat intelligence extraction framework. *arXiv preprint arXiv:2412.10872*.

Yizhe You, Jun Jiang, Zhengwei Jiang, Peian Yang, Baoxu Liu, Huamin Feng, Xuren Wang, and Ning Li. 2022. Tim: threat context-enhanced ttp intelligence mining on unstructured threat data. *Cybersecurity*, 5(1):3.

Appendix

A Running Example

In Figure 4, the given text describes various execution techniques used by attackers, such as launching executables and DLLs in memory, leveraging schtasks.exe to modify task schedules, and executing PowerShell commands. The ground truth labels identify the relevant MITRE ATT&CK techniques: T1059.001 (PowerShell) is explicitly mentioned in "*The ability to launch custom PowerShell commands,*" T1053 (Scheduled Task/Job) and its sub-technique T1053.005 (Scheduled Task) are referenced in "*The ability to leverage schtasks.exe to add or modify task schedules.*" The RAG output provided partial alignment with the ground truth labels but included T1218.011 (Rundll32), which is not explicitly mentioned. However, this retrieval step helped the LLM Output by reinforcing correct predictions ensuring T1059.001 and T1053.005 were present while also introducing T1053. Additionally, the LLM expanded the predictions to include T1071.001 (Web Protocols), which does not appear relevant in this case.

Example of TECHNIQUERAG Output

The ability to launch a .exe or .dll file in memory. The ability to leverage "schtasks.exe" to add or modify task schedules. The ability to launch custom PowerShell commands. The ability to leverage a standalone executable to load the DLL if the attacker otherwise has no way of doing so.

Ground Truth Labels:
1. T1059.001: Command and Scripting Interpreter: PowerShell
2. T1053: Scheduled Task/Job
3. T1053.005: Scheduled Task/Job: Scheduled Task

RAG Output (Our Re-ranker):
1. T1218.011: System Binary Proxy Execution: Rundll32
2. T1059.001: Command and Scripting Interpreter: PowerShell
3. T1053.005: Scheduled Task/Job: Scheduled Task

LLM Output (TECHNIQUERAG):
1. T1059.001: Command and Scripting Interpreter: PowerShell
2. T1053: Scheduled Task/Job
3. T1053.005: Scheduled Task/Job: Scheduled Task
4. T1071.001: Application Layer Protocol: Web Protocols

Figure 4: Example of MITRE ATT&CK *techniques* and *sub-techniques* highlighted in text with corresponding colored (implicit) indicators. IDs with "." denote sub-*techniques* (e.g., T1059.001). Greyed-out IDs indicate incorrect predictions.

B Data Statistics

Tables 5 and 6 shows the details of the employed datasets. The Expert split consists of actual sentences from full reports published by threat intelligence vendors. These sentences are multi-label, meaning they can be associated with multiple MITRE ATT&CK techniques. In contrast, the Tram split contains incomplete sentences, such as "*opens cmd.exe on the victim*", "*searches for specified files*", or "*icacls . /grant Everyone:F /T /C /Q*", often presenting isolated technique references without sufficient context. Tram is single-label, meaning each sentence corresponds to only one technique. The Procedures split, extracted from the MITRE knowledge base, consists of complete sentences that summarize a single technique mentioned in a report. These sentences provide structured descriptions of attack techniques but are also single-label. In total, the training splits contain 499 unique techniques, covering approximately 78% of the 637 techniques available in the MITRE ATT&CK Enterprise Framework.

Table 5: Dataset Statistics

Dataset	Split	Avg Word Count	Data
Expert	Train	38.00	472
	Test	71.42	158
Procedures	Train	13.36	10,999
	Test	13.43	1768
Tram	Train	2.94	3469
	Test	21.22	726

Table 6: Dataset statistics. *S+T* denotes the joint count of techniques and sub-techniques.

Dataset	Texts	S+T	Techniques	Avg. # Labels	Avg. # Tokens
TRAM	4797	193	132	1.16	23
Procedures	11723	488	180	1.00	12
Expert	695	290	151	**1.84**	**72**

C Error Analysis

Common Errors. Analysis of the prediction errors reveals several systematic patterns in MITRE ATT&CK technique classification. The most frequent error type involves under-prediction, where the model identifies only the most prominent technique while missing other techniques that are part of the same attack pattern. For example, when analyzing process injection scenarios, the model often identifies the primary technique (*T1055: Process Injection*) but fails to capture associated techniques like *T1106: Native API* or specific sub-techniques like *T1055.001: Dynamic-link Library Injection*. Another common pattern involves confusing similar techniques within the same tactic family, particularly between various *Command and Scripting Interpreter* techniques (T1059.*). The model also demonstrates a tendency to miss contextual techniques that are implied but not explicitly stated in the text, such as failing to identify *T1082: System Information Discovery* when enumeration of system resources is described as part of a larger operation. Additionally, there is a notable pattern of missing data staging and encoding techniques (T1074, T1132) when they are described as part of exfiltration workflows.

C.1 Contextual Inference Failures.

The model demonstrates limitations in capturing implicit relationships, often missing techniques that are logical precursors or consequences of explicitly described actions. It frequently identifies primary techniques while missing related concurrent techniques within the same attack pattern.

Class Imbalance Effects. The severe data imbalance fundamentally impacts model performance - only 47 out of 203 techniques (23.2%) have more than 50 training samples. Techniques with abundant data show high precision and recall, while rare techniques suffer from both misclassification and under-prediction.

C.2 Dependency on Re-ranker

In several cases, some *(sub-)*techniques are omitted, likely due to ambiguous language in the query or an overemphasis on the most actionable part of a compound query for example. This, error further propagate to our Generator, which uses output from Re-ranker as few-shot examples.

C.3 Similar Techniques

Several techniques within the MITRE ATT&CK framework share significant similarities and often use overlapping keywords, which can influence our initial BM25 rankings. For instance, T1598.003, T1566.002, and T1204.001 are all phishing-related techniques that have similar descriptions with minor distinctions.

D Different Domain Adaptation Methods

Table 7: Performance Comparison of TECHNIQUERAG Across Domain Adaptation techniques Levels (Percentage Scores)

Model	**Tram**						**Procedures**						**Expert**					
	Technique			Sub-Technique			Technique			Sub-Technique			Technique			Sub-Technique		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Zero-shot	65.7	**74.9**	69.9	60.0	**69.2**	64.3	85.8	87.4	86.6	85.8	87.4	86.6	41.3	48.3	44.5	35.1	**37.6**	36.3
+ CoT+Ref	64.1	66.7	65.4	52.0	54.3	53.1	77.4	84.3	80.7	69.1	76.9	72.8	48.6	41.7	44.9	42.4	33.5	37.4
TECHNIQUERAG	**76.0**	72.1	**74.0**	**72.7**	68.7	**70.7**	**91.1**	**91.1**	**91.1**	**91.1**	**88.09**	**88.11**	**75.2**	37.7	**50.2**	**70.1**	30.2	**42.2**

E Prompts

Vanilla RankGPT Prompt and Output

# System Prompt:
You are RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query.

## Objectives:
I will provide you with {num} passages, each indicated by a number identifier [ ].
Rank the passages based on their relevance to the query.

## Given Passages:
{Passage 1: Description}
{Passage 2: Description}
⋮
{Passage n: Description}

## Query:
Monero miner scripts are downloaded from TeamTNT's server and piped to `bash` using an SSH session on the underlying host as the `root` user by supplying the private key from `/tmp/TeamTNT`. Later, the private key `/tmp/TeamTNT` is removed as well.

## Output

T1552.004 > T1098.004 > T1563.001 > T1021 > T1555.002 > T1573.002 > T1546.004 > T1496 > T1059.004 > T1611

Figure 5: Example RankGPT prompt and its corresponding output. Green colored IDs are the correct ones.

Re-ranker Prompt

# System Prompt

Act as an expert security analyst specializing in ranking a given list of MITRE ATT&CK techniques by their relevance to a security query.

## Objectives:

1. Determine if the given query describes an adversarial or attack behavior.
2. If it does, identify and rank the most relevant techniques and sub-techniques from the provided list.

## Instructions for Ranking:

1. Break Down the Query:

- Decompose the security query into distinct attack steps.
- Identify explicit and implied adversarial behaviors.

2. Match Techniques:

- Map each behavior to relevant techniques/sub-techniques.
- Consider multiple relevant techniques.

3. Provide Explanation:

- Explain the connection between the query and the matched techniques.

## Final Output Format:

After reasoning, output the final ranking as: [Technique A] > [Technique B] > [Technique C] > ...
No variations or extra formatting allowed.

## Given Techniques:

{Technique 1: Description}
{Technique 2: Description}
⋮
{Technique n: Description}

## Query:

Monero miner scripts are downloaded from TeamTNT's server and piped to `bash` using an SSH session on the underlying host as the `root` user by supplying the private key from `/tmp/TeamTNT`. Later, the private key `/tmp/TeamTNT` is removed.

## LLM Output

The query describes an attack involving the following key steps and techniques:
Initial Access: The attacker gains access to the host via SSH using a private key stored in /tmp/TeamTNT. This aligns with T1552.004 (Unsecured Credentials: Private Keys) and T1021.004 (Remote Services: SSH).
Execution: The attacker downloads Monero miner scripts and pipes them to bash. This involves T1059.004 (Command and Scripting Interpreter: Unix Shell).
Persistence: The attacker uses SSH with a private key, which could imply persistence via T1098.004 (Account Manipulation: SSH Authorized Keys).
Resource Hijacking: The Monero miner script indicates T1588.001 (Obtain Capabilities: Malware) for downloading and executing the miner.
Key Techniques Identified: T1552.004: Exploitation of private keys for SSH access.
T1021.004: Use of SSH for remote access.
T1059.004: Execution of scripts via Unix shell (bash).
T1098.004: Potential persistence via SSH authorized keys.
T1588.001: Downloading and executing Monero miner scripts (malware).
Ranking Explanation:
Passages [1], [2], [6], [7], [8], [40] are highly relevant as they directly address the techniques used in the query. Passages [3], [5], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39] are less relevant as they describe techniques not directly involved in the query.
Final Ranking:
T1552.004 > T1098.004 > T1021.004 > T1059.004 > T1588.001 > T1496 > T1563.001 > T1546.004 > T1611 > T1140

Figure 6: Example re-ranker prompt and its corresponding output. Green colored IDs are the correct ones.

Self-Relflection Prompt

Your task is to analyze a given text describing malware behavior, extract the associated MITRE ATT&CK techniques to this text, explain their relevance to it.

## Context:
The MITRE ATT&CK framework is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. It's used by cybersecurity professionals to better understand and defend against cyber threats.

## Your Task:
Analyze the following text. Extract all the associated MITRE ATT&CK technique and provide a detailed explanation of why each technique is relevant to the text.

## Instructions:
Follow these steps to provide your analysis:
1. <thinking>: Explain your thought process as you analyze the given cyber threat description. Identify key actions, tools, or methods mentioned that could correspond to MITRE ATT&CK techniques in 30 words.
2. <reflection>: Reflect on your initial analysis. Consider if you've missed any potential techniques or if any of your initial thoughts need revision. Think about the confidence level of your associations in 30 words.
3. <output>: Based on your thinking and reflection, output the final list of MITRE ATT&CK techniques as technique IDs and their names. For example:
<output>
- T1221: Template Injection
- T1205.001: Traffic Signaling - Port Knocking
</output>

Ensure you use these exact tags (<thinking>, <reflection>, and <output>) in your response.

## Output Format:
<thinking>
Based on the given cyber threat description, I can identify several key actions and tools that correspond to MITRE ATT&CK techniques:
1. [Insert relevant observations from the text]
2. [Continue with more observations]

These observations suggest the following potential MITRE ATT&CK techniques:
- [List potential techniques with brief explanations]
</thinking>

<reflection>
Upon reflection, I should consider the following:
1. Are there any subtle indicators in the text that I might have overlooked: [Your answer in 20 words or less for question 1]
2. Have I considered the full context of the attack, including potential preliminary or subsequent steps not explicitly mentioned? [Your answer in 20 words or less for question 2]
3. Are there any techniques I've identified that might not be fully supported by the given information? [Your answer in 20 words or less for question 3] [Add any additional reflections or revisions to the initial analysis]
Confidence level: [State the confidence level in the identified techniques]
</reflection>

<output> [List the final list of the extracted MITRE ATT&CK techniques as technique IDs and their names.]
</output>

Figure 7: The employed prompt in self-reflection.