LLM-Assisted Advanced Persistent Threat Detection and Technique Explanation (LADE)

Anonymous ACL submission

Abstract

Advanced Persistent Threat (APT) attacks are sophisticated cyberattacks characterized by stealth, persistence, and long-term engagement with targeted systems. Traditional detection methods using machine learning (ML) and deep learning (DL) often rely on internal models or post hoc explainability techniques, which typically lack human-readable context and require manual interpretation. In this paper, we investigate the use of large language models (LLMs) for APT detection through code analysis. Specifically, we evaluate the ability of LLMs to (i) detect APT-related behavior in code snippet sequences, (ii) identify malicious components, and (iii) recognize relevant MITRE ATT&CK Tactics, Techniques, and Procedures (TTPs). Our results indicate that while LLMs show moderate effectiveness in detecting APTs and identifying malicious code, they perform well in recognizing ATT&CK techniques, when supplemented with domain-specific knowledge from the ATT&CK framework.

1 Introduction

Advanced Persistent Threat (APT) attacks represent a sophisticated class of cyber attacks characterized by stealth, persistence, and long-term engagement with targeted systems (Zhang et al., 2024b). These attacks typically unfold in multiple stages: Initial Compromise, where the attacker gains access; Persistence, established through backdoors or remote access tools; Lateral Movement, enabling the attacker to navigate within the network; and Exfiltration, where sensitive data is extracted (Wang et al., 2024). Unlike conventional attacks, APTs are designed for long-term exploitation, making timely and effective detection critical to minimizing their impact and preventing extended damage.

Conventional approaches for APT detection leveraging machine learning (ML) (e.g., (Han et al., 2020; Dong et al., 2023)) and deep learning (DL) models (e.g., (Du et al., 2017; Jia et al., 2024; Li et al., 2023a)), typically rely on large amounts of labeled or unlabeled training data. Their decision-making processes are explained either through internal mechanisms (Čík et al., 2021) or post hoc explainability techniques (Scott et al., 2017). These explanations often lack human-readable context, requiring manual interpretation to extract meaningful insights from highlighted features.

Large Language Models (LLMs) have shown potential in security tasks such as vulnerability assessment and patching (Liu et al., 2024; Nong et al., 2024), malware detection (Li et al., 2023b; Qian et al., 2025), and code de-obfuscation (Patsakis et al., 2024). However, the use of LLMs for APT detection is limited. Recent approaches such as SHIELD (Gandhi et al., 2025) and APT-LLM (Benabderrahmane et al., 2025) integrate LLMs with traditional methods for system log analysis to detect APT attacks. However, abstracting low-level execution details and transforming them into provenance graph representations in these log-based methods can lead to information loss. Although LLMs were utilized to generate natural language explanations in log-based detection, these explanations primarily reflect system-event interactions (e.g. a process writing to a file), which may not fully capture the attacker's intent.

To provide security analysts with deeper and more actionable insights into APT attacks and techniques, we investigate the use of LLMs for APT detection from a code analysis perspective. Code-level analysis preserves fine-grained execution details (e.g., parameters, flags, and execution contexts) that are often lost in high-level system logs. Traditional code-level analysis relies on static analysis, dynamic analysis, and machine learning applied to suspicious source code, binaries, or scripts (Salim et al., 2023; Chakkaravarthy et al., 2018; Fang et al., 2021). However, these approaches typically analyze individual samples without considering the temporal and contextual relationships present in multi-stage APT campaigns.

Other research on code-level analysis has focused on detecting malicious activities using rule-based approaches (Splunk, 2023, 2024) or by mapping these activities to MITRE ATT&CK techniques through similarity measures (Okuma et al., 2023). However, none of them explored the use of LLMs for detecting and explaining APT attacks.

Our work aims to address the following research questions: **RQ1:** Given a sequence of code snippets containing both benign and APT-related malicious content, can an LLM identify the presence of APT attack activities within the sequence? **RQ2:** If an APT attack is detected in a sequence, can the LLM accurately pinpoint the specific code snippets responsible for the attack? Accurate identification of malicious snippets can help reduce the workload for security analysts by narrowing their focus to critical code. **RQ3:** For each identified malicious snippet, can the LLM accurately map it to the corresponding MITRE ATT&CK technique (referred to as "TTP mapping")? **RQ4:** How does the incorporation of ATT&CK technique descriptions as external domain knowledge, along with various prompt engineering strategies, affect the performance of the LLM? To this end, we propose LLM-assisted APT Detection and Explanation (LADE), a novel framework for detecting APT attacks through code analysis. LADE employs a layered analysis approach that decomposes the code analysis task into three subtasks: detecting APT activities, localizing relevant code snippets, and mapping them to ATT&CK techniques. Each subtask is handled by a dedicated LLM, utilizing prompting strategies such as zero-shot, rubric-based, and chain-of-thought prompting. The responses are guided by APT-specific prompts, evaluation rubrics, and domain-specific knowledge that includes descriptions of ATT&CK techniques.

Due to the absence of standard APT benchmark datasets, we created our own datasets for evaluation. Our APT dataset contains PowerShell code snippets simulating APT attacks collected from Caldera (Corporation, a), an open-source framework developed by MITRE based on the ATT&CK framework (Corporation, b). Benign PowerShell code snippets were obtained from several public repositories (May, 2023; Fleschutz, 2023; Hub, 2024). The datasets used in this paper is available at [1]. With this dataset, we have assessed the impact of domain knowledge and prompt engineering strategies on LLM performance and evaluated LLM-generated explanations for TTP mappings using both standard metrics (ROUGE, BERTScore) and the "LLM-as-a-Judge" framework. Our evaluation results show that while LLMs demonstrate moderate performance in APT detection and malicious code identification, they perform well in TTP mapping, when supplemented with domain-specific knowledge.

[1]https://anonymous.4open.science/r/LADE-dataset

Our contributions are summarized as follows:

- To the best of our knowledge, this is the first study to explore the use of LLMs for both APT detection and TTP mapping from a code analysis perspective.
- We construct datasets for APT detection using open-source resources, including MITRE Caldera and public PowerShell repositories.
- Our evaluation results show that LLMs show moderate effectiveness in APT detection and malicious code identification, but achieve strong performance in TTP mapping when supplemented with domain knowledge.

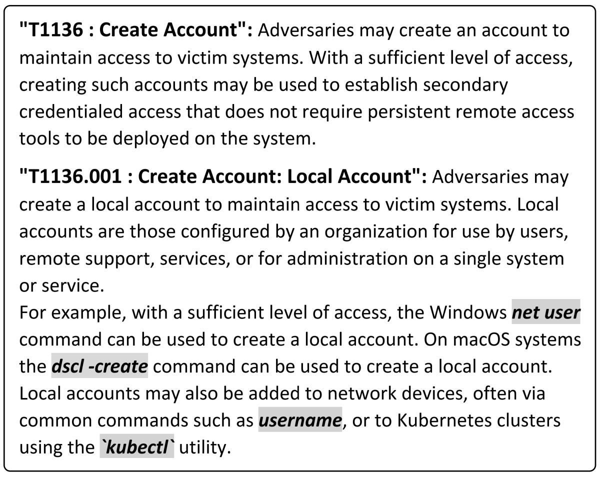
2 Background

This section provides an overview of MITRE ATT&CK framework and Rubric-based prompting.

MITRE ATT&CK: MITRE ATT&CK (Corporation, b) is an open-source knowledge base of adversarial behavior based on real-world threats and threat actors. It organizes behavior into a hierarchy of Tactics, Techniques, and Procedures (TTPs): Tactics represent adversary goals, Techniques describe how those goals are achieved, and Procedures are specific implementations. Figure 1 shows examples of MITRE ATT&CK technique description. Sub-technique "T1136.001 (Local Account)" (Corporation, d) is a variant of technique "T1136 (Create Account)" (Corporation, c), describing how an adversary creates a local account to gain persistence access. Throughout this paper, we use the term "technique" to refer to both techniques and sub-techniques unless otherwise noted.

APT code executed on the host machine can be chronologically logged using system monitoring tools (e.g., Sysmon or CrowdStrike Falcon (Microsoft, 2025; Elastic, 2025)) and host-based logging (e.g., PowerShell script block logging (Microsoft, 2024)).

Rubric Based Prompt Engineering: Providing rubrics to LLMs is a prompt engineering strategy predominantly used in the education field for tasks such as automated grading (Yancey et al., 2023; Tian et al., 2024). A rubric is generally defined as a set of criteria for assessment, accompanied by scoring guidelines, with scores either numeric or


"T1136 : Create Account": Adversaries may create an account to maintain access to victim systems. With a sufficient level of access, creating such accounts may be used to establish secondary credentialed access that does not require persistent remote access tools to be deployed on the system.

"T1136.001 : Create Account: Local Account": Adversaries may create a local account to maintain access to victim systems. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service.
For example, with a sufficient level of access, the Windows ***net user*** command can be used to create a local account. On macOS systems the ***dscl -create*** command can be used to create a local account. Local accounts may also be added to network devices, often via common commands such as ***username***, or to Kubernetes clusters using the ***`kubectl`*** utility.

Figure 1: ATT&CK Technique Description Examples

categorical. Providing rubrics to LLMs is suited for tasks where a correct answer (ground-truth) exists and can be assessed according to a hierarchy of correctness or relevance. Scoring guidelines can range from general measures, such as evaluating the semantic relevance between an input and its ground truth, to more domain-specific analyses, such as assessing how well an input aligns with criteria based on domain-specific knowledge. Rubric-based prompting is commonly used in "LLM-as-a-Judge" approaches (Li et al., 2024).

3 Overview of LADE

In APT attacks, once a target host is compromised, attackers issue malicious commands and scripts (APT code) from a Command and Control (C2) server. This code is often transmitted using evasion techniques, such as encryption or fragmentation, to avoid detection by network monitoring systems. Upon reaching the target, it is decrypted, reassembled, and executed through command-line interfaces like PowerShell (Sidhardhan et al., 2023). The executed code can be captured by host-based logging tools, such as PowerShell Script Block Logging (Microsoft, 2024). In this work, we assume that these logging mechanisms are neither tampered with nor disabled by attackers.

LADE leverages LLMs to perform APT detection, TTP mapping, and explanation by analyzing executed code snippets recorded on the host machine. Figure 2 illustrates the evaluation pipeline of LADE. LADE uses a layered analysis approach that decomposes the task into a sequence of manageable subtasks to enable structured interaction with the LLM. Given a sequence of code snippets, the LLM is first prompted to assess whether the input exhibits behavior indicating an APT attack. Upon detection of an attack, the model is tasked with localizing the specific code snippets responsible for the attack. Each identified snippet is subsequently mapped to the top 10 relevant MITRE ATT&CK techniques (referred to as "tagging"), along with a tagging summary that explains each assigned technique. We then evaluate the tagging summaries using both quantitative metrics (e.g., ROUGE) and the LLM-as-a-Judge approach.

4 Evaluation Methodology

This section describes our evaluation framework.

4.1 APT Attack Detection

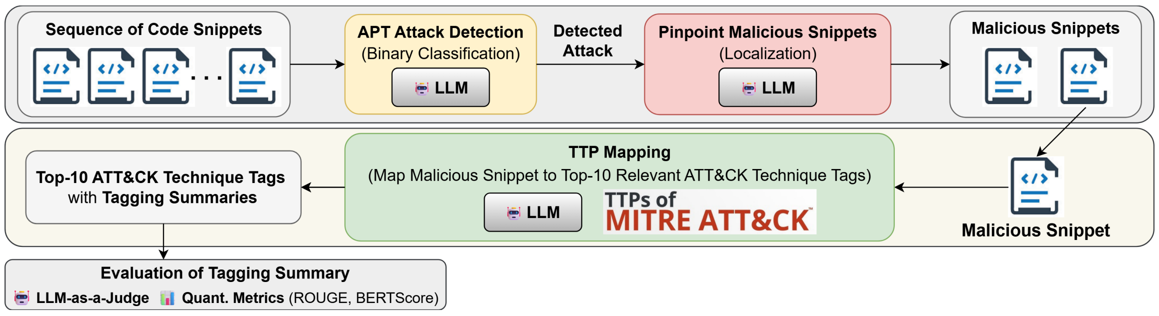
APT attacks typically follow a step-by-step progression toward a malicious goal, with both direct dependencies (e.g., data collection followed by exfiltration) and implicit relationships (e.g., separate reconnaissance actions). While some steps individually indicate malicious intent, others may appear benign in isolation but collectively reveal malicious intent (e.g., repeated discovery commands used to map the target environment). To address this, we prompt LLM to analyze sequences of code snippet for both independent and interdependent behaviors indicating adversary operations. We use a zero-shot prompt-engineering approach to provide instructions without examples. Zero-shot was chosen due to its simplicity and the difficulty of curating high-quality, labeled code sequences for few-shot prompting. In addition, LLM is prompted to generate natural language explanations to justify its classifications based on observed behaviors. The full prompt is given in Figure 3. An example explanation provided by GPT-4o is given in Figure 6 in the Appendix.

4.2 Pinpoint Malicious Snippets

If the LLM identifies a sequence of code snippets as containing APT activity, we prompt it to pinpoint the specific code snippets involved. We use a rubric-based prompting strategy to guide the LLM to evaluate each snippet's role in the malicious sequence. The rubric, shown in Figure 7 in the Appendix, defines four levels of malicious intent based on APT domain expertise. The highest level (4) indicates clear malicious intent or direct ties to coordinated malicious activity, while levels 3 to 1 represent decreasing suspicion, from likely indirect involvement (Level 3), to possibly context-dependent connections (Level 2), to benign (Level 1). Code snippets classified as levels 4, 3, and 2 are included in the malicious chain.

4.3 TTP Mapping

As a single snippet may exhibit behaviors associated with multiple MITRE ATT&CK techniques,



Sequence of Code Snippets
APT Attack Detection
(Binary Classification)
LLM
Detected Attack
Pinpoint Malicious Snippets
(Localization)
LLM
Malicious Snippets
TTP Mapping
(Map Malicious Snippet to Top-10 Relevant ATT&CK Technique Tags)
LLM
TTPs of MITRE ATT&CK
Malicious Snippet
Top-10 ATT&CK Technique Tags with Tagging Summaries
Evaluation of Tagging Summary
LLM-as-a-Judge
Quant. Metrics (ROUGE, BERTScore)

Figure 2: Architecture of LADE.

> Analyze the code sample, which consists of a sequence of code snippets. Your primary objective is to determine whether the code sample contains any adversary operation.
>
> - If the code sample contains snippets that individually or in tandem execute an adversarial objective, then classify it as containing an adversary operation.
> - If the code sample consists of benign code snippets without any signs of adversarial behavior, then classify it as not containing an adversary operation.
>
> Provide an explanation for your classification, referencing specific behaviors or patterns observed in the code sample.
>
> **Code Sample:** `{code-sample}`

Figure 3: Prompt for APT attack detection.

> **Prompt 1:** Given the code snippet and its referenced code (if available) below, your task is to generate a behavior description following the 5 steps provided in the subsequent prompts.
>
> **Code Snippet:** `{code-snippet}`
>
> **Referenced Code:** `{referenced-code}`
>
> **Prompt 2:** Identify the core commands of the code snippet and the referenced code (if provided).
>
> **Prompt 3:** Examine the inputs and outputs of the core commands.
>
> **Prompt 4:** Analyze the actions of the core commands.
>
> **Prompt 5:** Describe the behavior of the core commands in natural language.
>
> **Prompt 6:** Refine the description to highlight key actions, impacts, and high-level objectives, similar to a MITRE ATT&CK technique description.

Figure 4: Prompts for behavior description generation.

we prompt the LLM to map each identified snippet to the top 10 relevant techniques (refered to as “TTP mapping”). While LLMs demonstrate its possession of some internal knowledge of ATT&CK techniques, it is prone to hallucinations and may rely on outdated information (Welz and Lanquillon, 2024; Corporation, e). To mitigate this, we supply the LLM with the up-to-date ATT&CK technique descriptions as external domain knowledge.

We also leverage LLM’s code summarization capability to generate natural language behavior descriptions that align with ATT&CK technique descriptions. This process consists three steps: (1) code analysis, (2) behavior description generation, and (3) refinement to match official descriptions. We adopt a Chain-of-Thought (CoT) prompting strategy (Wei et al., 2022) to decompose the task into intermediate steps that mirror the analytical process of a security expert, as shown in Figure 4. Figure 8 in the Appendix shows a behavior description generated from a code snippet using LLM.

To tag ATT&CK techniques, we use a structured prompt-template shown in Figure 5, which applies to both behavior descriptions and code snippets. To ensure a systematic evaluation of semantic alignment, we apply a five-level rubric. The highest level (5) denotes excellent semantic alignment between the technique and behavior description, while lower scores indicate decreasing alignment. Techniques are ranked in descending order based on their alignment scores to identify the top 10 relevant techniques. If multiple techniques in the top 10 receive the same score, LLM evaluates their relative semantic alignment to finalize the ranking. In addition, for each of the top 10 techniques, a structured tagging summary is generated, which includes an explanation of the tag’s relevance. An example of a generated tagging summary is illustrated in Figure 9 in the Appendix.

4.4 Evaluation of Tagging Summary

We evaluate the tagging summary using the following three sequentially dependent evaluation criteria.

- **Criterion 1:** “Relevant Excerpt” accurately reflects the content in the “Cited Source.”
- **Criterion 2:** “Relevant Excerpt” is indeed relevant to the “Behavior Description.”
- **Criterion 3:** “Tagging Explanation” reflects both “Behavior Description” and “Relevant Excerpt” in justifying the tag.

Criterion 1 assesses the factual alignment between the excerpt and the corresponding MITRE ATT&CK domain knowledge from the cited source. Criteria 2 and 3 evaluate the semantic relevance of the excerpt to the behavior description and the coherence of the tagging explanation.

Tagging summaries are scored on a 1 (low) to 5 (high) scale according to the criteria described

TASK INSTRUCTION: Tag the given `{code snippet | behavioral description}` with the top 10 relevant ATT&CK techniques. Rank them in descending order of confidence level based on the following rubric. If multiple techniques share the same confidence level, assess their relative confidence to determine the order.

Rubric:

- **Level 1 (Minimal):** Minimal semantic alignment between `{code snippet and technique | behavioral description and technique description}`, with only general traits and weak supporting evidence.
- **Level 2 (Limited):** Limited semantic alignment between `{code snippet and technique | behavioral description and technique description}`, with some shared elements, but ambiguities weaken the match.
- **Level 3 (Adequate):** Adequate semantic alignment between `{code snippet and technique | behavioral description and technique description}`, with sufficient evidence, though some discrepancies remain.
- **Level 4 (Strong):** Strong semantic alignment between `{code snippet and technique | behavioral description and technique description}`, with significant evidence and only minor uncertainties.
- **Level 5 (Excellent):** Excellent semantic alignment between `{code snippet and technique | behavioral description and technique description}`, supported by comprehensive and compelling evidence.

Required Output Structure:

- **Tagging Explanation**: Explanation of the tag's relevance to `{code snippet | behavioral description}`.
- **Confidence Level**: Specify the confidence level for the tag.
- **Cited Source**: Provide a precise reference to the source supporting the match.
- **Relevant Excerpt**: Include an excerpt from the cited source that justifies the tag's relevance.

`{Code Snippet with Referenced Code (if available) | Behavioral Description}`: [Insert]

Figure 5: Prompt template for MITRE ATT&CK technique tagging.

above. Detailed rubrics for each criterion are provided in Figures 12, 13, and 14, respectively. Each criterion is evaluated with standard quantitative metrics such as ROUGE and BERTScore, and with an "LLM-as-a-Judge" approach, in which one LLM evaluates outputs generated by another and is often paired with rubrics for automated grading in educational settings (Yancey et al., 2023; Tian et al., 2024).

5 Experimental Results

This section presents our dataset and experimental results. Our experiments focus on GPT-4o (Achiam et al., 2023), which has demonstrated proficiency in cybersecurity tasks (Zhang et al., 2024a; Yu et al., 2024) and knowledge (Ferrag et al., 2024).

5.1 Dataset

Many existing approaches for detecting APT activities rely on proprietary enterprise datasets (Liu and Buford, 2023; Vinay and Mangal, 2024). Publicly available datasets (Greenberg, 1988; Lane and Brodley, 1997; Schonlau et al., 2001; Lin et al., 2018) primarily consist Unix shell commands intended for masquerade detection. These datasets are largely outdated and often contain truncated commands that omit command flag information and subshells due to privacy restrictions.

To overcome these limitations, we construct our own dataset for APT detection, which include benign and APT-labeled code snippet sequences. APT-labeled sequences are generated by converting Caldera's adversary profiles and emulation plans into APT snippets, which are then blended with benign snippets. Our dataset contains 33 APT-labeled sequences. Of these, 20 were derived from adversary profiles that model specific attack behaviors, with an average of 4 APT and 8 benign snippets. 13 were created from emulation plans. Among these 13 sequences, 8 represent larger, multi-stage APT attack scenarios with an average of 25 APT and 50 benign snippets, while the remaining 5 capture smaller-scale lateral movement behaviors with an average of 6 APT and 12 benign snippets. To evaluate LLM's detection performance across different attack stages, including early, middle, and late phases, we further divide the 8 larger sequences into 20% intervals, resulting in 40 segments for more fine-grained analysis. Preprocessing steps involved removing comments and print commands to avoid inadvertently revealing malicious intent. Code snippets intended for execution on the attacker's machine were excluded, and each sequence was curated to include only snippets meant for execution on the same machine.

Benign snippets in APT-labeled sequences were randomly sampled from a pool of 744 code snippets aggregated from the three publicly available repositories containing commonly used commands (May, 2023), general-purpose scripts (Fleschutz, 2023), and explicitly labeled benign commands (Hub, 2024). APT snippets were inserted at random positions among benign snippets, while preserving their sequential dependency. This setup mimics a realistic scenario on a victim machine, where malicious activities are interleaved with normal user behaviors. We maintain a 2:1 ratio of benign to APT snippets, reflecting moderate user activity. Considering that logging tools such as PowerShell Script Block Logging can capture resolved (i.e., expanded) code including external scripts, we represent each entry as a pair of "code snippet" and its corresponding "resolved code." Figures 10 and 11 in the Appendix give examples of resolved code and a self-contained code snippet, respectively.

Our dataset also includes 33 benign-labeled sequences, constructed by randomly sampling from the same pool of benign code snippets. Their count and size distribution (i.e., number of snippets per sequence) are matched to those of the APT-labeled sequences to ensure dataset balance. These are used to evaluate whether the LLM can distinguish between APT-labeled sequences from benign ones.

5.2 APT Attack Detection

We evaluated GPT-4o's ability to distinguish APT-labeled sequences from benign ones. GPT-4o correctly classified 57 out of 66 sequences, including 29 of 33 APT-labeled and 28 of 33 benign sequences, achieving an accuracy of 86.36% and an F1 score of 86.57%. For the 8 larger APT-labeled sequences that were divided into 40 segments, GPT-4o correctly classified 34 of them (85% accuracy). In particular, GPT-4o detected attacks within the first 20% of the sequence in 6 out of the 8 cases, demonstrating its ability to recognize early-stage attack patterns to minimize damages.

5.3 Pinpointing Malicious Snippets

For each correctly classified APT-labeled sequence, we prompt GPT-4o to identify malicious code snippets and evaluate its performance using standard metrics: precision (proportion of identified snippets that are malicious) and recall (proportion of actual malicious snippets correctly identified). We evaluated the performance of GPT-4o using the 33 full attack sequences and 40 segments obtained by dividing 8 multi-stage attacks into 20% intervals. GPT-4o achieves an average precision of 0.73 and recall of 0.74 on full attack sequences, which means that it accurately identifies 74% of malicious snippets with 27% false positives. On the 40 segments, precision drops to 0.63 while recall rises to 0.81. The improved recall suggests better detection with smaller input sizes, likely due to reduced information load. The decline in precision indicates more false positives, likely due to disrupted context from segmentation, making it harder to distinguish malicious from benign snippets.

5.4 TTP Mapping

This section evaluates GPT-4o's TTP mapping performance. All 87 distinct APT snippets in the dataset are used, with each mapped to the top 10 relevant ATT&CK techniques. Performance is assessed using multiple ranking metrics for a balanced evaluation, with higher scores indicating more accurate mappings.

- **Mean Reciprocal Rank (MRR)**: Mean $\frac{1}{\text{rank}}$ of ground-truth technique across snippets (0 if not in top 10). Favors higher ranks.
- **Normalized Discounted Cumulative Gain (NDCG)**: Mean $\frac{1}{\log_2(\text{rank}+1)}$ of ground-truth technique across snippets (0 if not in top 10), normalized. Softer penalty for lower ranks.
- **Hit Rate (HR)**: Fraction of snippets where ground-truth technique appears in top 10.

We evaluate the benefit of incorporating MITRE ATT&CK technique descriptions as an external knowledge base by comparing performance across three settings: (1) no domain knowledge, (2) coarse-grained domain knowledge containing descriptions of only the 203 techniques, and (3) full domain knowledge containing descriptions of all 203 techniques and 453 sub-techniques. These technique descriptions were provided to GPT-4o using ChatGPT's file-upload feature. We also compare two tagging approaches: (1) Behavior Description Tagging, which generates a natural language behavior description before tagging (i.e., *code→behavior description→TTP*), and (2) Direct Code Tagging, which tags code snippets directly (i.e., *code→TTP*). Table 1 presents results across these three settings. The results show that GPT-4o performs best with full domain knowledge across all metrics. An MRR above 0.7 suggests that the ground-truth technique ranks highly, on average within the top two positions. The NDCG metric, which penalizes lower-ranked correct techniques less severely than MRR, also indicates strong performance (approximately 0.75). A high Hit Rate (around 0.85 to 0.9) further confirms that GPT-4o ranks the correct technique within the top 10 positions in about 85%-90% of cases.

Our results show that domain knowledge improves the performance of both tagging methods. For Behavior Description Tagging, coarse-grained knowledge significantly increases tagging accuracy, and full domain knowledge yields further improvements. In contrast, Direct Code Tagging does not benefit from coarse-grained knowledge, likely due to the lack of implementation details,

Table 1: TTP mapping performance: behavior description vs. code across three domain-knowledge levels

	MRR	HR	NDCG
Behavior Description Tagging			
No Domain Knowledge	0.492	0.666	0.535
Coarse-grained Knowledge	0.696	0.816	0.726
Full Knowledge	**0.715**	**0.908**	**0.764**
Direct Code Tagging			
No Domain Knowledge	0.635	0.816	0.680
Coarse-grained Knowledge	0.629	0.851	0.684
Full Knowledge	**0.721**	**0.873**	**0.758**

but improves with full domain knowledge. Without domain knowledge, Behavior Description Tagging performs worse than Direct Code Tagging, suggesting that the generated descriptions, though prompted to align with MITRE techniques, may not match GPT-4o's internal knowledge. With coarse-grained knowledge, Behavior Description Tagging achieves higher MRR and NDCG scores (indicating better ranking) but a lower hit rate (fewer top 10 matches) than code snippet tagging. With full domain knowledge, Behavior Description Tagging achieves a higher hit rate but slightly lower MRR and NDCG. These results suggest that, when domain knowledge is provided, neither tagging approach clearly outperforms the other.

5.4.1 Evaluation of Tagging Summary

We evaluate the tagging summaries (Figure 9) using both LLM-as-a-Judge and quantitative metrics: ROUGE (Lin, 2004) and BERTScore (Zhang* et al., 2020). In "LLM-as-a-Judge," we used GPT-4o and Llama-3.1-405B (Grattafiori et al., 2024) as judges. GPT-4o's predecessor, GPT-4, has been reported to exhibit high agreement with human judgments under task-specific criteria (Murugadoss et al., 2025). However, using the same LLM for both generation and judgment is prone to self-preference bias (Panickssery et al., 2024). As a result, we also use Llama-3.1-405B, the largest model in the Llama 3 family in task-specific judgments (Murugadoss et al., 2025).

We use ROUGE-1 and BERTScore to evaluate similarity between candidate and reference texts. ROUGE-1 measures unigram overlap, providing a simple word-level metric, while BERTScore uses contextual embeddings to assess semantic similarity, making it more robust to phrasing differences. Below, we define candidate and reference text as follows for each evaluation criterion:

- **Criterion-1:** The excerpt is the candidate and the cited source is the reference, assessing whether the excerpt appears in the source.
- **Criterion-2:** The excerpt is the candidate and the behavior description is the reference, assessing whether the excerpt captures an described behavior.
- **Criterion-3:** The tagging explanation is the candidate and the behavior description combined with the excerpt is the reference, evaluating whether the tagging explanation effectively summarizes them.

For both metrics, we use their precision scores instead of recall or F1 measures, as our criteria focus on how well the candidate captures or summarizes the reference content.

Table 2 reports average scores for the tagging summaries. For LLM-judged evaluations, we report median rubric scores, as the 1–5 scale is ordinal and better summarized by the median than the mean, which assumes equal intervals between points, a statistically contentious assumption that may lead to misleading conclusions (Jamieson, 2004). For ROUGE and BERTScore, which are continuous metrics on a 0–1 scale, we report mean values. In all cases, higher scores reflect greater relevance or validity.

Our results show strong agreement between GPT-4o and Llama-3.1-405B assessments. For Criterion 1, both LLMs assign a rubric score of Strong (4), indicating close alignment between excerpts and source content. Criterion 2 receives a median score of Sufficient (3), suggesting the excerpts reasonably capture the described behaviors. Criterion 3 achieves the highest score, Excellent (5), reflecting that tagging explanations effectively summarize both excerpts and behavior descriptions. The rubrics for Criteria 1, 2, and 3 are presented in Figures 12, 13, and 14 in the Appendix, respectively.

For Criterion 1, a ROUGE score of 0.81 indicates frequent unigram matches between excerpts and source content. The BERTScore of 0.74, the highest among all criteria, also suggests strong semantic alignment. These results are consistent with the "Strong (4)" rubric ratings from LLM-as-a-Judge. In contrast, Criterion 2 receives the lowest average scores across all metrics, indicating weaker relevance between excerpts and behavior descriptions. This is expected: Criterion 1 often involves direct excerpts from the source, while Criterion 2 relies on explanations that conceptually bridge behavior descriptions (derived from code) and domain knowledge excerpts, which may not align as closely.

For Criterion 3, LLM-as-a-Judge assigns the highest rating, but ROUGE (0.67) and BERTScore (0.68) do not fall within their highest ranges.

Table 2: Tagging summary evaluation.

Evaluation Criteria	ROUGE	BERT Sc.	LLM-as-a-Judge	
			GPT-4o	Llama-405B
Criterion 1: Excerpt Validity	0.81	0.74	4	4
Criterion 2: Excerpt-Behavior Relevance	0.54	0.65	3	3
Criterion 3: Explanation Relevance	0.67	0.68	5	5

ROUGE, which relies on n-gram matching, does not account for paraphrasing. As a result, tagging explanations that paraphrase behavior descriptions and excerpts tend to receive lower ROUGE scores. BERTScore, in contrast, rewards tagging explanations that are semantically similar to behavior descriptions and excerpts, indicating closer alignment in the embedding space through token-level contextual similarity (Zhang* et al., 2020). When explanations convey ideas using more abstract or generalized language (e.g., summarizing concepts), LLM-as-a-Judge may assign high scores based on rubric alignment, whereas BERTScore may assign lower scores due to reduced token-level similarity in embedding space.

6 Related Work

Conventional Approaches for APT Detection: Traditional APT detection methods rely on signature-based or rule-based techniques using predefined patterns (Giura and Wang, 2012; Yu et al., 2019) or statistical anomaly detection (Mees, 2012; Ioannou et al., 2013). These approaches are applied across diverse data sources, including network traffic (Villeneuve and Bennett, 2012; Lu et al., 2019), system provenance graphs (Hossain et al., 2017; Milajerdi et al., 2019), and code artifacts (Bhatt et al., 2014; Su et al., 2015), but often struggle with stealthy or unseen attacks, leading to high false-positive or false-negative rates (Krishnapriya and Singh, 2024). Recent advancements leverage machine learning (ML) to model complex patterns for improved detection, including classical models (e.g., SVMs (Chu et al., 2019), Random Forests (Do Xuan, 2021), clustering (Han et al., 2020)) and deep learning techniques (e.g., LSTMs (Du et al., 2017), CNNs (Do Xuan and Duong, 2022), GNNs (Ren et al., 2023), and Bayesian neural networks (Anjum et al., 2022)). Although ML-based methods generally outperform traditional techniques (AL-Aamri et al., 2023), they often require extensive training data and computational resources. In addition, while many ML models support explainability through tools such as SHAP and LIME (Hasan et al., 2023; Mutalib et al., 2024), or counterfactual explanations (Welter et al., 2023), the resulting explanations, such as feature attribution scores, still demand considerable human effort to accurately infer attacker intent.

LLM for Cybersecurity: LLMs have been used for various cybersecurity tasks (Zhang et al.), including software vulnerability detection (Liu et al., 2024), program repair (Nong et al., 2024), anomaly detection (Karlsen et al., 2024), secure code generation (Wang et al., 2023), threat intelligence (Hu et al., 2024) and malware detection assisted by static/code analysis (Fang et al., 2024; Patsakis et al., 2024). A few recent works use LLMs for APT detection. APT-LLM (Benabderrahmane et al., 2025) encodes process behavior descriptions from system logs using encoder-only LLMs for anomaly detection. APTSniffer (Xu et al., 2025) leverages retrieval-augmented generation by extracting relevant sequences from encrypted traffic to craft few-shot prompts for APT detection. SHIELD (Gandhi et al., 2025) analyzes suspicious nodes in provenance graphs derived from system logs and uses chain-of-thought to detect malicious processes and generate attack summaries. However, its explanations mainly reflect low-level events (e.g., a process writing to a file), which may not capture the attacker's intent. LLMs have also been used for TTP mapping, but prior work has predominantly focused on their application to natural language texts such as threat reports (Rani et al., 2023; Ali and Peng, 2024). CmdCaliper (Huang et al., 2024) uses LLMs to synthesize a command-line similarity dataset for training an embedding model, which is evaluated on a limited set of TTPs. To the best of our knowledge, we are the first to use LLMs for APT detection and TTP mapping.

7 Conclusion

In this paper, we explored the use of large language models (LLMs) for APT detection, TTP mapping, and explanation generation based on executed code snippets collected from host machines. We developed evaluation datasets using open-source resources, including MITRE Caldera and publicly available PowerShell snippets. Our experiments show that while LLMs exhibit moderate effectiveness in detecting APT activity and identifying malicious code, they perform well in TTP mapping when supplemented with domain knowledge.

8 Limitations

Our dataset simulates a moderately active user environment, with a $2:1$ ratio of benign user code snippets to APT snippets (Section 5.1). In the future, we plan to explore high-activity scenarios, where benign snippets greatly outnumber APT snippets. In such cases, APT activity may be sparser and fall outside the LLM's context window, posing a key challenge for detection. In addition, all experiments in this work are conducted using GPT-4o, with the exception of the "LLM-as-a-Judge" evaluation (Section 5.4.1), which includes both GPT-4o and LLaMA-405B. While this study focuses on GPT-4o, future research could extend our methodology to other LLMs to assess its generalizability.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Abdullah Said AL-Aamri, Rawad Abdulghafor, Sherzod Turaev, Imad Al-Shaikhli, Akram Zeki, and Shuhaili Talib. 2023. Machine learning for apt detection. *Sustainability*, 15(18):13820.

Asad Ali and Min-Chun Peng. 2024. Ttpmapper: Accurate mapping of ttps from unstructured cti reports. In *2024 IEEE International Conference on Future Machine Learning and Data Science (FMLDS)*, pages 558–563. IEEE.

Md Monowar Anjum, Shahrear Iqbal, and Benoit Hamelin. 2022. Anubis: a provenance graph-based framework for advanced persistent threat detection. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 1684–1693.

Sidahmed Benabderrahmane, Petko Valtchev, James Cheney, and Talal Rahwan. 2025. Apt-llm: Embedding-based anomaly detection of cyber advanced persistent threats using large language models. *arXiv preprint arXiv:2502.09385*.

Parth Bhatt, Edgar Toshiro Yano, and Per Gustavsson. 2014. Towards a framework to detect multi-stage advanced persistent threats attacks. In *2014 IEEE 8th international symposium on service oriented system engineering*, pages 390–395. IEEE.

S Sibi Chakkaravarthy, V Vaidehi, and P Rajesh. 2018. Hybrid analysis technique to detect advanced persistent threats. *International Journal of Intelligent Information Technologies (IJIIT)*, 14(2):59–76.

ChooseALicense Contributors. 2024. MIT License (Markdown Format). `https://huggingface.co/datasets/choosealicense/licenses/blob/main/markdown/mit.md`. License: MIT. Accessed 2024-05-19.

Wen-Lin Chu, Chih-Jer Lin, and Ke-Neng Chang. 2019. Detection and classification of advanced persistent threats and attacks using the support vector machine. *Applied Sciences*, 9(21):4579.

Ivan Čík, Andrindrasana David Rasamoelina, Marián Mach, and Peter Sinčák. 2021. Explaining deep neural network using layer-wise relevance propagation and integrated gradients. In *2021 IEEE 19th world symposium on applied machine intelligence and informatics (SAMI)*, pages 000381–000386. IEEE.

The MITRE Corporation. a. Caldera.

The MITRE Corporation. b. Mitre att&ck.

The MITRE Corporation. c. Mitre att&ck: T1136 - create account.

The MITRE Corporation. d. Mitre att&ck: T1136.001 - create account: Local account.

The MITRE Corporation. e. Mitre att&ck version history.

Cho Do Xuan. 2021. Detecting apt attacks based on network traffic using machine learning. *Journal of Web Engineering*, 20(1):171–190.

Cho Do Xuan and Duc Duong. 2022. Optimization of apt attack detection based on a model combining attention and deep learning. *Journal of Intelligent & Fuzzy Systems*, 42(4):4135–4151.

Feng Dong, Liu Wang, Xu Nie, Fei Shao, Haoyu Wang, Ding Li, Xiapu Luo, and Xusheng Xiao. 2023. {DISTDET}: A {Cost-Effective} distributed cyber threat detection system. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6575–6592.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298.

Elastic. 2025. Crowdstrike exported fields. Accessed: 2025-05-01.

Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Asmita, Ryan Tsang, Najmeh Nazari, Han Wang, and Houman Homayoun. 2024. Large language models for code analysis: Do LLMs really do their job? In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 829–846, Philadelphia, PA. USENIX Association.

Yong Fang, Xiangyu Zhou, and Cheng Huang. 2021. Effective method for detecting malicious powershell scripts based on hybrid features. *Neurocomputing*, 448:30–39.

Mohamed Amine Ferrag, Fatima Alwahedi, Ammar Battah, Bilel Cherif, Abdechakour Mechri, and Norbert Tihanyi. 2024. Generative ai and large language models for cyber security: All insights you need. *Available at SSRN 4853709*.

Fleschutz. 2023. Mega collection of powershell scripts. GitHub Repository. Accessed: March 12, 2025.

Christoph Fleschutz. 2024. MegaPowerShellCollection. `https://github.com/fleschutz/PowerShell`. License: CC0 1.0. `https://github.com/fleschutz/PowerShell?tab=CC0-1.0-1-ov-file#readme`. Accessed 2024-05-19.

Parth Atulbhai Gandhi, Prasanna N Wudali, Yonatan Amaru, Yuval Elovici, and Asaf Shabtai. 2025. Shield: Apt detection and intelligent explanation using llm. *arXiv preprint arXiv:2502.02342*.

Paul Giura and Wei Wang. 2012. A context-based detection framework for advanced persistent threats. In *2012 International Conference on Cyber Security*, pages 69–74. IEEE.

Google Research. 2020. Google Research Apache 2.0 License. `https://github.com/google-research/google-research/blob/master/LICENSE`. Accessed: 2025-05-19.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Saul Greenberg. 1988. Using unix: Collected traces of 168 users.

Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525*.

Md Mahadi Hasan, Muhammad Usama Islam, and Jasim Uddin. 2023. Advanced persistent threat identification with boosting and explainable ai. *SN Computer Science*, 4(3):271.

Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. 2017. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 487–504.

Yuelin Hu, Futai Zou, Jiajia Han, Xin Sun, and Yilei Wang. 2024. Llm-tikg: Threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*, 145:103999.

Sian-Yao Huang, Cheng-Lin Yang, Che-Yu Lin, and Chun-Ying Huang. 2024. CmdCaliper: A semantic-aware command-line embedding model and dataset for security research. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20188–20206.

Infinite Dataset Hub. 2024. Real powershell scripts dataset. Hugging Face Datasets. Accessed: March 12, 2025.

Georgios Ioannou, Panos Louvieris, Natalie Clewley, and Gavin Powell. 2013. A markov multi-phase transferable belief model: An application for predicting data exfiltration apts. *Proceedings of the 16th International Conference on Information Fusion*, pages 842–849.

Susan Jamieson. 2004. Likert scales: How to (ab) use them? *Medical education*, 38(12):1217–1218.

Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. 2024. {MAGIC}: Detecting advanced persistent threats via masked graph representation learning. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5197–5214.

Egil Karlsen, Xiao Luo, Nur Zincir-Heywood, and Malcolm Heywood. 2024. Benchmarking large language models for log analysis, security, and interpretation. *J. Netw. Syst. Manag.*, 32(3).

Singamaneni Krishnapriya and Sukhvinder Singh. 2024. A comprehensive survey on advanced persistent threat (apt) detection techniques. *Computers, Materials & Continua*, 80(2).

Terran Lane and Carla E Brodley. 1997. An application of machine learning to anomaly detection. In *Proceedings of the 20th national information systems security conference*, volume 377, pages 366–380. Baltimore, USA.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2024. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Shaofei Li, Feng Dong, Xusheng Xiao, Haoyu Wang, Fei Shao, Jiedong Chen, Yao Guo, Xiangqun Chen, and Ding Li. 2023a. Nodlink: An online system for fine-grained apt attack detection and investigation. *arXiv preprint arXiv:2311.02331*.

Xiang Li, Tingting Zhu, and Wenbo Zhang. 2023b. Efficient ransomware detection via portable executable file image analysis by llama-7b.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. *arXiv preprint arXiv:1802.08979*.

Peiyu Liu, Junming Liu, Lirong Fu, Kangjie Lu, Yifan Xia, Xuhong Zhang, Wenzhi Chen, Haiqin Weng, Shouling Ji, and Wenhai Wang. 2024. Exploring {ChatGPT's} capabilities on vulnerability management. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 811–828.

Zefang Liu and John Buford. 2023. Anomaly detection of command shell sessions based on distilbert: Unsupervised and supervised approaches. *arXiv preprint arXiv:2310.13247*.

Jiazhong Lu, Kai Chen, Zhongliu Zhuo, and XiaoSong Zhang. 2019. A temporal correlation and traffic analysis approach for apt attacks detection. *Cluster computing*, 22:7347–7358.

Johnathan May. 2023. Common powershell commands. GitHub Gist. Accessed: March 12, 2025.

Wim Mees. 2012. Multi-agent anomaly-based apt detection. In *Proceedings of Information Systems Technology Panel Symposium*, volume 16.

Meta Platforms, Inc. 2024. LLaMA 3.1 405B License. `https://huggingface.co/meta-llama/Llama-3.1-405B/blob/main/LICENSE`. Accessed: 2025-05-19.

Microsoft. 2021. DeBERTa XLarge fine-tuned on MNLI. `https://huggingface.co/microsoft/deberta-xlarge-mnli`. License: MIT. Accessed: 2025-05-19.

Microsoft. 2024. About logging in windows - powershell. Microsoft Learn. Accessed: March 12, 2025.

Microsoft. 2025. Overview of sysmon capabilities. Accessed: 2025-05-01.

Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. 2019. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE symposium on security and privacy (SP)*, pages 1137–1152. IEEE.

MITRE Corporation. 2024. Caldera. `https://github.com/mitre/caldera`. License: Apache 2.0. `https://github.com/mitre/caldera/blob/master/LICENSE`. Accessed 2024-05-19.

Bhuvanashree Murugadoss, Christian Poelitz, Ian Drosos, Vu Le, Nick McKenna, Carina Suzana Negreanu, Chris Parnin, and Advait Sarkar. 2025. Evaluating the evaluator: Measuring llms' adherence to task evaluation instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19589–19597.

Noor Hazlina Abdul Mutalib, Aznul Qalid Md Sabri, Ainuddin Wahid Abdul Wahab, Erma Rahayu Mohd Faizal Abdullah, and Nouar AlDahoul. 2024. Explainable deep learning approach for advanced persistent threats (apts) detection in cybersecurity: a review. *Artificial Intelligence Review*, 57(11):297.

Yu Nong, Haoran Yang, Long Cheng, Hongxin Hu, and Haipeng Cai. 2024. Automated software vulnerability patching using large language models. *arXiv preprint arXiv:2408.13597*.

Momoka Okuma, Koki Watarai, Satoshi Okada, and Takuho Mitsunaga. 2023. Automated mapping method for sysmon logs to att&ck techniques by leveraging atomic red team. In *2023 6th International Conference on Signal Processing and Information Security (ICSPIS)*, pages 104–109. IEEE.

OpenAI. 2024. Terms of Use. `https://openai.com/policies/row-terms-of-use/`. Accessed: 2025-05-19.

Arjun Panickssery, Samuel Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802.

Constantinos Patsakis, Fran Casino, and Nikolaos Lykousas. 2024. Assessing llms in malicious code deobfuscation of real-world malware campaigns. *Expert Systems with Applications*, 256:124912.

Xingzhi Qian, Xinran Zheng, Yiling He, Shuo Yang, and Lorenzo Cavallaro. 2025. Lamd: Context-driven android malware detection and classification with llms. *arXiv preprint arXiv:2502.13055*.

Nanda Rani, Bikash Saha, Vikas Maurya, and Sandeep Kumar Shukla. 2023. Ttphunter: Automated extraction of actionable intelligence as ttps from narrative threat reports. In *Proceedings of the 2023 Australasian Computer Science Week*, pages 126–134.

Weiwu Ren, Xintong Song, Yu Hong, Ying Lei, Jinyu Yao, Yazhou Du, and Wenjuan Li. 2023. Apt attack detection based on graph convolutional neural networks. *International Journal of Computational Intelligence Systems*, 16(1):184.

Google Research. 2020. Rouge scoring implementation. GitHub repository.

Duraid Thamer Salim, Manmeet Mahinderjit Singh, and Pantea Keikhosrokiani. 2023. A systematic literature review for apt detection and effective cyber situational awareness (ecsa) conceptual model. *Heliyon*, 9(7).

Matthias Schonlau, William DuMouchel, Wen-Hua Ju, Alan F Karr, Martin Theus, and Yehuda Vardi. 2001. Computer intrusion: Detecting masquerades. *Statistical science*, pages 58–74.

M Scott, Lee Su-In, et al. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30:4765–4774.

Abhishek Sidhardhan, S Keerthana, and Jinesh M Kannimoola. 2023. Weaponizing real-world applications as c2 (command and control). In *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*, pages 458–463. IEEE.

Splunk. 2023. Hunting for malicious powershell using script block logging. Accessed: 2025-02-19.

Splunk. 2024. Endpoint detection - powershell script block logging. Splunk Research. Accessed: March 12, 2025.

Yunfei Su, Mengjun Li, ChaoJing Tang, and Rongjun Shen. 2015. A framework of apt detection based on dynamic analysis. In *2015 4th National Conference on Electrical, Electronics and Computer Engineering*, pages 1047–1053. Atlantis Press.

Xiaoyi Tian, Amogh Mannekote, Carly E Solomon, Yukyeong Song, Christine Fry Wise, Tom Mcklin, Joanne Barrett, Kristy Elizabeth Boyer, and Maya Israel. 2024. Examining llm prompting strategies for automatic evaluation of learner-created computational artifacts. In *Proceedings of the 17th International Conference on Educational Data Mining*, pages 698–706.

Tianyi Zhang. 2020. BERTScore MIT License. `https://github.com/Tiiiger/bert_score/blob/master/LICENSE`. Accessed: 2025-05-19.

Unknown. 2024. CommonPowerShellCommands. `<INSERT-REPO-URL>`. No license explicitly specified. Accessed 2024-05-19.

Nart Villeneuve and James Bennett. 2012. Detecting apt activity with network traffic analysis. *Trend Micro Incorporated Research Paper*, pages 1–13.

Vaishali Vinay and Anjali Mangal. 2024. Scade: Scalable command-line anomaly detection engine. *arXiv preprint arXiv:2412.04259*.

Jiexin Wang, Liuwen Cao, Xitong Luo, Zhiping Zhou, Jiayuan Xie, Adam Jatowt, and Yi Cai. 2023. Enhancing large language models for secure code generation: A dataset-driven study on vulnerability mitigation. *arXiv preprint arXiv:2310.16263*.

Yuntao Wang, Han Liu, Zhendong Li, Zhou Su, and Jiliang Li. 2024. Combating advanced persistent threats: Challenges and solutions. *IEEE Network*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Felix Welter, Florian Wilkens, and Mathias Fischer. 2023. Tell me more: Black box explainability for apt detection on system provenance graphs. In *ICC 2023-IEEE International Conference on Communications*, pages 3817–3823. IEEE.

Laslo Welz and Carsten Lanquillon. 2024. Enhancing large language models through external domain knowledge. In *International Conference on Human-Computer Interaction*, pages 135–146. Springer.

Hongbo Xu, Chengxiang Si, Chenxu Wang, Peishuai Sun, Qingyun Liu, et al. 2025. Aptsniffer: Detecting apt attack traffic using retrieval-augmented large language models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Kevin P Yancey, Geoffrey Laflair, Anthony Verardi, and Jill Burstein. 2023. Rating short l2 essays on the cefr scale with gpt-4. In *Proceedings of the 18th workshop on innovative use of NLP for building educational applications (BEA 2023)*, pages 576–584.

Han Yu, Aiping Li, and Rong Jiang. 2019. Needle in a haystack: attack detection from large-scale system audit. In *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pages 1418–1426. IEEE.

Zhengmin Yu, Jiutian Zeng, Siyi Chen, Wenhan Xu, Dandan Xu, Xiangyu Liu, Zonghao Ying, Nan Wang, Yuan Zhang, and Min Yang. 2024. Cs-eval: A comprehensive large language model benchmark for cybersecurity. *arXiv preprint arXiv:2411.16239*.

Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, et al. 2024a. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. *arXiv preprint arXiv:2408.08926*.

Bo Zhang, Yansong Gao, Boyu Kuang, Changlong Yu, Anmin Fu, and Willy Susilo. 2024b. A survey on advanced persistent threat detection: A unified framework, challenges, and countermeasures. *ACM Comput. Surv.*

Jie Zhang, Haoyu Bu, Hui Wen, Y Liu, H Fei, R Xi, L Li, Y Yang, H Zhu, and D Meng. When llms meet cybersecurity: a systematic literature review (2024). *arXiv preprint arXiv:2405.03644*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. GitHub repository.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Appendix

.1 Implementation and Dataset Details

This section provides additional details on the implementation and dataset used in the experiments described in Section 5.

.1.1 Implementation Details

For the experiments in Sections 5.2, 5.3, and 5.4, we used the pre-trained GPT-4o model solely for inference without fine-tuning. The model's architecture details, including the number of parameters and hyperparameter configurations, are proprietary to OpenAI. In Section 5.4.1, we used GPT-4o, LLaMA-3.1-405B, ROUGE, and BERTScore together for evaluation. Both LLMs were used exclusively for inference. LLaMA-3.1-405B, developed by Meta AI, is open-source and contains approximately 405 billion parameters, supporting transparency and reproducibility. ROUGE was computed using Google's rouge-score library (Research, 2020) with the ROUGE-1 metric and default settings. As a symbolic n-gram overlap metric, it does not involve learned parameters. BERTScore was calculated using the bert-score library (Zhang et al., 2019) with the recommended "microsoft/deberta-xlarge-mnli" model (approximately 750 million parameters (Microsoft, 2021)), which the authors specify as having the best correlation with human evaluation (Zhang et al., 2019). All other settings were kept at their defaults. ROUGE and BERTScore were run locally on a machine with a 2.8 to 3.35 GHz AMD EPYC 7402P CPU and an NVIDIA Tesla T4 GPU with 15 GB of memory. CUDA was enabled. No hyperparameter search was conducted. All tools were used with their default configurations in an inference-only setup. All software and models were used in accordance with their respective licenses and intended purposes. This includes OpenAI's Terms of Use for GPT-4o (OpenAI, 2024), Meta's open-source license for LLaMA-3.1-405B (Meta Platforms, Inc., 2024), and the Apache 2.0 (Google Research, 2020) and MIT (Tianyi Zhang, 2020) licenses for ROUGE and BERTScore, respectively, which permit use for scientific and research purposes. All reported results are based on single runs.

.1.2 Dataset Details

As specified in Section 5.1, our dataset is constructed using publicly available, open-source resources and repositories. In particular, we extracted code snippets from the MITRE Caldera framework, which is distributed under the permissive Apache 2.0 License (MITRE Corporation, 2024) and is explicitly intended for research and academic purposes. Additionally, benign PowerShell code snippets were collected from open-source repositories released under permissive licenses, including CC0 1.0 (Fleschutz, 2024) and the MIT License (ChooseALicense Contributors, 2024). One such repository does not explicitly specify a license (Unknown, 2024) but is publicly accessible and was used in accordance with standard academic practices. While the dataset includes code snippets that could potentially be used for malicious purposes, they are sourced directly from the Caldera repository (Corporation, a) and do not introduce any new or unpublished malicious content. These examples are sanitized using metadata placeholders (e.g., `{IP-address}`) and contain no personally identifiable information. For evaluation purposes, the placeholders are replaced with arbitrary but realistic values generated using GPT-4o, ensuring both semantic and contextual coherence. Our dataset has been carefully curated for academic and research use. Upon publication of this work, we intend to release it under the MIT License.

.2 AI Assistance

We used ChatGPT to lightly polish the writing in some sections of the paper. This included minor edits for clarity, grammar, and phrasing, without affecting the technical content or analysis.

> *"The code snippets exhibit several concerning behaviors that suggest the execution of an adversarial operation. The use of* `cmd.exe` *with* `paexec.exe` *indicates an attempt to execute a command on a remote machine, which is a common tactic in lateral movement during cyber intrusions.*
> *The inclusion of credentials (*`-u hospitality kmitnick -p c3aa78cd0827b8e8`*) further solidifies this as a potential malicious operation, as it implies unauthorized access attempts. Additionally, the* `robocopy` *command is used to replace a system file (*`srrstr.dll`*), which is indicative of attempts to compromise system integrity.*
> *The overall sequence of copying files from sensitive directories and executing commands remotely aligns with known malicious methods, suggesting a coordinated effort to exfiltrate data or establish persistence."*

Figure 6: GPT-generated explanation of detected APT activity

Level	Description
4 (Clearly)	**The code snippet clearly exhibit malicious intent or is part of coordinated malicious activity through direct associations.** For example, a snippet utilizing credential dumping tools commonly associated with cyberattacks, or a snippet that compresses sensitive files into an archive followed by another snippet that exfiltrates the archive to an external location is in this category.
3 (Likely)	**The code snippet is likely part of the malicious chain. While not explicitly malicious on its own, its involvement in the attack can be inferred through indirect links to other suspicious snippets.** For example, a snippet that modifies firewall rules to permit remote connections may not appear directly malicious, but if a subsequent snippet establishes an unauthorized remote connection, it indirectly contributes to the attack.
2 (Possibly)	**The code snippet performs actions common in legitimate contexts, but could have a plausible connection to the malicious chain. This level captures supporting actions in APT attacks that, while appearing benign on their own, contribute to a broader malicious objective.** For example, a commonly used discovery command (e.g., nmap) that maps an environment could potentially contribute to the malicious chain.
1 (Unlikely)	**The code snippet is benign, with no dual-use potential and no plausible connection to the malicious chain.** Its benign nature is further supported by the lack of any association with suspicious snippets.

Figure 7: Pinpoint Malicious Snippets - APT Domain-Expertise Informed Rubric

Code Snippet

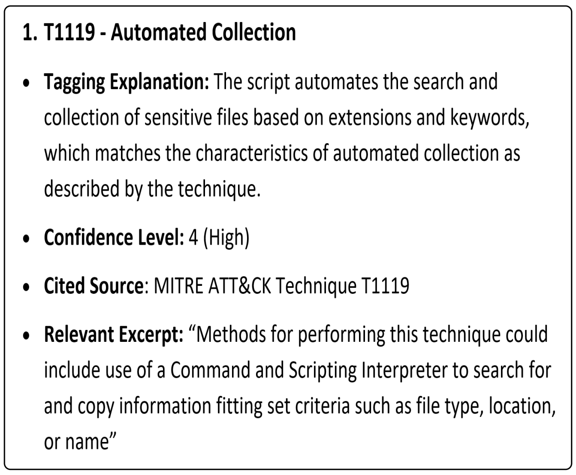
```
$env:APPDATA;
$files = ChildItem -Path $env:USERPROFILE \
    -Include "*.doc", "*.docx", "*.xls", "*.
         xlsx", "*.ppt", "*.pptx", "*.pps", "*.
         wps", "*.wpd", "*.odt", "*.lwp", "*.
         jtd", "*.pdf", "*.zip", "*.rar", "*.
         docx", "*.url", "*.xlsx", "*.pptx", "*.
         ppc", "*.pst", "*.ost", "*.psw", "*
         pass*", "*login*", "*admin*", "*sif*",
          "*sifer*", "*.vpn", "*.jpg", "*.txt",
          "*.lnk" \
    -Recurse \
    -ErrorAction SilentlyContinue |
    Select -ExpandProperty FullName;
Compress-Archive -LiteralPath $files \
    -CompressionLevel Optimal \
    -DestinationPath $env:APPDATA\Draft.Zip \
    -Force
```

Behavior Description

This PowerShell script systematically searches a user's home directory and its subdirectories for a variety of files, including documents, archives, email data, images, and text files, while also specifically targeting filenames associated with credentials, administrative access, and VPN configurations. Once identified, the files are collected and compressed into a single ZIP archive, Draft.Zip, which is stored in the AppData directory, a hidden system folder often overlooked by users. By suppressing errors and recursively scanning directories, the script ensures it gathers as many files as possible without interruption, even if certain locations are restricted.

The script does not initiate exfiltration but effectively stages sensitive data in a single compressed archive, making it easy to transfer through subsequent actions. Storing the archive in the AppData folder minimizes visibility and reduces the likelihood of immediate detection. The capability to systematically locate, collect, and consolidate files into an archive suggests an intent to facilitate data theft, unauthorized access, or operational persistence. While the script does not explicitly exhibit destructive behavior, its ability to aggregate potentially sensitive information—especially files containing credentials or login details—poses a significant risk to confidentiality and security.

Figure 8: GPT-generated behavior description of a code snippet.



1. T1119 - Automated Collection

- **Tagging Explanation:** The script automates the search and collection of sensitive files based on extensions and keywords, which matches the characteristics of automated collection as described by the technique.
- **Confidence Level:** 4 (High)
- **Cited Source**: MITRE ATT&CK Technique T1119
- **Relevant Excerpt:** "Methods for performing this technique could include use of a Command and Scripting Interpreter to search for and copy information fitting set criteria such as file type, location, or name"

Figure 9: Tagging Summary

Code-Snippet:

```
./post.ps1; email;
```

Resolved Code:

```
function email { Send-Mail ... };
```

Figure 10: Example where dependencies from external scripts or functions are resolved at execution time.

Code-Snippet:

```
schtasks /create /sc "DAILY" /tn "daily
task" /tr "C:" /ru "john doe"
```

Resolved Code:

N/A

Figure 11: Example where the executed code is self-contained (no external dependencies).

Rating	**Description**
5 (Excellent)	The 'Relevant Excerpt" is quoted verbatim and appears exactly in the "Cited Source," without any alterations or discrepancies.
4 (Strong)	The 'Relevant Excerpt" is paraphrased but accurately reflects subset(s) of the "Cited Source," without adding or omitting information that would alter the meaning or intent.
3 (Satisfactory)	The 'Relevant Excerpt" is paraphrased and largely reflects subset(s) of the "Cited Source," but with some minor differences or omissions that slightly alter the meaning or intent.
2 (Weak)	The 'Relevant Excerpt" is paraphrased and partially reflects subset(s) of the "Cited Source," but with significant differences, omissions, or misrepresentations that noticeably alter the meaning or intent.
1 (None)	The "Relevant Excerpt" fails to reflect any subset of the "Cited Source," being fabricated, missing, or entirely misaligned.

Figure 12: Criterion 1 (Excerpt Validity)

Rating	Description
5 (Excellent)	The “Relevant Excerpt” directly and accurately addresses the core behavior outlined in the “behavior description” with no ambiguity or gaps.
4 (Strong)	The “Relevant Excerpt” effectively addresses the core behavior outlined in the “behavior description,” accommodating minor variations that do not significantly affect relevance or clarity.
3 (Satisfactory)	The “Relevant Excerpt” adequately addresses the core behavior outlined in the “behavior description,” though some aspects are unclear, missing, or loosely connected.
2 (Weak)	The “Relevant Excerpt” only partially addresses the core behavior outlined in the “behavior description,” offering minimal relevance or clarity.
1 (None)	The “Relevant Excerpt” fails to address or relate to the core behavior outlined in the “behavior description” in any meaningful way.

Figure 13: Criterion 2 (Excerpt-Behavior Alignment)

Rating	Description
5 (Excellent)	The “Tagging Explanation” is directly relevant and effectively addresses both the “Behavior Description” and the “Relevant Excerpt,” offering a complete and well-justified rationale for the tag.
4 (Strong)	The “Tagging Explanation” is substantially relevant and meaningfully addresses both the “Behavior Description” and the “Relevant Excerpt,” though it may have minor omissions or slightly less depth than a comprehensive justification.
3 (Satisfactory)	The “Tagging Explanation” is generally relevant and adequately addresses both the “Behavior Description” and the “Relevant Excerpt,” but it may not explore key aspects in depth or rely on less specific evidence.
2 (Weak)	The “Tagging Explanation” is loosely relevant and only weakly addresses either the “Behavior Description” or the “Relevant Excerpt,” providing an unclear or insufficient rationale.
1 (None)	The “Tagging Explanation” is irrelevant and does not address either the “Behavior Description” or the “Relevant Excerpt.”

Figure 14: Criterion 3 (Explanation Relevance)