

TALLINN UNIVERSITY OF TECHNOLOGY  
School of Information Technologies

ARTUR NIKITCHUK 221916IVCM

**ADVANCING CYBER THREAT DETECTION THROUGH  
HOST-BASED SECURITY LOG ANALYSIS WITH LARGE  
LANGUAGE MODELS**

Master's Thesis

Supervisor: Prof. Hayretin Bahsi  
PhD

Tallinn 2025

TALLINNA TEHNIKAÜLIKOOL  
Infotehnoloogia teaduskond

ARTUR NIKITCHUK 221916IVCM

**KÜBEROHTUDE TUVASTAMISE EDENDAMINE SUURTE  
KEELEMUDELITE PÕHISE TURVALOGI ANALÜÜSI ABIL**

Magistritöö

Juhendaja: Prof. Hayretdin Bahsi  
PhD

Tallinn 2025

## **Author's Declaration of Originality**

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: ARTUR NIKITCHUK

12.01.2025

## Abstract

The increasing complexity of cyber threats, particularly the rise of "Living off the Land" (LOTL) attacks, presents significant challenges to traditional detection methods. LOTL attacks exploit legitimate tools and processes within target environments, often bypassing conventional signature-based systems. Although machine learning models have shown potential in identifying malicious activity through behavioral analysis, their deployment is resource-intensive and often inaccessible to many organizations.

This research explores the potential of Large Language Models, such as GPT-4o, to detect and classify malicious activities based on Windows Event logs, providing a novel approach to enhancing incident response and digital forensics processes. Using the Atomic Red Team framework, a custom dataset was developed, simulating 1064 distinct attack techniques. This dataset includes logs from Sysmon, Security, System, PowerShell, and Application sources, offering a comprehensive resource for experimentation.

Various prompting strategies such as zero-shot, one-shot, few-shot, Chain of Thought (CoT), and CoT+5-shot were tested to evaluate LLM performance in two primary tasks: detecting malicious activities and classifying them within the MITRE ATT&CK framework. Results reveal that LLMs, when optimized with effective prompting strategies, can detect malicious activities in Windows Event logs, reducing the burden of manual analysis. However, challenges remain in accurately classifying these activities into predefined categories.

This study demonstrates that LLMs have the potential to bridge the gap between large-scale log analysis and actionable threat intelligence, offering a scalable and efficient method for detecting evolving cyber threats. Additionally, it introduces a novel open-source dataset and a customizable script for generating Windows attack logs, enabling further experimentation and adaptation across diverse environments. By leveraging LLMs, this research advances state-of-the-art log-based threat detection and contributes to making cybersecurity practices more accessible and efficient.

The thesis is written in English and is 117 pages long, including 7 chapters, 11 figures and 45 tables.

## **Annotatsioon**

### **Küberohtude tuvastamise edendamine suurte keelemudelite põhise turvalogi analüüsi abil**

Küberohtude keerukus, eriti "Living off the Land" (LOTL) rünnakud, seab traditsioonilistele tuvastusmeetoditele tõsiseid väljakutseid. LOTL-rünnakud kasutavad sihtkeskkonna tööriistu, möödudes sageli signatuuripõhistest süsteemidest. Kuigi masinõppe mudelid suudavad tuvastada pahatahtlikku tegevust, on nende rakendamine ressursimahukas ja paljudele organisatsioonidele raskesti kättesaadav.

See uurimus uurib suurte keelemudelite, nagu GPT-4o, võimet tuvastada ja klassifitseerida pahatahtlikke tegevusi Windowsi Event logide põhjal, parandades intsidentidele reageerimist ja kohtuekspertiisi protsesse. Atomic Red Team raamistikuga loodi 1064 rünnakutehnikat simuleeriv andmestik, mis hõlmab logisid Sysmoni, Security, Systemi, PowerShell'i ja rakenduste logidest, pakkudes ulatuslikku katseresurssi.

LLM-ide efektiivsuse hindamiseks testiti erinevaid prompt strateegiaid, sealhulgas zero-shot, one-shot, few-shot, Chain of Thought (CoT) ja CoT+5-shot. Uuring keskendus kahele peamisele ülesandele: pahatahtlike tegevuste tuvastamine ja nende klassifitseerimine MITRE ATT&CK raamistikus. Tulemused näitavad, et LLM-id, kui neid optimeerida sobivate prompt strateegiatega, suudavad tuvastada pahatahtlikke tegevusi Windowsi Event logide põhjal, vähendades käsitsi analüüsi koormust. Siiski jäävad väljakutsed nende tegevuste täpselt klassifitseerimiseks eelmääratud kategooriatesse.

See uuring näitab, et LLM-id suudavad tõhusalt ületada suurte logianalüüsise ja praktilise ohuluure vahelisi lünki, pakkudes skaleeritavat lahendust küberohtude tuvastamiseks. Samuti tutvustab see avatud lähtekoodiga andmestikku ja skripti Windowsi ründelogeide genereerimiseks, toetades edasisi katsetusi ja kohandusi. LLM-ide rakendamine parandab logipõhiste ohtude tuvastamist ja muudab küberjulgeoleku tõhusamaks ning kättesaadavamaks.

Lõputöö on kirjutatud inglise keeles ja on 117 lehekülge pikk, sealhulgas 7 peatükki, 11 joonist ja 45 tabelit.

## List of Abbreviations and Terms

LLM	Large Language Model
IR	Incident Response
SOC	Security Operations Center
DF	Digital Forensics
DFIR	Digital Forensics and Incident Response
TTP	Tactics, Techniques and Procedures
PS	Powershell
LotL	Living-Off-The-Land
VM	Virtual Machine
VBox	Virtual Box
Snapshot	A point-in-time backup of a Virtual Machine
Atomic	A single adversary emulation script snippet from the Atomic Red Team Framework
IOC	Indicator of Compromise
IOA	Indicator of Attack
Token	Character sequences, utilized by Large Language Models to segment text into processable units
EDR	Endpoint Detection and Response
CoT	Chain of Thought prompt

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>10</b>
1.1	Research Problem	11
1.2	Contributions & Novelty	12
1.3	Thesis Structure	13
<b>2</b>	<b>Background information</b>	<b>14</b>
2.1	Large Language Models	14
2.2	Incident Response and Digital Forensics	15
2.3	Red Teaming and Adversary Emulation	16
<b>3</b>	<b>Related Work</b>	<b>17</b>
3.1	LLMs In Cybersecurity	17
<b>4</b>	<b>Method</b>	<b>22</b>
4.1	Dataset Generation	22
4.1.1	Dataset Generation Methodology	23
4.1.2	Controller & Target Machine Configuration	24
4.1.3	Network Architecture	26
4.1.4	Attack Simulation & Log Collection Toolchain	29
4.1.5	Benign Dataset	36
4.2	Data Pre-processing	37
4.3	Large Language Models	43
4.3.1	Model Selection	44
4.3.2	Foundational Prompts	45
4.3.3	Few-shot Prompts	49
4.3.4	Chain of Thought Prompts	53
4.4	Evaluation and Performance Metrics	58
<b>5</b>	<b>Results</b>	<b>61</b>
5.1	Dataset Comparison	61
5.2	Malicious Activity Binary Detection	64
5.2.1	Zero-shot	64
5.2.2	Few-Shot	67
5.2.3	Chain of Thought	74
5.2.4	Few-shot + CoT	77
5.2.5	Binary Detection Discussion	79

5.3	Malicious Activity Multiclass Classification . . . . .	82
5.3.1	Zero-shot . . . . .	82
5.3.2	Few-Shot . . . . .	85
5.3.3	Chain of Thought . . . . .	92
5.3.4	Few-shot + CoT . . . . .	94
5.3.5	Multiclass Classification Discussion . . . . .	96
<b>6</b>	<b>Discussion . . . . .</b>	<b>98</b>
6.1	Limitations . . . . .	100
6.2	Future Work . . . . .	101
<b>7</b>	<b>Conclusion . . . . .</b>	<b>103</b>
	<b>References . . . . .</b>	<b>105</b>
	<b>Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis . . . . .</b>	<b>109</b>
	<b>Appendix 2 – Dataset Generation Script . . . . .</b>	<b>110</b>



## List of Figures

1	Network Diagram of The Dataset Generation Setup . . . . .	28
2	BPMN diagram of the dataset generation process . . . . .	33
3	Initial Count of Simulated Tactic Categories . . . . .	35
4	Final Count of Simulated Tactic Categories . . . . .	36
5	ChatGPT Knowing What Is Atomic Red Team Framework . . . . .	38
6	ChatGPT Identifies Atomic Red Team Artifacts, Suggesting The Likeli- hood Of A Security Test In Progress . . . . .	39
7	Example of TTP ID Being Leaked Inside A Log Due To Atomic Red Team Execution . . . . .	40
8	ChatGPT Recognizing MITRE ATT&CK TTP IDs With 0 Additional Context . . . . .	41
9	An Example Of Attacker Tooling Artifacts Within The Generated Dataset	42
10	Example Of Few-Shot Prompt . . . . .	49
11	Example Of A Chain-of-Thought Prompt . . . . .	53

## List of Tables

1	Attacker Tool And Script Names Replaced From The Dataset . . . . .	42
2	Pre-Attack Log Clearing Artifacts . . . . .	43
3	Performance Comparison Of LLM Models Across Benchmarks MMLU, GPQA, MATH, HumanEval, MGSM, DROP (F1, 3-shot), SimpleQA, And Associated Costs For Input And Output Tokens (OpenAI, 2024a) . . . . .	45
4	LLM Analysis Detection Results Classification Logic . . . . .	58
5	Zero-shot Detection Task Results On <code>evtx-unfiltered</code> . . . . .	62
6	Zero-shot Detection Task Results On <code>evtx-atomic-removed</code> . . . . .	62
7	Zero-shot Detection Task Results On <code>evtx-atomic-and-tools-removed</code> . . . . .	63
8	Zero-shot Binary Detection Task Results . . . . .	65
9	Zero-shot Binary Detection Task Results Summary Table . . . . .	66
10	One-shot Binary Detection Task Results . . . . .	67
11	One-shot Binary Detection Task Results Summary Table . . . . .	67
12	Comparison of Recall Scores Between Zero-shot And 1-shot Binary Detection Prompts . . . . .	68
13	3-shot Binary Detection Task Results . . . . .	70
14	3-shot Binary Detection Task Results Summary Table . . . . .	70
15	Comparison Of Recall Scores Between Zero-shot And 3-shot Binary Detection Prompts . . . . .	71
16	5-shot Binary Detection Task Results . . . . .	73
17	5-shot Binary Detection Task Results Summary Table . . . . .	73
18	Comparison Of Recall Scores Between Zero-shot And 5-shot Binary Detection Prompts . . . . .	74
19	Chain-of-Thought Binary Detection Task Results . . . . .	75
20	Chain-of-Thought Binary Detection Task Results Summary Table . . . . .	75
21	Comparison Of Recall Scores Between Zero-shot And CoT Binary Detection prompts . . . . .	76
22	Chain-of-Thought + 5-shot Binary Detection Task Results . . . . .	77
23	Chain-of-Thought + 5-shot Binary Detection Task Results Summary Table . . . . .	77
24	Comparison Of Recall Scores Between Zero-shot And CoT + 5-shot Binary Detection Prompts . . . . .	78
25	Comparison of Prompts Precision, Recall and F1 Scores in Binary Detection Task . . . . .	79
26	Comparison Of Recall Metrics Across Different Prompts . . . . .	80

27	Zero-shot Classification Results Table . . . . .	82
28	Zero-shot Recall, Precision, F1-Score Per Class . . . . .	83
29	One-shot Classification Results Table . . . . .	85
30	One-shot Recall, Precision, F1-Score Per Class . . . . .	86
31	Zero-shot vs One-shot Macro and Micro Metrics . . . . .	87
32	3-shot Classification Results Table . . . . .	88
33	3-shot Recall, Precision, F1-Score Per Class . . . . .	89
34	Zero-shot vs 3-shot Macro and Micro Metrics . . . . .	89
35	5-shot Classification Results Table . . . . .	90
36	5-shot Recall, Precision, F1-Score Per Class . . . . .	91
37	Zero-shot vs 5-shot Macro and Micro Metrics . . . . .	91
38	Chain-of-Thought Classification Results Table . . . . .	92
39	CoT Recall, Precision, F1-Score Per Class . . . . .	93
40	Zero-shot vs CoT Macro and Micro Metrics . . . . .	93
41	Chain-of-Thought + 5-shot Classification Results Table . . . . .	94
42	CoT + 5-shot Recall, Precision, F1-Score Per Class . . . . .	95
43	Zero-shot vs CoT Macro and Micro Metrics . . . . .	95
44	Comparison of Prompts Micro and Macro Metrics in Classification Task .	96
45	Comparison Of Per-class Recall in Classification Task . . . . .	97

# 1. Introduction

As technology continues to advance rapidly, cyber attackers are evolving alongside it, employing increasingly sophisticated techniques to evade detection. One notable trend is the rise of "Living off the Land" (LOTL) attacks (CrowdStrike, 2024), which leverage legitimate tools and processes already present in a target environment. These attacks are particularly challenging to detect, as they bypass traditional pattern- or signature-based methods that are commonly used by security solutions. In response, leading Endpoint Detection and Response (EDR) and antivirus vendors have adopted custom-trained machine learning (ML) models to identify malicious activity based on behavioral patterns and sequences of actions (Martínez Torres et al., 2019, Handa et al., 2019). However, training such models is both time-consuming and resource-intensive, creating a significant barrier for widespread adoption.

LOTL attacks typically rely heavily on log-based detection and investigation, which demand considerable manual effort and technical expertise to discern subtle patterns indicative of malicious behavior (Kaminsky, 2024). This research seeks to explore whether LLMs, such as GPT-4o, can effectively detect and classify malicious activities within Windows logs, potentially reducing the need for extensive manual analysis and accelerating the investigation process.

To evaluate the capabilities of publicly available LLMs, a custom dataset was generated using the Atomic Red Team framework, simulating 1,064 attack techniques on a target Windows machine. These logs span various Windows Event log categories, including Sysmon, Security, System, PowerShell, and Application logs. The dataset was analyzed using GPT-4o with different prompting strategies: zero-shot, one-shot, 3-shot, 5-shot, Chain of Thought (CoT), and a combination of CoT with 5-shot prompting. This study offers insights into how LLMs can enhance the detection and classification of LOTL-style attacks. The findings have the potential to reduce technical and resource barriers for organizations, enabling quicker and more efficient responses to increasingly sophisticated cyber threats.

The research was conducted in two stages: the first stage focused on detecting malicious activities, and the second aimed at classifying these activities into their respective MITRE ATT&CK categories.

## 1.1 Research Problem

The evolving landscape of cyber threats necessitates innovative approaches in incident response and digital forensics to effectively mitigate risks and respond to security incidents. The first step in mitigating these risks is properly understanding them, and at this stage, the investigative process plays a crucial role (Cichonski et al., 2012). Investigation serves as the foundation for identifying and analyzing malicious activities, as it helps security professionals uncover the tactics, techniques, and procedures (TTPs) used by attackers.

Traditional methods often struggle with the detection of LOTL style of attacks, as they blend in well with regular system usage (Kaminsky, 2024). In this context, LLMs, known for their potential to process and understand unstructured data, emerge as a promising tool. This research endeavors to probe whether leveraging LLMs can enhance the operational efficiency of incident response and digital forensic processes, especially through the lens of host log analysis. By exploring the capabilities of LLMs in detecting and classifying cyber threats, this study aims to bridge the gap between the overwhelming data volume produced by modern systems and actionable intelligence needed for timely and accurate incident response. Current state-of-the-art research offers minimal coverage of LLM applications in incident response and digital forensics, particularly in the area of on-host threat detection. There is a notable lack of data and studies exploring the potential of LLMs for this specific use case. This thesis aims to fill this gap.

Central to this research one main research question and three sub-research questions were identified:

Main research question [MRQ]: **Can Large Language Models be used during incident response and digital forensic processes?**

The main research question was sub-divided into 3 research questions:

- [RQ1] Can large language models be used for the **detection** of cyberattacks based on forensic Windows logs?
- [RQ2] Can large language models be used for the **classification** of cyberattacks based on forensic Windows logs?
- [RQ3] What are optimal prompting strategies to achieve most accurate results when analyzing windows forensic logs using large language models?

In every incident response process, quick detection of malicious activity is crucial to mitigate the impact of cyberattacks. However, manual investigation of logs is both time-

consuming and requires significant technical expertise. Therefore, the first research question **RQ1** seeks to explore how well LLMs can detect malicious activity within Windows logs. By leveraging the natural language processing capabilities of LLMs, this research aims to evaluate their potential to identify indicators of compromise, indicators of attack and other anomalies in a more efficient manner than traditional methods.

Once malicious activity is detected, the next step is to understand the intent behind the attack, which involves classifying the activity into specific attack categories. This classification process is critical for guiding an appropriate response. Thus, the second research question **RQ2** examines how effectively LLMs can classify malicious activities based on available artifacts within Windows logs. This question focuses on whether LLMs can correctly categorize various attack techniques, following the MITRE ATT&CK framework, to facilitate a deeper understanding of the attacker's actions and potential further directions. Building on the findings from the first two research questions, the third research question **RQ3** aims to identify the most effective prompting strategy for detecting and classifying malicious activities, optimizing the performance of LLMs in both tasks.

## 1.2 Contributions & Novelty

This research offers several contributions to the domains of cybersecurity and digital forensics. First, this research fills a significant yet underexplored gap in the intersection of cybersecurity and LLM technology. Despite the growing interest in LLMs across various fields, their application in log-based anomaly detection and categorization remains largely untapped. This study offers a novel approach, demonstrating the potential of LLMs to enhance incident response by enabling more accurate and efficient identification of malicious activities, thereby advancing the state of the art in log analysis and threat intelligence.

In addition, this research presents a newly created dataset comprising 1064 distinct attack technique logs, generated using the Atomic Red Team framework to simulate attacks against a Windows machine. For each simulated attack, a set of Windows Event logs commonly utilized in incident response and forensic investigations has been produced as "attack artifacts". These artifacts include logs from Sysmon, Security, System, Application, and PowerShell logs. The logs were made available as an open-source resource, thereby serving as a tool for future studies and experimentation. What sets this dataset apart is its novelty and breadth. Unlike the majority of existing host-intrusion datasets, which are either outdated or narrowly focused on specific types of attacks (e.g., DDoS, C2 traffic, or Data Loss Prevention), this dataset offers a comprehensive, up-to-date collection (CAIDA, 2007, Creech, 2014, Catak et al., 2021, Stratosphere, 2015). It contains 1064 unique attack

techniques, covering 13 different MITRE ATT&CK categories, with attack complexities ranging from basic to advanced. This diversity makes the dataset a versatile and valuable asset for experimentation and research,

In addition to the dataset, the script that was used to generate the dataset is also being shared with the community. This adds value by allowing users to customize the dataset creation process. Instead of relying solely on the default Windows 10 setup, users can replace it with their own system snapshot. This flexibility makes it possible to simulate attacks tailored to specific research needs or organizational setups, enabling researchers and practitioners to create scenarios that better reflect their unique environments.

### **1.3 Thesis Structure**

This thesis is organized as follows. Chapter 2 provides background information necessary to contextualize the research. It discusses LLMs, their architecture, and potential applications. It also introduces incident response and digital forensics concepts, along with red teaming and adversary emulation, setting the stage for the methodology.

Chapter 3 reviews related work, focusing on the application of LLMs in cybersecurity. It highlights existing studies and identifies gaps that this thesis seeks to address.

Chapter 4 details the methodology employed in this research. It includes dataset generation, data pre-processing, and the development of experimental prompts. This chapter also explains the selection of LLMs and the evaluation metrics used to assess performance.

Chapter 5 presents the results of the experiments. It discusses findings from both the binary detection of malicious activities and the multiclass classification of cyberattacks. Various prompting strategies, including zero-shot, few-shot, chain-of-thought (CoT), and combined approaches, are evaluated and compared in terms of their performance.

Chapter 6 delves into a discussion of the findings, interpreting the results in the context of the research questions. It examines the challenges and limitations encountered, while proposing potential improvements and directions for future work.

Finally, Chapter 7 concludes the thesis, summarizing the key insights and contributions. It also reflects on the broader implications of the research and its potential impact on the field of incident response and digital forensics.

## **2. Background information**

### **2.1 Large Language Models**

Large Language Models (LLMs) represent a significant advancement within the field of artificial intelligence, particularly in natural language processing (NLP) (Chang et al., 2024). These models are a product of deep learning advancements and are designed to understand, interpret, and generate human language with an impressive level of fluency and accuracy. LLMs, such as OpenAI's GPT and Google's BERT, leverage layers of neural networks and are trained on large datasets, enabling them to capture contextual nuances and semantic structures (OpenAI et al., 2023, Devlin et al., 2018).

The foundational architecture for LLMs, the Transformer model, was introduced by Vaswani et al., 2017 and has since become a cornerstone for subsequent innovations. Unlike its predecessors, the Transformer architecture leverages mechanisms such as self-attention and feed-forward neural networks, allowing it to learn contextual relationships between words in a sentence more efficiently. This stand-alone framework facilitated the development of autoregressive models like GPT (Generative Pre-trained Transformer) series by OpenAI and bidirectional models like BERT (Bidirectional Encoder Representations from Transformers) by Google, each contributing unique capabilities to the field of language modeling (OpenAI et al., 2023, Devlin et al., 2018).

An important feature of LLMs is their capability in few-shot and zero-shot situations, where they can perform specific tasks with minimal task-specific data (Brown et al., 2020). This ability is derived from the pre-training phase that provides the models with a broader understanding of language, complemented by a fine-tuning phase that adapts the models to specific applications or domains. As a result, LLMs have demonstrated versatility across a range of language tasks, from translation and summarization to question answering, analysis and conversational agents (Zhao et al., 2023). In recent years, LLMs have been applied across diverse fields, ranging from automated content creation to personalized recommendation systems. Their ability to integrate and process large amounts of unstructured data makes them appealing for challenging tasks such as those found in incident response and digital forensics. Here, they offer potential advantages in anomaly detection, log analysis, and streamlined report generation, by quickly sifting through logs and communications to identify patterns and alert investigators to potential security incidents.



Despite their remarkable capabilities, LLMs pose several challenges and limitations. They require significant computational resources for both training and deployment, leading to concerns about sustainability and environmental impact. Additionally, issues related to bias and ethical use have garnered attention, as LLMs can inadvertently amplify societal biases present in their training data (Hadi et al., 2023). Another limitation is their tendency to produce "hallucinations," where the model generates outputs that are factually incorrect or nonsensical but appear plausible. This behavior undermines trust in their reliability, particularly in high-stakes domains such as cybersecurity or digital forensics (Huang et al., 2023). Thus, while LLMs stand at the forefront of AI-driven language understanding and generation, their successful integration into more specialized fields will depend on ongoing research and development efforts that address inherent limitations.

## **2.2 Incident Response and Digital Forensics**

Incident response and digital forensics are two integral components of cybersecurity that work together to mitigate the effects of cyber incidents and improve organizational resilience. Incident response refers to the systematic approach organizations take to manage and mitigate the consequences of cybersecurity breaches or attacks (Cichonski et al., 2012). This process contains the identification, containment, eradication, and recovery from incidents to restore normal operations while minimizing damage and reducing recovery time and costs. Effective incident response relies on well-structured frameworks, such as the NIST Special Publication 800-61, "Computer Security Incident Handling Guide," which provides organizations with a blueprint to develop and refine their incident response capabilities (Cichonski et al., 2012). Digital forensics, on the other hand, is the scientific discipline concerned with the identification, recovery, preservation, and analysis of information from digital devices in a method that is legally acceptable. This area of study has seen advancements as digital crime has grown more sophisticated (Garfinkel, 2010). Digital forensics plays a crucial role in understanding the nature of an incident, uncovering the methods used by attackers, and providing evidence that can be used in judicial or administrative proceedings. It involves processes and methodologies to ensure that the integrity and chain-of-custody of digital evidence are maintained, thereby enabling investigators to reconstruct events and derive actionable intelligence (Casey, 2009).

Together, incident response and digital forensics form a framework for cybersecurity resilience and incident management. The synergy between these disciplines ensures efficient remediation of security events and supports improvement of organizational defenses against future threats. The increasing complexity and frequency of cyberattacks in recent years underscore the need for continuous evolution and adaptation in both domains. With advancements in technologies such as machine learning and artificial intelligence, there

emerges an opportunity to incorporate new methodologies, such as LLMs, which offer the potential to augment capabilities in both fields. These models have the potential to enhance incident response and digital forensic efforts by improving the speed, accuracy, and depth of data analysis, supporting improved decision-making during and after cyber incidents.

### **2.3 Red Teaming and Adversary Emulation**

Red teaming and adversary emulation are strategies within the broader landscape of cybersecurity that aim to proactively identify and mitigate potential vulnerabilities within an organization's defenses. Red teaming involves an assessment approach whereby a group of security professionals, known as the red team, is tasked with emulating potential attackers to uncover weaknesses within the existing security posture (NIST, 2015). This process is usually more than a technical exercise; it also includes testing organizational readiness and response mechanisms. The primary goal of red teaming is to provide a holistic evaluation that encompasses both technical vulnerabilities and procedural weaknesses, simulating realistic attack scenarios that a sophisticated adversary might employ (Cranford, 2023).

Adversary emulation, a sub-discipline of red teaming, focuses on recreating the tactics, techniques, and procedures (TTPs) used by specific threat actors. By employing this approach, organizations can gain a deeper understanding of the methods that adversaries may use to target their systems (Picus, 2023). This simulation is grounded in threat intelligence, which provides insights into real-world threat actor behaviors, thus enhancing the authenticity and relevance of the exercise. Adversary emulation allows organizations to test their defenses against specific threats (such as specific threat actors or groups), assess the effectiveness of their detection and response mechanisms, and adjust their security strategies accordingly.

Both red teaming and adversary emulation contribute significantly to an organization's cybersecurity strategy by exposing gaps that may not be apparent through traditional security assessments (such as vulnerability management) or audits. These methodologies help organizations to move from a reactive security stance to a proactive one, identifying and addressing potential threats before they can be exploited by malicious actors. By enhancing an understanding of the organization's response capabilities, these exercises play an important role in refining incident response plans and digital forensic processes.

## 3. Related Work

### 3.1 LLMs In Cybersecurity

Cheshkov et al., 2023 in their *Technical Report: Evaluation of ChatGPT Model for Vulnerability Detection* investigated ChatGPT's competency in identifying vulnerabilities within Java code. Utilizing a real-world dataset, their analysis centered on binary and multi-label classification tasks involving CWE vulnerabilities. The findings indicated that GPT-3-turbo achieved F1 scores ranging from 0.118 to 0.231 for different CWE types, with a higher score of 0.530 for negatives. Despite being trained with extensive datasets, the models demonstrated limited efficacy in testing scenarios, with an overall accuracy of 0.383, a macro average of 0.187, and a weighted average of 0.33. These results underscore the need for further refinement and tuning of LLMs for effective vulnerability detection. A similar study by Purba et al., 2023 titled *Software Vulnerability Detection using Large Language Models* investigated LLMs' ability to recognize security vulnerabilities in executable code, specifically targeting SQL injections and buffer overflow issues. Using datasets such as code gadgets and CVEfixes, the study revealed that while LLMs struggle with high false positive rates, they excel at identifying common patterns associated with vulnerabilities, particularly when fine-tuned.

In the domain of Cyber Threat Intelligence (CTI), Clairoux-Trepanier et al., 2024, explored the application of GPT-3.5-turbo for extracting threat intelligence from cybercrime forums in their work *The Use of Large Language Models (LLM) for Cyber Threat Intelligence (CTI) in Cybercrime Forums*. The model achieved an average accuracy of 0.9623, a precision of 0.9, and recall of 0.882. However, this approach raises privacy concerns, especially when threat intelligence involving sensitive information is shared with external LLMs. Furthermore, the nature of publicly accessible forums like XSS and Exploit.in suggests that high-stakes attackers might not disclose their plans in such forums, thus limiting the study's applicability, yet it unveils a promising direction of AI-assisted threat intelligence.

The work of Bayer et al., 2024 in *CySecBERT: A Domain-Adapted Language Model for the Cybersecurity Domain* introduced a specialized BERT model aimed at better analyzing cybersecurity content. CySecBERT outperformed the baseline BERT in detecting MS Exchange threats with an F1 score of 0.8869. Although promising for enhancing regular document analysis, its effectiveness in real-world threat detection remains uncertain.

Song et al., 2024 presented a noteworthy Log-based threat detection system titled *Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection*. Their multi-agent approach, using datasets like CERT r4.2, achieved high precision and accuracy, with a minimal false positive rate on CERT r4.2 (precision=0.943, FP rate=0.037, accuracy=0.961). Despite the impressive performance, the adapting this approach to individual organizational contexts may demand significant resources, potentially outweighing the benefits compared to traditional methods.

The study by Ferrag et al., 2023, *Revolutionizing Cyber Threat Detection with Large Language Models: A Privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices*, developed SecurityBERT, showcasing over 0.982 accuracy against a vast array of attack types in IoT environments. While outstanding, its focus on IoT diverges in applicability from traditional desktop and server environments, displaying differing activity and attack features.

Exploring LLMs in incident response procedures, Hays and White, 2024 discussed in their work *Employing LLMs for Incident Response Planning and Review*, how embedding LLMs into Incident Response Planning (IRP) can streamline the management of dynamic response strategies. Despite not focusing on direct threat detection, the research highlights valuable potential in efficient IR documentation creation and maintenance.

Lastly, Scanlon et al., 2023 in *ChatGPT for Digital Forensic Investigation: The Good, the Bad, and the Unknown* initiated a preliminary investigation into GPT-4's role in digital forensic investigations. Their exploration of ChatGPT in incident analysis and artifact detection provides a general overview without deep technical insights.

In terms of datasets, there is a wide range of options available; however, the majority tend to be either narrowly concentrated on a single category of attacks among many or are quite outdated. For instance, datasets such as the CAIDA DDoS 2007 include a substantial amount of DDoS-related activities but are significantly outdated and overly specialized in DDoS situations (CAIDA, 2007). Similarly, the Windows and Linux ADFA IDS datasets face the same issue, having been developed in 2013, and thus lacking data that represents more current attacker techniques (Creech, 2014).

Furthermore, datasets that concentrate on malware, such as the *Windows Malware Dataset with PE API Calls*, primarily focus on analyzing malware, particularly through examining Portable Executable (PE) files. While this approach is relevant to examining host-based artifacts, it fails to effectively address Living Off The Land (LOTL) style attacks (Catak et al., 2021).

There are numerous datasets available that focus on malware-related network traffic, such as the one provided by the Stratosphere Laboratory. These datasets primarily emphasize network-level data instead of artifacts found on individual hosts (Stratosphere, 2015).

Log analysis is a broad topic that encompasses the examination and interpretation of logs generated by various systems, applications, and network devices to extract information, identify anomalies, and support troubleshooting or security investigations. The analysis of Windows forensic logs (i.e., Sysmon, Security, Application and other logs in the scope of this paper) is simply a subset of this broader domain. To explore potential synergies between state-of-the-art LLM-based log parsing techniques and threat detection use cases, a comprehensive literature review of LLM applications in the log parsing domain was conducted.

Ma et al., 2024 in their work *LLMParser: An Exploratory Study on Using Large Language Models for Log Parsing* explored the potential of LLMs in log parsing. They introduced LLMParser, a generative LLM-based approach that achieved a parsing accuracy (PA) of 0.96, surpassing state-of-the-art parsers like Drain and Logram. The study highlighted that few-shot approach was more effective than in-context learning, achieving a PA of 0.96 compared to 0.46. The research also found that increasing the size of the training shot did not always improve PA. The authors attribute this to potential sampling issues. Furthermore, the study revealed that more complex LLMs do not necessarily yield better results, as demonstrated by Flan-T5-base's (250 million parameters) comparable performance to LLaMA-7B (7 billion parameters).

In a separate study conducted by Xu et al., 2024, titled *DivLog: Log Parsing with Prompt Enhanced In-Context Learning*, authors present an approach to log parsing by leveraging the in-context learning capabilities of LLMs. The DivLog framework addresses the limitations of existing log parsers by automating the log parsing process without requiring human effort in feature design or hyperparameter tuning. DivLog selects diverse log examples to build a candidate set and uses these examples to guide the LLM in generating log templates. Evaluated on 16 public datasets from LogPAI, DivLog achieved a parsing accuracy of 98.1%, precision template accuracy of 92.1%, and recall template accuracy of 92.9%, outperforming existing methods by significant margins.

In another research by Le and Zhang, 2023, titled *Log Parsing with Prompt-based Few-shot Learning*, authors propose a technique to parse logs using prompt-based few-shot learning. The study addresses the limitations of existing log parsers, which are known to suffer from low accuracy and robustness due to their reliance on statistical features and domain-specific knowledge. Authors propose LogPPT, which leverages a pre-trained

language model (RoBERTa) to capture the semantic information of log messages, enabling it to identify keywords and parameters with minimal labeled data. The method eliminates the need for preprocessing and domain-specific hyperparameter adjustments, achieving over 0.9 average Group Accuracy and Parsing Accuracy on 16 public log datasets.

Lastly, Jiang et al., 2024 in their research *LILAC: Log Parsing using LLMs with Adaptive Parsing Cache* addressed the limitations of existing log parsers by leveraging LLMs for log parsing. The authors argue that traditional syntax-based log parsers rely heavily on crafted rules, which can lead to performance degradation when log data deviate from these rules. Semantic-based log parsers, on the other hand, are constrained by limited labeled log messages, often failing to understand complex log semantics. LILAC introduces a novel framework that utilizes LLMs' pre-trained knowledge to parse log messages without the need for manually designed rules. The framework comprises two main components: the ICL-enhanced parser and the adaptive parsing cache. The ICL-enhanced parser leverages in-context learning to adapt LLMs to diverse log data, while the adaptive parsing cache tackles inconsistent outputs and high LLM cost by refining templates and minimizing redundant LLM queries. Evaluations on large-scale log datasets demonstrated that LILAC outperforms state-of-the-art baselines in accuracy and efficiency, achieving a 66.8% and 69.5% improvement in F1 score for grouping and template accuracy, respectively.

The application of advancements in LLM-based log parsing methods for threat detection tasks requires further exploration and validation. Threat detection based on forensic logs extends beyond accurate log parsing and needs to include the identification of unusual patterns, anomalies, or signatures that point to malicious activities. Modern approaches increasingly leverage machine learning algorithms to improve detection efficiency and handle the huge data volumes generated by today's systems. For example, analyzing historical data from a particular host or user log enables ML algorithms to improve the accuracy of anomaly detection while reducing false positives (Khan et al., 2023). Moreover, ensuring the robustness of LLM-based parsers in handling adversarial inputs, such as obfuscated or deliberately misleading log entries, is crucial for practical deployment. For example, a malicious actor could generate malicious log entries designed to mimic "prompts", thus conducting a *prompt injection attack* through log poisoning. Defensive measures are required for such cases, and a human-in-the-loop approach may be necessary to ensure the accuracy and reliability of threat detection. Scalability and efficiency are additional concerns, particularly in large-scale enterprise environments where log volumes can reach terabytes per day. Issues such as LLM querying overhead (cost, network latency), inaccuracy, and hallucinations further complicate the deployment.

The existing literature illustrates the increasingly significant role that Large Language

Models are beginning to play within the field of cybersecurity, with each study examining different aspects of their application (vulnerability detection, cyber threat intelligence, etc). Although these developments are promising, a recurring theme emerges highlighting the need for continued advancement and contextual adaptation to improve the efficacy and reliability of LLMs in cybersecurity settings. In particular, there remains a gap in the available literature focused on the application of LLMs in forensic tasks or practical incident response activities, beyond documentation purposes. Similarly, although there are numerous datasets related to host intrusion, they are often either too narrowly defined or outdated, relying on attack techniques that are no longer relevant. There is a clear lack of comprehensive and current datasets for host-based intrusions and attacks that cover a wide range of tactics. This thesis aims to fill this gap.

## 4. Method

This section outlines the methodology employed in this thesis to validate the hypothesis concerning the effectiveness of LLMs in digital forensic and incident response processes. It provides a detailed account of the dataset generation process, encompassing both malicious and benign datasets, as well as the procedures for dataset processing, LLM model selection, and prompt creation. Furthermore, it explains the evaluation of the collected results. The methodology is structured to ensure reproducibility and compliance with academic standards.

### 4.1 Dataset Generation

This thesis aims to evaluate the effectiveness of LLMs in processing and analyzing forensic artifact logs gathered during digital forensic investigations and incident response activities. To achieve this, the research framework and dataset require the incorporation of two critical components:

1. **Realistic Attack Logs & Artifacts:** The attack logs and artifacts must encompass a wide range of tactics, techniques, and procedures (TTPs) utilized by adversaries. These logs should be diverse and realistic, mirroring the latest trends in cyberattacks to ensure the research's relevance and applicability. The cyberattacks and targeted environment must accurately represent real-world scenarios.
2. **Proven LLM Model:** The chosen LLM should have a sufficiently large context window to process a meaningful volume of logs and possess adequate reasoning capabilities to facilitate effective analysis within the experimental parameters.

Regarding the second component, the approach is relatively straightforward. Recent studies indicate that GPT-4 and its variants (such as GPT-4o and GPT-4-turbo) represent the state of the art in majority of the benchmarks (Vellum, 2024). These models consistently rank among the top three across various tasks, including text comprehension, which is essential for log analysis.

For the first component, the situation is more complex. As discussed in the "Related Work" section, a review of openly available datasets has identified a lack of publicly available datasets that meet these specific criteria - particularly datasets focused on host-based logs from Windows (or Linux systems) that are forensically rich and reflect current attack



vectors. To fill this gap, a decision was made to generate a custom dataset.

### 4.1.1 Dataset Generation Methodology

In the context of leveraging Large Language Models for log analysis during incident response and digital forensic processes, the generation of a robust and representative dataset is crucial. Considering the current market share statistics of operating systems, Windows 10 was selected as the operating system due to its widespread use in both domestic and corporate settings at the time of the study. At the time of this study (November 2024), the operating system market share for Windows stood at 25.67%, ranking it as the second most widely used OS globally, surpassed only by Android (46.06%) (Statcounter, 2024). This decision was made to ensure the research's wide-ranging relevance and applicability, thereby enhancing the impact of the findings and supporting future research and experimental endeavors. For simulating cyberattacks, the study employed the Atomic Red Team framework, known for its extensive range of attack simulations, user-friendliness, and compatibility with various operating systems (R. Canary, 2024). Further details about the Atomic Red Team Framework can be found in the 4.1.4 Attack Simulation & Log Collection Toolchain section, where its role and methodology in creating the dataset are discussed in depth.

Following an assessment of various cloud infrastructure providers, development frameworks, and virtualization software, a particular methodology was selected based on its scalability, efficiency, and reproducibility:

1. Deploy a Virtual Machine (VM) running Windows 10 as the target
2. Install Sysmon on the VM to collect additional logs and artifacts during and after attack execution
3. Create a backup snapshot of the VM's pre-attack state.
4. For each cyberattack simulation:
  - (a) Use Atomic Red Team to install any prerequisite software on the target
  - (b) Simulate the cyberattack on the target
  - (c) Automatically export Windows Logs in the native EVTX format post-attack
  - (d) Revert the Windows 10 VM to the backup snapshot after each attack simulation to prevent residual artifacts from affecting the dataset's quality

This procedure is repeated for all available cyberattack simulations using the Atomic Red Team framework.

The subsequent section provides a detailed description of the target virtual machine's setup,

automation configuration, and relevant networking details.

## 4.1.2 Controller & Target Machine Configuration

The dataset generation setup was designed to allow for a simulation of realistic cyberattack scenarios, enabling the generation of authentic after-attack host artifact dataset for subsequent analysis. The setup comprised two primary machines, each serving distinct roles within the process:

1. The Controller (also serving as the attacker) machine
2. The Target machine

**Controller machine** acts as a central hub for orchestrating the attack simulation and managing the dataset generation process. It is also tasked with the extraction and aggregation of generated logs from the Target machine. The Controller machine uses Oracle VM Virtual Box VboxManage.exe appliance to control the Target virtual machine (boot up, shutdown, backup restoration) (Oracle, 2024). The Controller machine was equipped with the following specifications:

### ■ Operating System

- Windows 10 OS
  - \* Version: 22H2 (64-bit)
  - \* OS Build: 19045.4894
  - \* Experience Version: Windows Feature Experience Pack 1000.19060.1000.0
  - \* Update Version: KB5043064

### ■ Powershell

- PSVersion: 5.1.19041.4894
- BuildVersion: 10.0.19041.4894
- WSMANStackVersion: 3.0
- PSRemotingProtocolVersion: 2.3
- SerializationVersion: 1.1.0.1

### ■ Virtualization

- **Oracle VM VirtualBox:** Version 7.0
- Virtual Network Interface Card (NIC) Configurations:
  - \* NAT Mode
  - \* Bridged Mode

### ■ Configuration Details

- **WinRM:** Client & service configured to allow PSRemoting.
- **Firewall Rule:** Added to enable connectivity to the Target machine.

- **Windows Defender:** Disabled for uninterrupted operation.
  - \* Version: 7.0.18r162988 (Qt5.15.2)

- **Hardware**

- **RAM:** 16 GB

The Controller machine is tasked with simulating cyberattacks against the Target machine, utilizing the Atomic Red Team framework. This framework is managed via the Invoke-AtomicRedTeam PowerShell module, which streamlines attack simulations within Windows environments (Canary, 2024). Given the increasing trend of Living-Off-The-Land attacks, PowerShell Remoting was deemed an appropriate method for dataset generation.

The **Target Machine** is a Windows 10 virtual machine, emulated by the Controller using Oracle VM VirtualBox. It is configured to both initiate and accept connections with the Controller machine via PSRemoting. The primary purpose of the Target Machine is to be subjected to cyberattacks orchestrated by the Controller, thereby producing realistic post-attack logs. The Controller manages the Target Machine through the VBoxManage.exe application. The main specifications and components of the Target Machine are as follows:

- **Operating System**

- Windows 10 OS
  - \* Version: 22H2 (64-bit)
  - \* OS Build: 19045.2965
  - \* Experience Version: Windows Feature Experience Pack 1000.19041.1000.0
  - \* Update Version: KB5020683

- **Powershell**

- PSVersion: 5.1.1.19041.2973
- BuildVersion: 10.0.19041.2673
- WSMANStackVersion: 3.0
- PSRemotingProtocolVersion: 2.3
- SerializationVersion: 1.1.0.1
- Disabled PS ExecutionPolicy
- Enabled PSRemoting

- **Configuration Details**

- **WinRM:** Client & service configured to allow PSRemoting.
- **Firewall Rule:** Added to allow connectivity from the attacker.
- **Windows Defender:** Disabled.

- **System Monitor (Sysmon)**

- Version: v15.15
- Used for monitoring and generation of after-attack logs.

## ■ Hardware

- **RAM:** 4 GB (4096 MB)
- **Video Memory:** 128 MB
- **Virtual HDD:** 50 GB

To ensure successful connection between the Controller and the Target, two prerequisites are essential: (I) the Controller and Target machines must be located within the same virtual network and subnet, and (II) PowerShell Remoting must be activated on both machines. The following PowerShell commands were executed on both the Controller and Target hosts to meet these requirements, after which a virtual backup of the Target was performed:

```
1 Enable-PSRemoting -SkipNetworkProfileCheck -Force
2 winrm set winrm/config/client/auth '@{Basic="true"}'
3 winrm set winrm/config/service/auth '@{Basic="true"}'
4 winrm set winrm/config/client '@{AllowUnencrypted="true"}'
5 winrm set winrm/config/service '@{AllowUnencrypted="true"}'
6 Set-NetFirewallRule -Name 'WINRM-HTTP-In-TCP' -RemoteAddress Any
7 Set-Item WSMan:\localhost\Client\TrustedHosts -Value '*' -Force
```

The script establishes an overly permissive firewall rule, which is unsuitable for a production environment. However, given that the primary aim of this process is to generate attack logs, the authors of this thesis consider this configuration acceptable within a controlled, experimental setting. This approach allows for generation of attacker activity artifacts without the constraints of strict security policies that could otherwise hinder the generation of relevant events and subsequently - the dataset itself. Furthermore, due to the dynamic nature of cyberattacks and the diverse obfuscation and security solution bypass techniques employed throughout attack cycles, it is crucial to maintain a flexible testing environment capable of accommodating a wide array of attack vectors. This flexibility enables the generation of a more comprehensive and diverse dataset.

If future research necessitates, the dataset can be manipulated directly to simulate a more restricted format of collected logs. This approach would enable the study of LLM threat detection capabilities under more restricted conditions, where log data may be incomplete or constrained by stricter security policies, while still retaining the integrity of the original experimental findings.

### 4.1.3 Network Architecture

In the context of Windows-specific attacks, adversarial activities on Windows hosts and their associated networks generally adhere to one of two primary strategies:

1. Conducting malicious operations directly on the compromised host, such as utilizing PowerShell on the already infiltrated machine, a tactic commonly referred to as "post-exploitation" activity
2. Exploiting a variety of native Windows tools for remote control and management to execute malicious scripts on a target over the network. This involves transferring malicious code across the network and executing it either on the machine or directly inside its memory. This method typically leaves little to no trace of execution on the target machine's disk, aside from any potential on-host logging. Examples of such tools include PSexec, WinRM, WMIC, among others

Both methodologies exploit the inherent functionalities of Windows. Consequently, an effective network configuration for simulating both scenarios (1 and 2) in attack simulation cycles can be relatively simple: a single target host for simulating direct attacks (scenario 1) and one attacker host for simulating remote attacks by connecting to the target host (scenario 2). This configuration facilitates the simulation of both direct, on-host malicious activities and remote attacks using native Windows tools during remote exploitation. It is acknowledged that in scenario 2, a single attacker machine can target multiple machines simultaneously (fan-out approach), and conversely, multiple attacker machines can target a single victim host (fan-in approach). However, for the purposes of this experiment, it is sufficient to restrict the configuration to a single attacker and a single target host. This streamlined setup effectively captures the essential artifacts of adversarial activity without introducing unnecessary complexity. The network architecture employed during the dataset generation process is depicted in the diagram below:

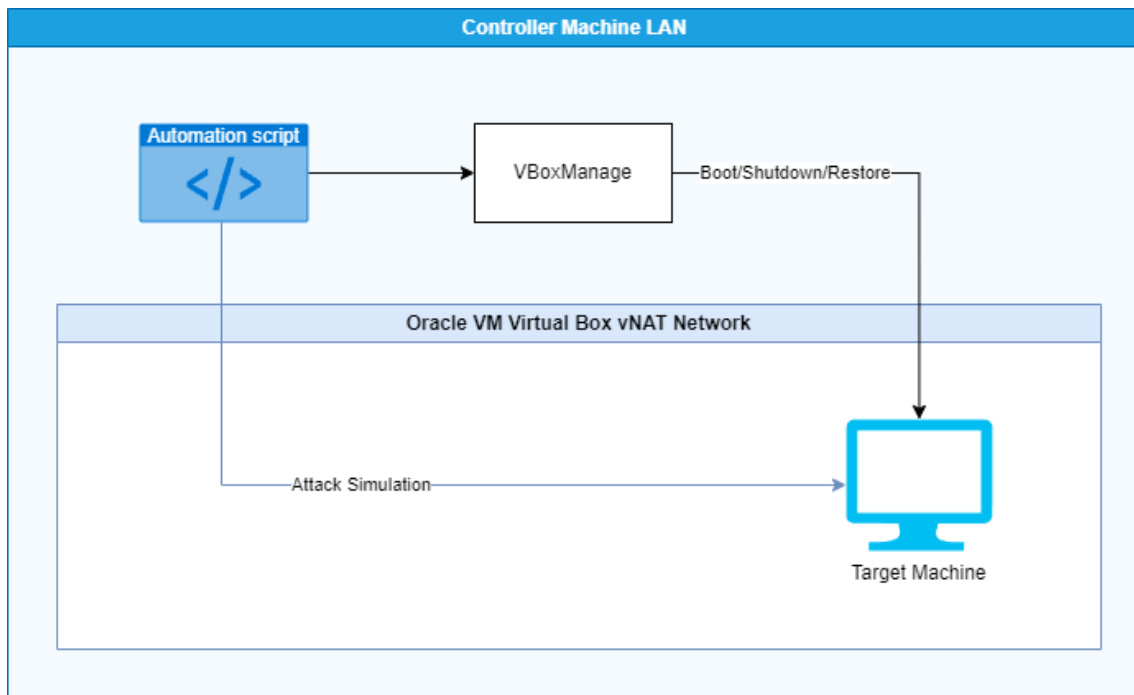


Figure 1. Network Diagram of The Dataset Generation Setup

The diagram above illustrates a virtualized network architecture employed to simulate cyberattacks during the dataset generation process. Below is a detailed description of its components:

1. **Controller Machine LAN:** This machine functions as the central controller for the entire setup, utilizing an automation script to manage and configure the Target virtual machine. It uses VBoxManage command-line tool to manage Oracle VirtualBox Virtual Machines. Additionally, the Controller acts as the adversary, executing simulated cyberattacks against the Target. Communication with the Target machine is facilitated through PSRemoting (PowerShell Remoting), a Windows feature that enables remote command execution (similar to SSH on Linux).
2. **Oracle VM VirtualBox vNAT Network:** This virtualized network environment operates within Oracle VM VirtualBox, providing an isolated network for both the Controller and Target machines.
3. **Target Machine:** This virtual machine serves as the recipient of the attacks. It interacts with the Controller machine via PowerShell remoting, enabling the simulation of the attack scenarios.

A detailed description of the associated hosts, specifically the Controller and Target machines, is provided in 4.2.2 Controller & Target Machine Configuration section. Both machines were configured with a private network connection. This setup was chosen to simulate real-world internal networks and to streamline the dataset generation process,

which relies on PSRemoting.

Future research or more complex scenarios could benefit from expanding the network architecture to include multiple attacker and target hosts. This could facilitate the simulation of multi-host pivoting or computer worms, thereby generating a more comprehensive dataset. Moreover, experiments involving Large Language Models log analysis capabilities may provide valuable insights into their application for analyzing entire networks at a time, rather than just individual hosts during digital forensics and incident response processes. However, this thesis concentrates on a single attacker-host and single target-host configuration to maintain clarity in analyzing direct and remote attack methods within a controlled environment.

#### **4.1.4 Attack Simulation & Log Collection Toolchain**

This section describes the configuration and operation of the attack simulation toolchain, a critical element of this thesis. The toolchain is essential for generating attack artifacts on the Target machine, thereby supporting the dataset creation. The core of the attack simulation process is managed by the Atomic Red Team Framework. Atomic Red Team is a comprehensive library of tests that enables security teams to evaluate security controls by systematically and scalably emulating adversarial behavior (R. Canary, 2024). These tests are precisely defined and formatted for integration into automation frameworks. Within the cybersecurity industry, Atomic Red Team framework serves several purposes, including validating organizational host visibility, assessing detection coverage, and simulating adversarial actions. Within the scope of this thesis, Atomic Red Team is employed to replicate adversarial behaviors on the Target host, producing forensic post-attack artifacts and generating a dataset of event logs that mirror realistic attack scenarios.

Atomic Red team has readily-available adversary emulation scripts for Windows, Linux & Mac. The emulation script library for Windows is most extensive, and contains a total of 1384 different scripts that emulate various MITRE ATT&CK mapped adversarial activities (R. Canary, 2024). Adversarial activities can be emulated either on the host where Atomic Red Team is installed (locally) or remotely against a target machine through the use of Powershell Remoting sessions. The exact quantity of accessible "atomics" (individual attacks available for simulation) varies based on the operating system. For example, certain atomics are exclusively available for emulation on Windows Servers, since they exploit a Windows Server specific vector, while others are applicable to the standard (non-server) versions of Windows. For this dataset generation process, the connectivity between the attacker and the target components was established through Powershell Remoting. Due to this, Atomic Red Team framework was provided with Target

machine login credentials (more accurately, a Powershell session). Additionally, if any of the emulated attacks require prerequisite software (for instance, the TeamViewer backdoor emulation requires TeamViewer to be installed on the target machine), Atomic Red Team can handle prerequisite installation automatically.

The fundamental logical process of attack simulation can be divided into the following steps:

1. On the Controller machine, within the Orchestration component, sequentially select an Atomic Red Team TTP ID (e.g., TTP1124) from the list of available "atomics"
2. On the Controller machine, within the Attacker component, establish a PowerShell Remoting session with the Target machine
3. On the Controller machine, within the Attacker component, conduct a pre-requisite check for the selected "atomic" on the Target via the established PowerShell session
4. (Conditional) If prerequisites are missing, install them on the Target
5. On the Controller machine, within the Attacker component, execute the Atomic Red Team adversary emulation script against the Target
6. Return to step 1 and sequentially select the next atomic

The aforementioned attack simulation steps are cyclically repeated on the target machine. To finalize the dataset generation process, it is necessary to incorporate steps that facilitate the extraction of the generated attack artifacts and logs from the Target machine back to the Controller machine. Additionally, the Target needs to be reverted to the pre-attack state after each simulated attack. The final dataset generation process, which includes log export and backup restoration, is as follows:

1. On the Controller machine, within the Orchestration component, sequentially select an Atomic Red Team TTP ID (e.g., TTP1124) from the list of available "atomics"
2. On the Controller machine, within the Attacker component, establish a PowerShell Remoting session with the Target machine
3. On the Controller machine, within the Attacker component, conduct a pre-requisite check for the selected Atomic Red Team "atomic" on the Target via the established PowerShell session
4. (Conditional) If prerequisites are missing, install them
5. On the Controller machine, within the Attacker component, execute the Atomic Red Team adversary emulation script against the Target
6. On the Controller Machine, within the Orchestration component, extract logs from the Target machine and store them locally on the Controller machine
7. On the Controller Machine, within the Orchestration component, revert the Target



machine back to the pre-attack state

8. Return to step 1 and sequentially select the next atomic

In the process described above, the process of log and dataset generation can be horizontally scaled. This means that we can have several hosts at the same time, on which attack simulations take place.

To preserve the integrity of the test environment and prevent interference between logs and artifacts from different attack simulations, it was deemed essential to implement a mechanism for segmenting logs after each attack cycle. This approach ensures that each attack log snippet corresponding to a simulated attack method is isolated and accurately represents the specific Tactics, Techniques, and Procedures (TTPs) executed during that cycle. The use of Windows OS native timestamping and metadata tagging enhances the traceability and usability of the collected data for subsequent analysis. The decision was made to automate the backup restoration process, where after each simulated attack and log export, the target machine is reverted to its pre-attack state. This ensures that logs remain distinct and do not cross-contaminate the TTPs of different simulated attacks.

Windows includes several log sources that are enabled by default. As this paper aims to assess the effectiveness of LLMs in Incident Response and Digital Forensic processes, it is essential that the log sources generated correspond with those commonly employed in Digital Forensics and Incident Response (DFIR) activities. Consequently, the selection focused on the following:

- **Application Log:** This log captures events associated with software installed on a system, offering insights into application errors, misconfigurations, and potential malicious activities within application operations. In a forensic context, analyzing application logs can help identify the root cause of software malfunctions or detect anomalies indicative of a security breach. For instance, in a real-world scenario, a sudden spike in application errors could signal a targeted attack exploiting a software vulnerability (Microsoft, 2022).
- **System Log:** This log documents events related to operating system components, serving as a resource for identifying system errors, failures, and other system-level activities that may indicate operational issues or security threats. Forensic investigators typically use system logs to trace the sequence of events leading to a system crash or to uncover evidence of tampering with system files. In practice, a forensic analyst might examine system logs to determine if a system failure was caused by a hardware malfunction or a deliberate act of sabotage (Microsoft, 2022).
- **Security Log:** This log records security-related events as defined by the system's audit policy, including successful and failed authentication attempts, policy changes,

and other incidents that may suggest unauthorized access or malicious activities. In forensic investigations, security logs are used for reconstructing the timeline of an intrusion and identifying compromised accounts. For example, a series of failed login attempts followed by a successful one could indicate a brute-force attack that warrants further investigation (Bridge, 2024).

- **Windows PowerShell Log:** This log captures all PowerShell activities, such as command execution and script usage, which are frequently employed in Living-Off-The-Land attacks, advanced persistent threats, and other malicious activities. Forensic analysts can analyze PowerShell logs to detect suspicious command sequences or scripts that may have been used to execute unauthorized actions. In a real-world scenario, identifying unusual PowerShell activity could reveal an attacker leveraging built-in tools to maintain persistence or exfiltrate data (Microsoft, 2024).

To enrich the dataset and simulate a realistic corporate or organizational environment, the target machine was equipped with the Sysmon monitoring solution (markruss, 2024). This addition serves two primary purposes: it reflects the common practice of employing supplementary monitoring tools beyond native logs, and it offers additional context and enrichment for analysis and research.

The Windows operating system utilizes the XML Event Log (EVTX) format for its native log files (Metz, 2024). Following the attack simulation and log extraction methodologies outlined earlier in this document, EVTX logs from various sources (Application, System, Security, PowerShell, and Sysmon) were collected from the Target machine after each simulated attack. Subsequently, these logs were securely transferred to and stored on the Controller machine. The overall log dataset generation process is described on the diagram below:

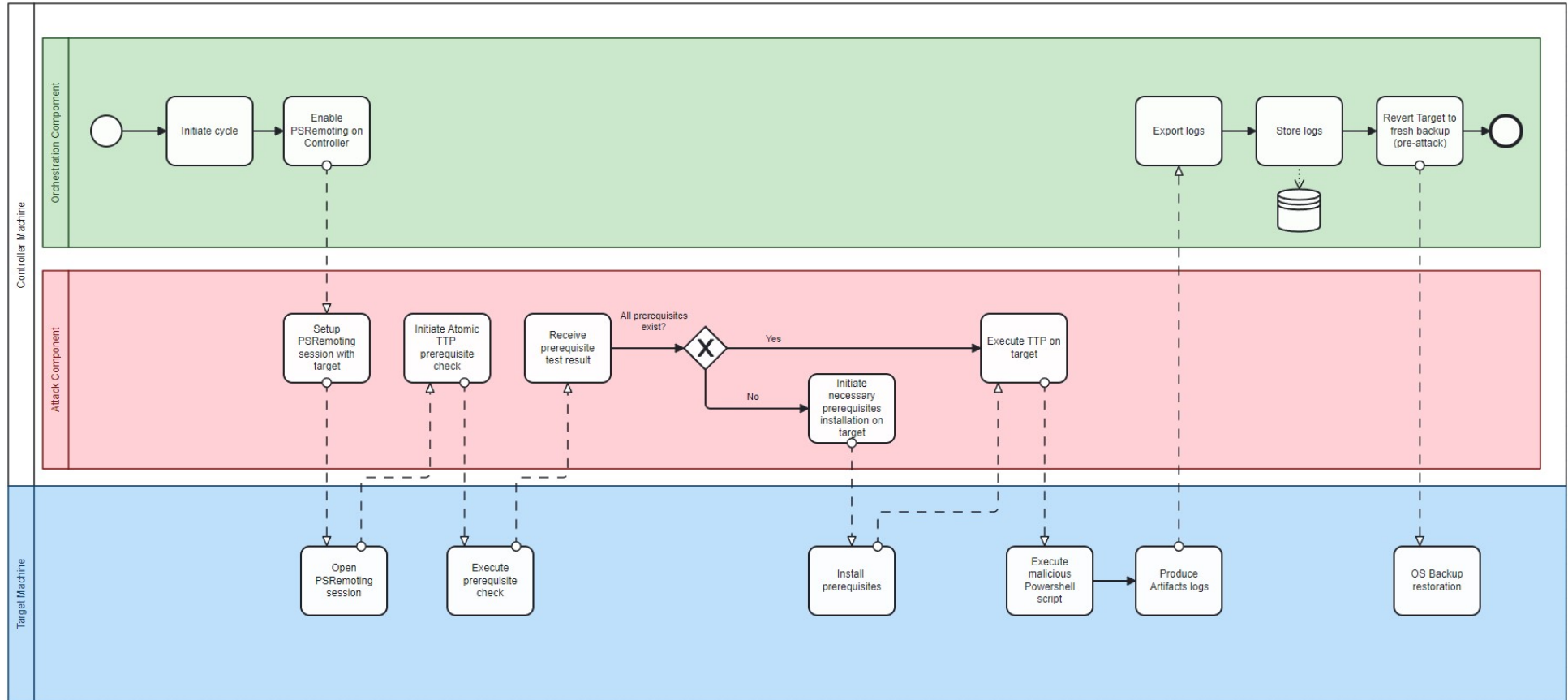


Figure 2. BPMN diagram of the dataset generation process

Following each attack simulation, the target virtual machine is restored to its pre-attack state, ensuring the isolation of logs from each simulated attack. This method produced a clean dataset, without log cross-contamination between different attack activities.

In practical scenarios, security analysts often face the challenge of sifting through voluminous log sets to isolate forensically relevant events. To mimic a scenario where logs from a malicious attack are intermingled with regular Windows activities, the orchestration intentionally allowed the Windows system to operate for an additional 30 seconds after the attack before extracting the logs. This interval allows for the generation of routine Windows logs, thereby creating a more realistic environment for simulation purposes and consequently, a dataset that more accurately mirrors real-world conditions.

Following the simulation of all available attacks, a dataset comprising 1108 distinct attack logs with attack artifacts was produced. Each attack was logged by previously defined Application, System, Security, Powershell and Sysmon logs. These logs varied in complexity and length, encompassing between 30 and 250 discrete events each. However, it is important to note that not every executed "atomic" action yielded successful outcomes in the attack simulation execution. This inconsistency can be attributed to the idiosyncrasies of the Windows operating system, which, through its updates and background processes, occasionally disrupts the stability of certain simulation scripts.

The Atomic Red Team Framework has execution monitoring capabilities, where successful attack simulations return an exit code of "0". A regex extraction search conducted on the execution logs revealed that 258 attack simulation scripts did not execute successfully, as indicated by an exit code of "1" for ERROR, or by the absence of an exit code, suggesting incomplete execution. Further analysis, however, disclosed that certain attack simulations did indeed execute successfully but failed to log the completion accurately. This discrepancy was attributed to the attack simulation scripts being outdated or incompatible with the current Windows OS version. Given that the Atomic Red Team Framework is open-source, allowing contributions from the public, the variability in the quality of the attack simulation scripts is an expected outcome. In the end, the actual number of failed attack simulations was 44, bringing the **total number of successfully simulated attack tactics to 1064**.

The distribution of successfully simulated tactic categories within the final dataset is as follows:

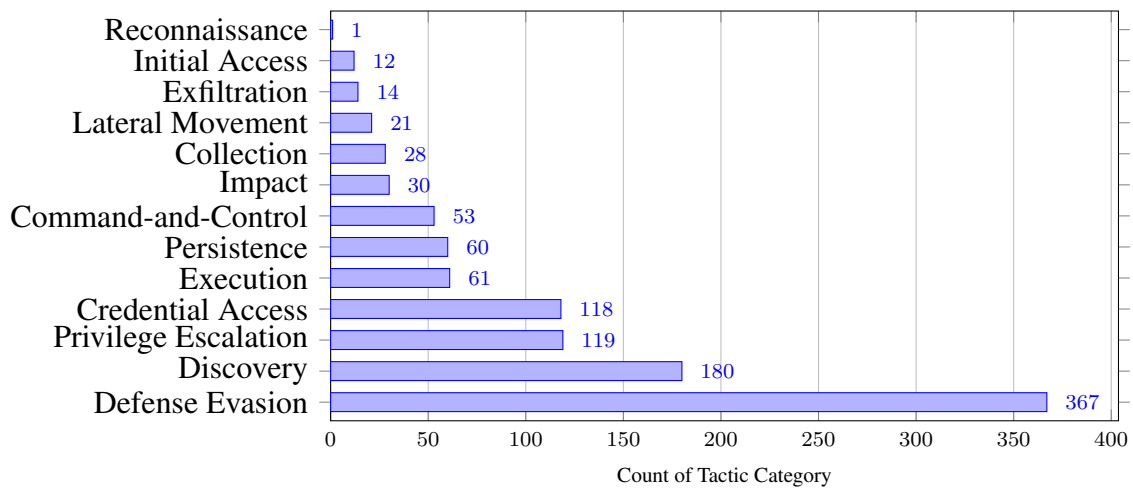


Figure 3. Initial Count of Simulated Tactic Categories

The disparity observed in the distribution of tactic categories, such as 1 reconnaissance tactic compared to 367 defense evasion tactics, can be attributed to the prioritization framework employed by Atomic Red Team, which emphasizes certain atomic tests over others. This discrepancy likely stems from multiple factors, with the most prominent being the variability and complexity associated with simulating certain types of attacks in a consistent and reproducible manner. Additionally, other contributing factors might include the perceived importance of specific tactics in real-world attack scenarios, the ease of implementation for testing purposes, and the availability of relevant data to support simulation efforts.

For example, defense evasion tactics often encompass a wide range of methods to bypass security controls, such as disabling logging or obfuscating commands, which are commonly encountered in various attack vectors. These tactics are frequently prioritized due to their critical role in real-world attacks and their impact on the detection and mitigation process.

On the other hand, reconnaissance tactics may involve gathering information about the target environment, which can vary significantly between organizations and scenarios. This variability poses challenges in creating standardized and repeatable tests for such tactics, leading to their underrepresentation in simulation frameworks like Atomic Red Team. The author of this thesis conducted an analysis that identified a significant overlap between the "reconnaissance" and "discovery" categories within the dataset. This overlap primarily stemmed from the similarity in tools employed and the shared objective of mapping the environment from within the host or network. As a result, to enhance the efficiency of the research, the decision was made to merge these two minor categories into a unified "Discovery" category. As a result, the final distribution looked as follows:

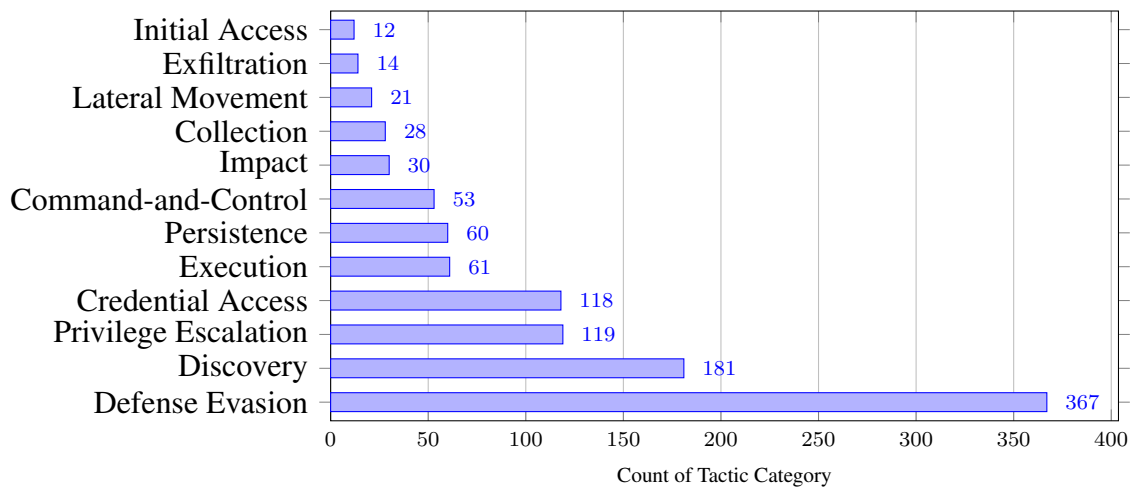


Figure 4. Final Count of Simulated Tactic Categories

In summary, the compiled dataset encompassed a broad spectrum of varied attack tactics. When these tactics were simulated, they produced a diverse collection of data that included contemporary artifacts of attack techniques. This dataset is particularly valuable for analysis utilizing LLMs and other methodologies.

#### 4.1.5 Benign Dataset

After collecting the initial attack logs in EVTX format, the next step involved developing a control dataset composed of benign logs. Two types of logs were selected for this benign dataset: Sysmon and Application logs. To generate a Sysmon dataset reflecting normal behavior, the author of this thesis deployed the Sysmon monitoring tool on a personal computer over a three-month period, from August 2024 to October 2024. Concurrently, the Application log was utilized to gather benign data from the author's computer.

Subsequent to the extraction of both Sysmon and Application logs, the data was converted into JSON format. A deliberate methodological approach was then applied to ensure that both the malicious and benign logs shared a consistent underlying structure and format.

- A total of 1064 simulated attacks were conducted.
- Each attack was logged by five distinct log sources:
  - The cumulative number of log entries across all sources and attacks amounted to 87 916
  - To determine the average number of log entries per attack, the total log entries (87 916) were divided by the number of attacks (1064), resulting in approximately 82.62 log entries per attack
- Benign Sysmon and Application logs spanning three months were segmented into

1064 uniform chunks

- Each chunk contained 83 log entries, designed to mimic the average characteristics of the malicious dataset

As a result, the malicious and benign activity datasets shared the format and the structure of the logs.

## **4.2 Data Pre-processing**

Large Language Models excel in analyzing textual data. Consequently, all logs in EVTX format were converted into JSON, a transformation validated in preliminary tests as being more effective and suitable for LLM analysis due to the proximity of data type descriptors to their corresponding values. Using EvtxECmd, a tool for processing Windows logs created by Eric Zimmerman, the EVTX log files were transformed into JSON format (Zimmerman, 2023).

During the dataset creation and attack simulation process, the Atomic Red Team Framework generated various artifacts within the exported logs. When analyzed by a LLMs, these artifacts provided indications of malicious activity. It was noted that LLMs, including ChatGPT and GPT-4, demonstrated familiarity with the Atomic Red Team framework and were capable of inferring malicious or red teaming activities from logs containing these artifacts.

Do you know what is Atomic Red Team framework?



Yes, the Atomic Red Team framework is an open-source project developed by Red Canary. It provides a collection of tests designed to simulate various tactics, techniques, and procedures (TTPs) used by attackers. The framework allows security teams to evaluate their defenses against known threats in a controlled environment.

#### Key Features:

1. **Modular Tests:** Atomic Red Team uses "atoms," which are small, modular tests that represent specific attack techniques. Each test is easy to execute and can be run individually.
2. **Mapping to MITRE ATT&CK:** The tests are mapped to the MITRE ATT&CK framework, providing a clear reference for understanding which tactics and techniques each test targets.
3. **Documentation and Resources:** Each test comes with documentation that includes detailed descriptions, expected results, and guidance on how to execute them.
4. **Integration with Other Tools:** Atomic Red Team can be integrated with various security tools and platforms, making it versatile for different security environments.
5. **Continuous Improvement:** The community around Atomic Red Team actively contributes to the project, ensuring that the tests evolve alongside emerging threats and attack techniques.

Overall, the framework is valuable for red teams, blue teams, and security researchers looking to enhance their understanding of attack scenarios and improve their security posture through hands-on testing.

Figure 5. ChatGPT Knowing What Is Atomic Red Team Framework

When presented with an unprocessed log snippet containing Atomic Red Team artifacts, ChatGPT successfully inferred that the log likely signified either security testing activities or malicious behavior:



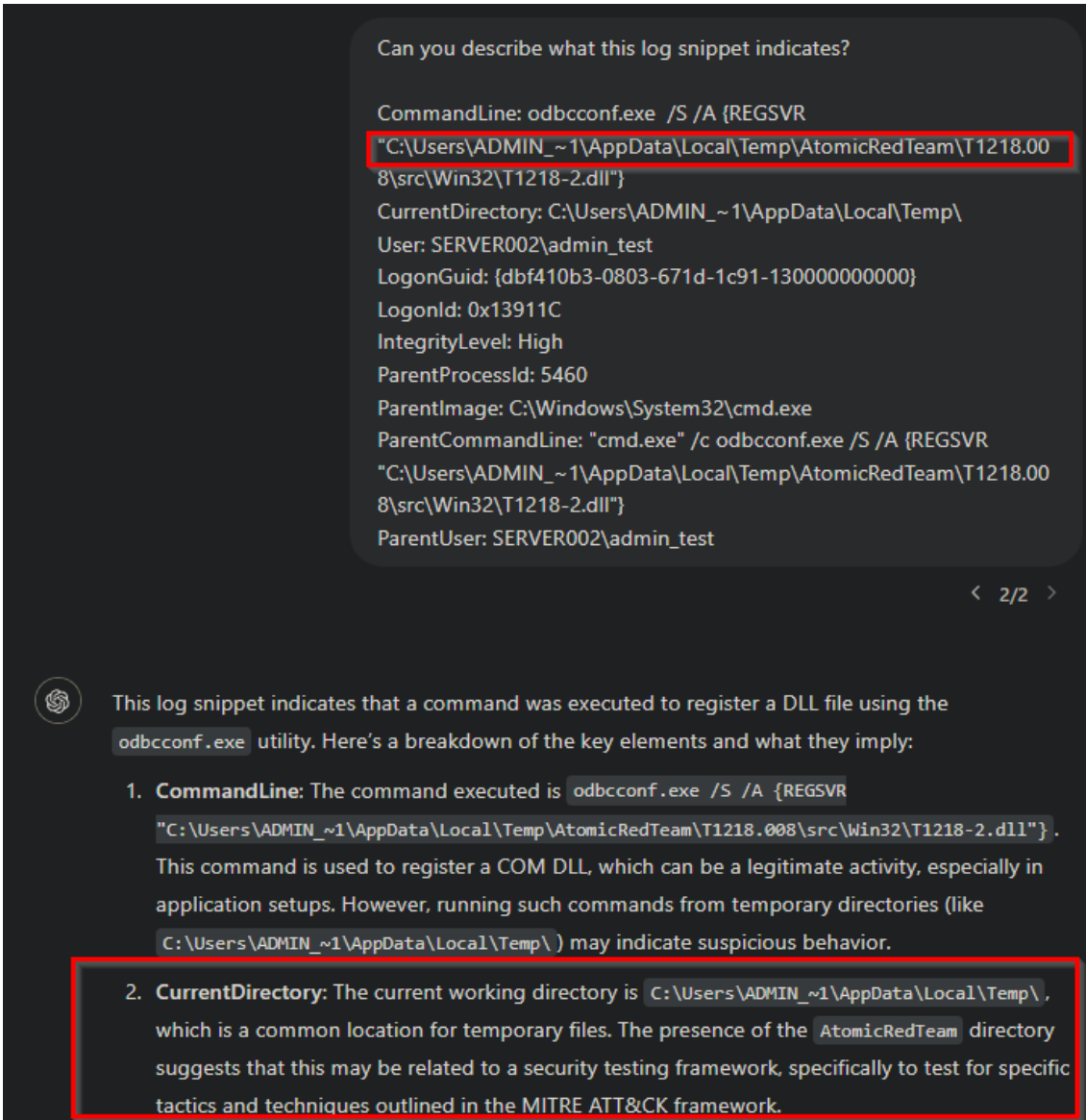


Figure 6. ChatGPT Identifies Atomic Red Team Artifacts, Suggesting The Likelihood Of A Security Test In Progress

To preserve the integrity of the analysis, all keywords and artifacts related to the Atomic Red Team were systematically removed from the dataset. Specifically, a Regex replacement was used to replace the following set of keywords and strings from the entirety of dataset:

1. atomic
2. atomics
3. red
4. team
5. atomic-red-team
6. redcanaryco
7. canary

8. redcanary
9. atomicredteam
10. payload
11. Red team
12. fake
13. wevtutil.exe
  - an artifact of Atomic Red Team Framework initiating connection to the target
14. conhost.exe
  - an artifact of Atomic Red Team Framework initiating connection to the target

Upon further examination of the produced dataset, it was observed that the atomic red team often inadvertently disclosed the tactic ID during simulated attacks. This occurred through filenames or downloaded scripts, thereby providing the LLM with indirect clues about the nature of the activity during log analysis:

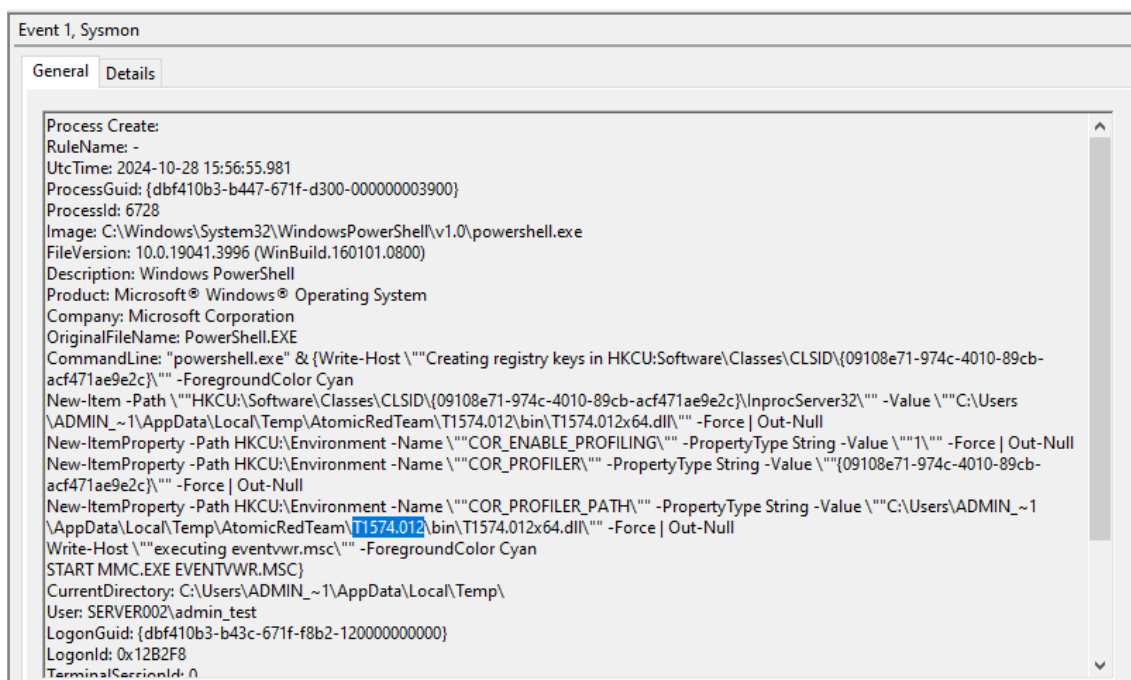


Figure 7. Example of TTP ID Being Leaked Inside A Log Due To Atomic Red Team Execution

To preserve the integrity of the experiment, all 1604 atomic TTP identifiers (e.g., "T1218.008") were substituted with random five-character strings in the text. This measure was necessary because ChatGPT can directly recognize specific TTPs, as demonstrated on the image below:

The third category of artifacts generated from red teaming simulations using the Atomic

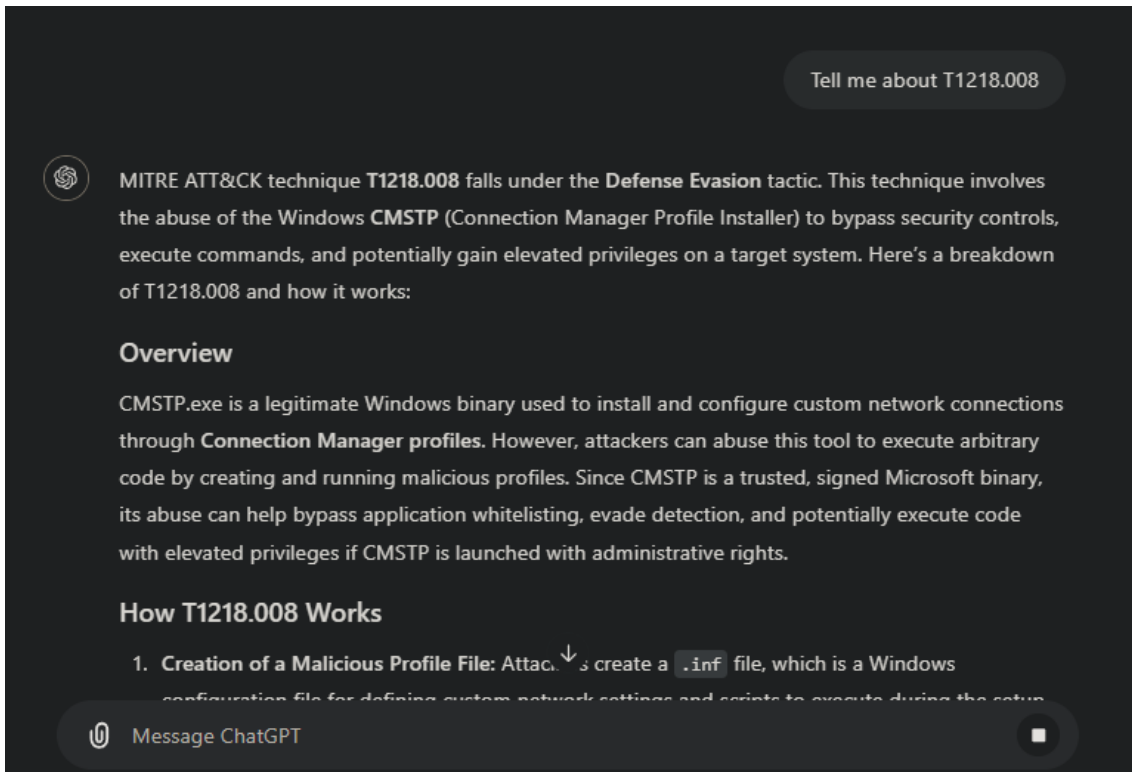


Figure 8. ChatGPT Recognizing MITRE ATT&CK TTP IDs With 0 Additional Context

Red Team framework that required scrutiny was the exposure of attacker tool names. For instance, the execution of the simulated TTP T1082-15 - WinPwnitm4nprivesc resulted in artifacts like "S3cur3Th1sSh1t" and "WinPwn.ps1":

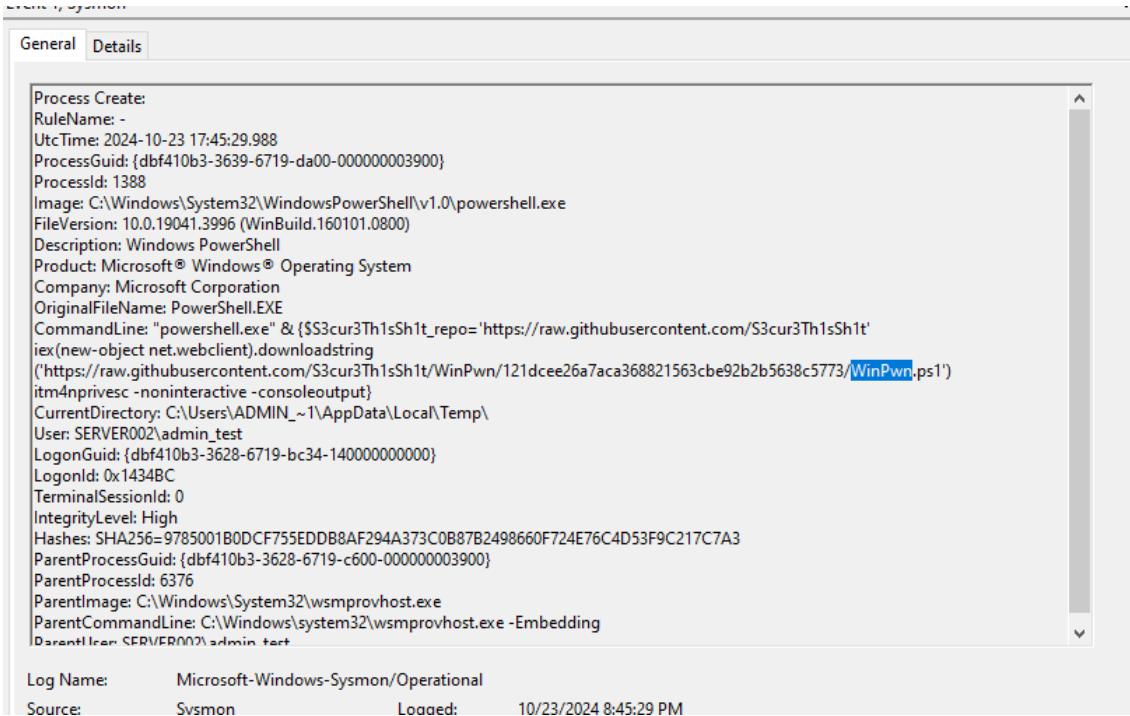


Figure 9. An Example Of Attacker Tooling Artifacts Within The Generated Dataset

These artifacts are inherently recognized by ChatGPT, which introduces a bias in detecting malicious activities. During the manual review of the dataset, a specific set of keywords associated with red teaming tools was identified and subsequently excluded from the dataset:

S3cur3Th1sSh1t	WinPwn	hlldz	Meterpreter
Empire	Cobalt Strike	Mimikatz	PowerSploit
SharpHound	BloodHound	Responder	Invoke-Phantom
PowerSharpPack	PowerSharp	Sharpweb	MzHmO
SharpUp	BC-SECURITY	Get-System	Invoke-WMIExec
Kevin-Robertson	obfuscatedps	dcuac	Invoke-SharpWatson
Rubeus	ExternalPayloads	nanodump	gsecdump
uacme	crackmapexec	phantom	powerview
uacbypass	Psiphon	exfiltration	dnsexfil
Invoke-DNSExfiltrator	Invoke-MalDoc	blackbyte	netwire
ursnif	bad potato	juicy potato	petitpotam
octopus	attack	golden ticket	kerberoast
MalDoc	NPPSpy	ProcDump	pypykatz
Out-Minidump	createdump	dumper	Qakbot
adidnsdump	Get-AdComputer	ADSIsearcher	botnet
exfil	PsExec	Evil-WinRMv	Backstab
SOAPHound	ADRecon	Kerbrute	Snaffler
LaZagne	Firepwd	WebBrowserPassView	SDelete

Table 1. Attacker Tool And Script Names Replaced From The Dataset

Furthermore, log entries associated with pre-attack log clearing on the target machine and post-attack log export artifacts were excluded from the dataset. Specifically, entries containing the following command lines and/or Windows log Event IDs were removed:

Category	Log Clearing Artifacts
Pre-attack log clearing artifacts in Sysmon	"wevtutil.exe" cl Microsoft-Windows-Sysmon/Operational "wevtutil.exe" cl Application "wevtutil.exe" cl System "wevtutil.exe" cl Security "wevtutil.exe" cl "Windows PowerShell"
Pre-attack log clearing artifacts in Security log	EventID 1102 only if it was "first event" in the log
Pre-attack log clearing artifacts in System log	EventID 104 only if it was within first 3 events in the log

Table 2. Pre-Attack Log Clearing Artifacts

As a result of the aforementioned manipulations, a total of 4 types of datasets were produced:

1. A dataset containing artifacts from both the Atomic Red Team framework and associated red team tools (referred as `evtx-unfiltered`)
2. A dataset from which Atomic Red Team framework artifacts have been replaced, but associated red team tool artifacts remain (referred as `evtx-atomic-removed`)
3. A dataset devoid of both Atomic Red Team framework artifacts and associated red team tool artifacts (referred as `evtx-atomic-and-tools-removed`)
4. A benign dataset comprising of author's Sysmon log as a control dataset (referred to as `evtx-benign`)

### 4.3 Large Language Models

This section explains the rationale for selecting the specific LLM and presents the design of the utilized prompts.

According to latest research, smaller LLM models, which are trained on fewer parameters and datasets, exhibit significant constraints relative to larger models (Narang, 2024, Touvron et al., 2023). These constraints become apparent in areas such as language comprehension, generalization capabilities, and performance on specific tasks. Smaller models frequently encounter challenges in nuanced reasoning, maintaining context over extended sequences, and producing outputs that demonstrate human-like coherence and creativity

(Nalpas, 2024). These limitations underscore the necessity of evaluating model selection based on the intended application and task requirements. While smaller models are often favored for their computational efficiency and reduced deployment costs, their restricted capacity makes them less suitable for scenarios demanding high linguistic precision or contextual depth.

In choosing the most suitable LLM, it is important to balance model performance against resource constraints. For tasks that require an in-depth understanding of context and multi-step reasoning, research shows that larger LLMs are generally preferred because of their robustness and reliability (Narang, 2024). Similarly, smaller LLMs are more appropriate for simpler, domain-specific tasks where efficiency and response speed are more important. In the context of this study, where LLMs are employed to analyze Windows after-attack forensic artifacts, a focus on robustness and reasoning capabilities has led to the preference for a larger model.

Furthermore, preliminary validation emphasizes the greater adaptability of larger models, which are more effective at generalizing across a wide range of datasets and unforeseen inputs. Smaller models are less versatile and often require more extensive fine-tuning to achieve similar results, thus limiting their applicability in dynamic or variable scenarios. While such models may be suitable for specific tasks, such as on-premise deployment or fine tuning for detecting specific activities like privilege escalation, this study concentrates on exploring the model's intrinsic, out-of-the-box capabilities.

### **4.3.1 Model Selection**

At the time of composing this paper, one of the most recent models developed by OpenAI is GPT-4o, a variant closely related to GPT-4 and GPT-4-turbo (OpenAI, 2024b). These models are recognized as among the most advanced publicly available language models globally, except for OpenAI's latest model, o1, which is optimized for handling more complex tasks but at a significantly higher financial cost (OpenAI, 2024c). The GPT-4o model has been evaluated against its predecessors like GPT-4 and GPT-4-turbo on standardized benchmarks including MMLU, GPQA, MATH, HumanEval, MGSM, DROP (F1, 3-shot), and SimpleQA (OpenAI, 2024a). It has demonstrated performance that is nearly on par with these models. Given its comparable performance, coupled with significantly reduced costs (one-fourth of the input price and one-third of the output price per 1 million tokens) GPT-4o was selected as the primary tool for this research. This choice was driven by its advantageous pricing-to-performance ratio, which makes it highly applicable for practical and real-world applications.

The table below provides a summary of benchmarks alongside the pricing of the respective models.

Model	Prompt	MMLU	GPQA	MATH	HumanEval	MGSM	DROP (F1, 3-shot)	SimpleQA	Input Cost	Output Cost
<b>o1</b>				MATH-500 <sup>[6]</sup>						
o1-preview	n/a <sup>[7]</sup>	90.8	73.3	85.5	<b>92.4</b>	90.8	74.8	<b>42.4</b>	15.00/1M	60.00/1M
o1-mini	n/a	85.2	60.0	90.0	<b>92.4</b>	89.9	83.9	7.6	3.00/1M	12.00/1M
o1 (work in progress)	n/a	<b>92.3</b>	<b>77.3</b>	<b>94.8</b>	n/a	n/a	n/a	n/a	n/a	n/a
<b>GPT-4o</b>										
gpt-4o-2024-11-20	assistant	85.7	46.0	68.5	90.2	90.3	81.5	38.8	2.50/1M	10.00/1M
gpt-4o-2024-08-06	assistant <sup>[2]</sup>	88.7	53.1	75.9	90.2	90.0	79.8	40.1	2.50/1M	10.00/1M
gpt-4o-2024-05-13	assistant	87.2	49.9	76.6	91.0	89.9	83.7	39.0	5.00/1M	15.00/1M
gpt-4o-mini-2024-07-18	assistant	82.0	40.2	70.2	87.2	87.0	79.7	9.5	0.150/1M	0.600/1M
<b>GPT-4 Turbo and GPT-4</b>										
gpt-4-turbo-2024-04-09	assistant	86.7	49.3	73.4	88.2	89.6	86.0	24.2	10.00/1M	30.00/1M
gpt-4-0125-preview	assistant	85.4	41.4	64.5	86.6	85.1	81.5	n/a	10.00/1M	30.00/1M
gpt-4-1106-preview	assistant	84.7	42.5	64.3	83.7	87.1	83.2	n/a	10.00/1M	30.00/1M

Table 3. Performance Comparison Of LLM Models Across Benchmarks MMLU, GPQA, MATH, HumanEval, MGSM, DROP (F1, 3-shot), SimpleQA, And Associated Costs For Input And Output Tokens (OpenAI, 2024a)

### 4.3.2 Foundational Prompts

As of the writing of this thesis, significant research is focused on finding the most optimized and accurate prompting methodologies for large language models. Various techniques, such as few-shot prompting, Chain of Thought (CoT), self-refinement, self-criticism, and thought generation prompts, are being explored (Schulhoff et al., 2024). There is a general agreement that the effectiveness of a prompting approach is often dependent on the specific use case. However, few-shot and Chain of Thought methodologies are frequently considered the most efficient and accurate for tasks requiring classification and detection (Schulhoff et al., 2024, White et al., 2023). This can be attributed to the structured nature of these methodologies, which enhances the model’s ability to reason through complex tasks and make precise predictions.

#### Binary Detection Task

In this task, the objective is for the large language model to effectively differentiate malicious activities from benign ones, ensuring accurate distinction between the two categories. Prior to using more advanced prompting techniques for the binary detection of malicious activities in Windows logs, it is essential to establish a baseline detection result. This initial benchmark serves as a reference point, enabling subsequent comparisons with the outcomes derived from more sophisticated prompting strategies, such as few-shot and chain of thought.

The foundational prompt, also referred to as a zero-shot prompt, is used to establish

a baseline. Additionally, the same prompt is utilized in few-shot prompting, where supplementary examples of malicious logs are used to improve detection accuracy. In the chain of thought prompting approach, the foundational prompt is further developed by incorporating a detailed step-by-step reasoning process, which aids in clarifying and improving the model's analytical capacity in forensic investigations.

The foundational (zero-shot) prompt for **binary detection** is as follows:

#### Foundational (Zero-Shot) Binary Detection Task Prompt

As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs. You were given these logs from a Windows machine. Your goal is to determine whether there are any indicators of malicious activity within these logs. Please carefully analyze the following log entries, considering any suspicious behavior such as unusual process execution, command-line arguments, privilege escalation, or any other signs of potential compromise.

Provide a detailed analysis of the logs, explaining:

- Any suspicious or anomalous behavior that stands out in the logs.
- Whether any events appear to indicate malicious activity or compromise.

Your response should include a clear assessment of the situation, any threats or risks identified, and guidance on how to proceed. Finalize your finding into ONE of the two resolutions:

- **BENIGN** - if logs contain benign activity
- **MALICIOUS** - if logs contain activity that you are 100% sure is malicious

Below are the logs for your investigation:

[INSERT LOGS HERE]

To optimize the effectiveness of the zero-shot prompt, it was constructed using several established prompting techniques, supported by current research, to enhance the outputs of LLMs. Since LLMs operate under the "garbage in, garbage out" principle, it is crucial to ensure the initial prompt is of high quality. The following strategies and best practices were used when developing the foundational prompt:

### 1. Role-Based Prompting

**Technique:** Assigning a specific role to the model.

**Prompt Section:** *"As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs."*

Establishing the model's role as a Security Analyst helps orient its focus towards relevant expertise.



## 2. Task Definition

**Technique:** Defining the task goal.

**Prompt Section:** *"Your goal is to determine whether there are any indicators of malicious activity within these logs."*

Defining a clear objective, such as identifying malicious activity within logs, enhances the model's understanding of the expected outcome.

## 3. Focused Areas of Concern

**Technique:** Highlighting areas to focus on.

**Prompt Section:** *"Please carefully analyze the following log entries, considering any suspicious behavior such as unusual process execution, command-line arguments, privilege escalation..."*

Directing the model to concentrate on specific indicators of compromise within the logs ensures targeted analysis.

## 4. Structured Output Request

**Technique:** Requesting a structured response.

**Prompt Section:** *"Provide a detailed analysis of the logs, explaining: - Any suspicious or anomalous behavior..."*

Employing a structured approach guarantees that the model's output is both comprehensive and systematically organized.

## 5. Decision Framework

**Technique:** Clear decision-making instructions.

**Prompt Section:** *"Finalize your finding into ONE of the two resolutions: BENIGN, MALICIOUS."*

Providing explicit categories for classification aids the model in accurately categorizing its findings.

## 6. Output Clarification

**Technique:** Clear evaluation criteria.

**Prompt Section:** *"if logs contain benign activity"*

Clarifying what constitutes benign, or malicious activity ensures the model's accurate interpretation of data.

## 7. Encouraging Detailed Response

**Technique:** Requesting a detailed analysis.

**Prompt Section:** *"Provide a detailed analysis of the logs..."*

Encouraging a thorough examination fosters the model's diligence, reducing the likelihood

of missing critical information.

**Multiclass Classification Task** For the multiclass classification task, the foundational prompt from the detection task was used and slightly altered to fit the classification task better:

**Foundational (Zero-Shot) Multiclass Classification Task Prompt**

As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs. You were given these logs from a Windows machine. Your goal is to determine whether there are any indicators of malicious activity within these logs. Please carefully analyze the following log entries, considering any suspicious behavior such as unusual process execution, command-line arguments, privilege escalation, or any other signs of potential compromise.

Provide a detailed analysis of the logs, explaining:

- Any suspicious or anomalous behavior that stands out in the logs.
- Whether any events appear to indicate malicious activity or compromise.

Your response should include a clear assessment of the situation, any threats or risks identified, and guidance on how to proceed. Finalize your finding into ONE of the categories below:

- Collection
- Lateral Movement
- Command and Control
- Discovery
- Execution
- Defense Evasion
- Impact
- Exfiltration
- Initial Access
- Persistence
- Credential Access
- Privilege Escalation
- Benign

Below are the logs for your investigation:

[INSERT LOGS HERE]

After completing the development of the foundational prompts for binary detection and multiclass classification tasks, the analysis of the existing dataset was conducted using the GPT-4o model. A detailed discussion of the specific findings is presented in the Results

section of this thesis.

### 4.3.3 Few-shot Prompts

Few-shot prompting demonstrates its effectiveness by providing a limited number of examples (k-shots) to illustrate the desired task or behavior to the model. This technique works through contextual anchoring, where task-specific examples provide the model with contextual cues that guide its output, thereby reducing ambiguity and enhancing accuracy, particularly in tasks requiring classification or prediction. Furthermore, few-shot prompting replicates the structure of data encountered during training, allowing the model to utilize knowledge (provided context via prompt) more effectively. It is adaptable across various tasks, making it both versatile and applicable to general and domain-specific scenarios.

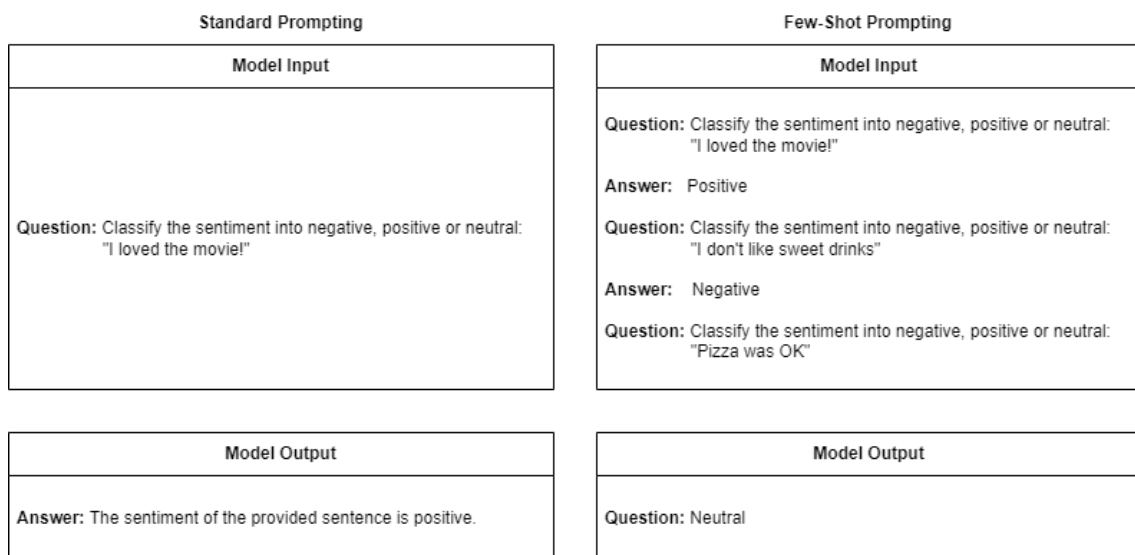


Figure 10. Example Of Few-Shot Prompt

In the context of this thesis, where Windows log analysis is prioritized, there are several methodologies for implementing few-shot prompting. One approach is to provide a specific malicious command, such as a particular PowerShell command, and explicitly label it as malicious. Alternatively, one might provide a whole log snippet containing both benign and malicious entries, simply indicating the presence of malicious content within the snippet. A third strategy involves supplying logs with both benign and malicious components, clearly explaining which parts belong to each category. However, a significant limitation of few-shot prompting arises from the fixed context length in most mainstream large language models available at the time of this thesis. When too many examples are provided, the actual log intended for analysis might not fit within the prompt's input limit, potentially leading to truncated results and negatively impacting the accuracy of analysis outcome.

Therefore, the decision was made to include only malicious strings in few-shot prompting to mitigate this issue and ensure accurate analysis results.

To determine the optimal number of examples, or "k-shots", needed for the few-shot prompt, the initial foundational prompt was supplemented 1, 3, 5 malicious snippets sourced from a previously generated malicious dataset. The selection procedure involved the following steps:

1. Randomly select 5 simulated attack techniques **from each category**
2. Analyze the Sysmon logs for each selected attack technique, extracting relevant malicious command lines
3. Incorporate these extracted malicious strings into the foundational prompt to build the few-shot prompt

If one-shot prompts yield more accurate results than zero-shot, and performance continues to enhance progressively through 3-shot and 5-shot prompts, it is logical to infer that increasing the number of k-shot examples enhances model performance. Consequently, the next logical step would be to test 10-shot prompts.

During the initial phase of experimentation, it was demonstrated that OpenAI's GPT models are capable of identifying both the Atomic Red Team framework and various artifacts related to attacker tools. Consequently, it was decided to source malicious log sections from the `evtx-atomic-and-tools-removed` dataset. This choice ensures the inclusion of more realistic examples, given that in real-world scenarios, (although possible) an adversary is unlikely to use an unaltered version of tools to attack an organization. Typically, executables are renamed and tool configurations modified. For example altering the default Metasploit reverse shell port from 4444 to a less conspicuous option like port 53, commonly associated with DNS traffic.

In the Binary Classification task, the 1,3,5-shot was supplemented with 1, 3, 5 examples of both malicious and benign log snippets.

1. **One-shot:** supplemented with 1 log snippet per class
2. **3-shot:** supplemented with 3 log snippets per class
3. **5-shot:** supplemented with 5 log snippets per class

Overall, the prompt follows the following structure:

### Few-shot Binary Detection Task Prompt

As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs. You were given these logs from a Windows machine. Your goal is to determine whether there are any indicators of malicious activity within these logs. Please carefully analyze the following log entries, considering any suspicious behavior such as unusual process execution, command-line arguments, privilege escalation, or any other signs of potential compromise.

Provide a detailed analysis of the logs, explaining:

- Any suspicious or anomalous behavior that stands out in the logs.
- Whether any events appear to indicate malicious activity or compromise.

Your response should include a clear assessment of the situation, any threats or risks identified, and guidance on how to proceed. Finalize your finding into ONE of the two resolutions:

- **BENIGN** - if logs contain benign activity
- **MALICIOUS** - if logs contain activity that you are 100% sure is malicious

Below are examples provided with example log snippets and resolutions:

Malicious log example:

Malicious Log here

Benign log example:

Benign Log here

Below are the logs for your investigation:

[INSERT LOGS HERE]

For the classification task, the few-shot prompt showcased above is combined with the zero-shot classification prompt.

1. **One-shot:** supplemented with 1 log snippet *from each class*
2. **Three-shot:** supplemented with 3 log snippets *from each class*
3. **5-shot:** supplemented with 5 log snippets *from each class*

### Few-shot Multiclass Classification Task Prompt

As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs. You were given these logs from a Windows machine. Your goal is to determine whether there are any indicators of malicious activity within these logs. Please carefully analyze the following log entries, considering any suspicious behavior such as unusual process execution, command-line arguments, privilege escalation, or any other signs of potential compromise.

Provide a detailed analysis of the logs, explaining:

- Any suspicious or anomalous behavior that stands out in the logs.
- Whether any events appear to indicate malicious activity or compromise.

Your response should include a clear assessment of the situation, any threats or risks identified, and guidance on how to proceed. Finalize your finding into ONE of the categories below:

- Collection
- Lateral Movement
- Command and Control
- Discovery
- Execution
- Defense Evasion
- Impact
- Exfiltration
- Initial Access
- Persistence
- Credential Access
- Privilege Escalation
- Benign

Below are examples provided with example log snippets and resolutions:

EXAMPLE OF {CATEGORY}:

{LOG}

Below are the logs for your investigation:

[INSERT LOGS HERE]

The list of few-shot prompts utilized in this research is available in the Appendices section of this document.

### 4.3.4 Chain of Thought Prompts

Chain of Thought (CoT) prompting guides the model in "thinking aloud" by deconstructing a task into intermediate steps before arriving at a final solution. This method empowers structured reasoning, enabling the model to process tasks incrementally, thus reducing logical errors and mirroring human problem-solving approaches. Research indicates that CoT prompting surpasses standard direct-answer methods, particularly in tasks requiring arithmetic, logical reasoning, or multi-step inference. It also facilitates error identification and correction by relying on intermediate reasoning steps, allowing both users and models to refine errors iteratively.

Standard Prompting	Chain-of-Thought Prompting
<p style="text-align: center;"><b>Model Input</b></p> <p><b>Question:</b> John has 6 candies. He buys 2 candy boxes. Each box has 5 candies. How many candies does he have now?</p> <p><b>Answer:</b> 16</p> <p><b>Question:</b> The bookstore has 50 books. They sold 11 books and bought 10 more. How many books are at the bookstore?</p>	<p style="text-align: center;"><b>Model Input</b></p> <p><b>Question:</b> John has 6 candies. He buys 2 candy boxes. Each box has 5 candies. How many candies does he have now?</p> <p><b>Answer:</b> John started with 6 candies. 2 candy boxes, each with 5 candies is 10. <math>6 + 10 = 16</math>. The answer is 16.</p> <p><b>Question:</b> The bookstore has 50 books. They sold 11 books and bought 10 more. How many books are at the bookstore?</p>
<p style="text-align: center;"><b>Model Output</b></p> <p><b>Answer:</b> 37 ✘</p>	<p style="text-align: center;"><b>Model Output</b></p> <p><b>Answer:</b> The bookstore had 50 books. The book store sold 11 so <math>50 - 11</math> is 39. The bookstore bought 10 more so <math>39 + 10 = 49</math>. The answer is 49 ✔</p>

Figure 11. Example Of A Chain-of-Thought Prompt

In this thesis, the primary focus is on Windows Event Log analysis, aiming for an analytical methodology that mirrors the process of a digital forensic expert or security analyst. Consequently, the Chain-of-Thought approach should embody a structured framework similar to the step-by-step analytical process used by the professionals. This involves segmenting the investigation into distinct, logical phases to optimize thoroughness and mitigate oversight risks. Hence, the structured CoT augmented version of the Foundational Prompt should be as follows:

## Chain-of-Thought Binary Detection Task Prompt

As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs. You were given these logs from a Windows machine. Your goal is to determine whether there are any indicators of malicious activity within these logs. Please carefully analyze the following log entries, considering any suspicious behavior such as unusual process execution, command-line arguments, privilege escalation, or any other signs of potential compromise.

Provide a detailed analysis of the logs, explaining:

- Any suspicious or anomalous behavior that stands out in the logs.
- Whether any events appear to indicate malicious activity or compromise.

Follow the thinking process outlined below:

### **Step 1: Familiarization with the Logs**

Begin by examining the logs to extract essential details such as event IDs, timestamps, user accounts, and process names. Identify any unusual patterns, high-frequency occurrences, or anomalies.

### **Step 2: Examination of High-Risk Event Categories**

Identify event types typical of potential malicious activities:

- Process creation events (Event ID 4688) exhibiting irregular command-line arguments or unfamiliar executables.
- Logon attempts (Event ID 4624, 4625) with unexpected account activity, including failed logons or uses of administrative credentials.
- Events signaling privilege escalation or changes in permissions (Event ID 4672, 4674).
- Modifications to security groups or unauthorized access efforts.

### **Step 3: Contextual Correlation**

Assess suspicious events within the timeline and parallel activities:

- Does the activity correspond with normal usage patterns for this system?
- Are there corroborating logs suggesting a compromise, such as anomalies in network traffic or related security alerts?

### **Step 4: Evaluation of Potential Indicators of Compromise (IoCs)**

Scrutinize processes running from atypical directories (e.g., C:\Users\Username\AppData\Roaming or Temp directories). Search for unexpected system binaries, known exploit signatures, or unusual parent-child process correlations. Identify persistence mechanisms like novel scheduled tasks or registry alterations.

### **Step 5: Threat Categorization**

Classify each observed occurrence into the following categories:

- **Benign:** Routine activity consistent with established behaviors within the environment.
- **Malicious:** Evidence of compromise, such as unusual activity, verified malware execution or explicit indications of privilege abuse.



## Chain-of-Thought Binary Detection Task Prompt (Continued)

### **Step 6: Comprehensive Report Formulation**

Draft a detailed account of observations, including:

- Summaries of any identified suspicious or malicious activities.
- Supporting evidence for each classification.
- Recommended actions, such as containment measures for malicious activities, additional investigation for suspicious instances, or determining that no action is required for benign events.

### **Final Conclusion**

Conclude with one of the following determinations:

- **BENIGN:** No compromise indicators; activities are normal.
- **MALICIOUS:** Identified potential risks ,clear evidence of malicious conduct.

Below are the logs for your investigation:

[INSERT LOGS HERE]

The prompt for classification task was slightly altered, with added description of classification steps.

## Chain-of-Thought Multiclass Classification Task Prompt

As a Senior Security Analyst, you have been tasked with investigating a series of Windows event logs. You were given these logs from a Windows machine. Your goal is to carefully analyze the logs and classify each observed activity into one of the following categories:

- Collection
- Lateral Movement
- Command and Control
- Discovery
- Execution
- Defense Evasion
- Impact
- Exfiltration
- Initial Access
- Persistence
- Credential Access
- Privilege Escalation
- Benign

Provide a detailed analysis of the logs, explaining:

- Any behavior that stands out and aligns with the specified tactics.
- The rationale for categorizing the observed activity into one of the specified categories.

Follow the thinking process outlined below:

### **Step 1: Familiarization with the Logs**

Begin by examining the logs to extract essential details such as event IDs, timestamps, user accounts, and process names. Identify any unusual patterns, high-frequency occurrences, or anomalies.

### **Step 2: Examination of High-Risk Event Categories**

Identify event types typical of potential security activities, focusing on:

- Process creation events (Event ID 4688) exhibiting irregular command-line arguments or unfamiliar executables.
- Logon attempts (Event ID 4624, 4625) with unexpected account activity, including failed logons or uses of administrative credentials.
- Events signaling privilege escalation or changes in permissions (Event ID 4672, 4674).
- Modifications to security groups or unauthorized access efforts.
- Indicators of lateral movement, data exfiltration, or system discovery.

### **Step 3: Contextual Correlation**

Assess suspicious events within the timeline and parallel activities:

- Does the activity correspond with normal usage patterns for this system?
- Are there corroborating logs or patterns suggesting a connection to specific tactics?

## Chain-of-Thought Multiclass Classification Task Prompt (Continued)

### Step 4: Categorization by Tactic

Scrutinize the logs and classify each event based on observed behaviors and the MITRE ATT&CK framework tactics:

- **Collection:** Activities aimed at gathering sensitive data.
- **Lateral Movement:** Events suggesting access to other systems within the network.
- **Command and Control:** Evidence of external communication to manage compromised systems.
- **Discovery:** Efforts to gain knowledge about the system or network environment.
- **Execution:** Processes indicating payload execution or script running.
- **Defense Evasion:** Activities to evade detection, such as disabling security tools.
- **Impact:** Behaviors affecting the availability or integrity of systems.
- **Exfiltration:** Indications of data leaving the network.
- **Initial Access:** The first indications of compromise or access.
- **Persistence:** Mechanisms ensuring continued access.
- **Credential Access:** Attempts to obtain user credentials.
- **Privilege Escalation:** Events indicative of gaining higher-level permissions.
- **Benign:** Routine activity consistent with established behaviors within the environment.

### Step 5: Comprehensive Report Formulation

Draft a detailed account of observations, including:

- Summaries of categorized activities.
- Supporting evidence for each classification.
- Recommended actions, such as containment measures, further investigation, or no action for benign events.

### Step 6: Comprehensive Report Formulation

Draft a detailed account of observations, including:

- Summaries of any identified suspicious or malicious activities.
- Supporting evidence for each classification.
- Recommended actions, such as containment measures for malicious activities, additional investigation for suspicious instances, or determining that no action is required for benign events.

Below are the logs for your investigation:

[INSERT LOGS HERE]

The results from the Chain-of-Thought Prompt being used in the analysis of the `evt-x-atomic-and-tools-removed` dataset are discussed in the Results section of this document.

## 4.4 Evaluation and Performance Metrics

The evaluation framework of this study is structured to assess the performance and dependability of LLM outputs across different prompts. The selected metrics focus on qualities of model performance, such as precision, sensitivity to prompt techniques & styles, and the capacity to sustain consistent performance across varied scenarios. These attributes are vital in the cybersecurity domain, where LLM outputs have the potential to directly influence decisions in incident response and digital forensic investigations.

The table below explains how LLM classifications of "Benign" and "Malicious" correspond to the traditional data analysis metrics of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN):

	<b>Benign detected as</b>	<b>Malicious detected as</b>
<b>Benign</b>	TN	FN
<b>Malicious</b>	FP	TP

Table 4. LLM Analysis Detection Results Classification Logic

The following metrics were selected to evaluate the performance of the LLM:

**Recall** measures the model's ability to identify all relevant instances within a dataset, crucial for ensuring comprehensive detection of potential malicious activity.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

**Precision** augments recall by assessing the accuracy of the LLM's positive predictions. A high precision score suggests that the model's outputs are pertinent and trustworthy, with few false positives. In incident response contexts, false positives might result in unnecessary investigations, diverting resources from genuine threats. Evaluating precision in conjunction with recall offers deeper insight into the model's performance.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

The **F1-score**, which is the *harmonic mean* of precision and recall, provides a balanced measurement that captures both the accuracy and the comprehensiveness of the model's performance in detecting malicious activity. By focusing on F1-scores, the study aims to address the potential trade-offs between precision and recall, ensuring that the large language Models employed are not only identifying malicious activity correctly but are also capturing as many relevant instances as possible. This metric is particularly critical

in the context of incident response and digital forensics, where both false positives and false negatives have consequences. Thus, the F1-score serves as an integral part of the framework to assess the efficacy and reliability of LLMs in these applications.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

To provide a broader context to gathered results, macro-averaged metrics are used since they consider each class equally, shedding light on the model's overall performance across a variety of attack types. Similarly, micro-averaged metrics consolidate results based on individual event contributions, thus emphasizing more frequent classes such as benign logs.

The macro F1 score is calculated by the formula:

$$F1_{\text{macro}} = \frac{2 \cdot \text{Precision}_{\text{macro}} \cdot \text{Recall}_{\text{macro}}}{\text{Precision}_{\text{macro}} + \text{Recall}_{\text{macro}}}$$

Where:

$$\text{Precision}_{\text{macro}} = \frac{1}{N} \sum_{i=1}^N \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

$$\text{Recall}_{\text{macro}} = \frac{1}{N} \sum_{i=1}^N \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

Since the LLM is employed for two distinct tasks: Binary Classification (Malicious vs. Benign) and Multiclass Classification (identifying the specific *attack tactic* being used), the evaluation metrics are tailored to reflect the objectives and outcomes of each task. This differentiation ensures that the assessment captures the performance results of each classification scenario.

Evaluation metrics can be summarized as follows:

### **Binary Classification Task:**

- **Per Individual Class:**

- Recall

- **Per Dataset:**

- Precision
- Recall
- F1

### **Multiclass Classification Task:**

- **Per Individual Class:**

- Precision
- Recall
- F1

- **Per Dataset:**

- Macro Precision
- Macro Recall
- Macro F1
- Micro F1

Evaluating metrics at the class level aids in identifying biases or gaps in the large language model's interpretation of particular event patterns. For example, if the recall for "Command and Control" is low while precision is high, it may suggest that the model is confident in its detection when it does occur but struggles to identify all subtle instances. In contrast, a high recall but low precision for "Execution" may point to an over-detection issue, where benign activities are misclassified as malicious, suggesting a need for improved training data or post-processing adjustments. Therefore, per-class calculations are essential to gain deeper insights into the model's strengths and weaknesses across different classification categories, enabling targeted improvements to enhance performance. Beyond these standard metrics, non-formal qualitative evaluations are conducted to examine the practical relevance of the LLM's outputs during the analysis of Windows Event logs. For instance, during the analysis of LLM outputs, an analysis is performed to assess whether the responses adhere to industry best practices. By integrating quantitative and qualitative analyses, this evaluation framework aims to cultivate a comprehensive understanding of the LLM's capabilities and limitations within the realm of cybersecurity applications.

## 5. Results

### 5.1 Dataset Comparison

As described in the method section, as a result of the dataset generation process, 3 malicious datasets were produced:

1. **evtx-unfiltered** - a dataset containing artifacts from both the Atomic Red Team framework and associated red team tools used by the framework during attack simulation
2. **evtx-atomic-removed** - a dataset from which Atomic Red Team framework artifacts have been replaced, but associated red team tool artifacts remain
3. **evtx-atomic-and-tools-removed** - a dataset devoid of both Atomic Red Team framework artifacts and associated red team tool artifacts

To evaluate the influence of the Atomic Red Team Framework and attack tooling artifacts on the dataset, we used a foundational zero-shot prompt across all three datasets with a detection task. The zero-shot prompt for this task was defined in the 4.3.2 Foundational Prompts section. This approach aimed to quantify the extent to which these artifacts affect the capabilities of large language models in detecting malicious activities within the logs.

First, the model was tasked with analyzing a dataset that had not been cleaned of artifacts originating from the Atomic Red Team Framework and red teaming tools (**evtx-unfiltered**).

Technique Category	Malicious	Benign	Recall
Collection	28	0	1
Lateral Movement	21	0	1
Command and control	53	0	1
Discovery	181	0	1
Execution	61	0	1
Defense Evasion	367	0	1
Impact	30	0	1
Exfiltration	14	0	1
Initial Access	12	0	1
Persistence	60	0	1
Credential Access	118	0	1
Privilege Escalation	119	0	1
<b>Average</b>	<b>1064</b>	<b>0</b>	<b>1</b>

Table 5. Zero-shot Detection Task Results On `evtx-unfiltered`

The findings immediately demonstrated that Atomic Red Team artifacts, alongside artifacts from red team tools, such as the direct names of these tools, led to results that were not realistic. As outlined in the Dataset Generation Method section, GPT models exhibit a relative proficiency in identifying both Atomic Red Team artifacts and elements from red teaming tools. Consequently, a subsequent experiment was conducted with a similar setup, with the key difference being that the Detection Task was performed on a dataset **evtx-atomic-removed** from which Atomic Red Team artifacts had been excluded, while the artifacts of the attacking tools were retained.

Technique Category	Malicious	Benign	Recall
Collection	23	5	0.821428
Lateral Movement	12	9	0.571428
Command and control	39	14	0.735849
Discovery	84	97	0.464088
Execution	52	9	0.852459
Defense Evasion	290	77	0.790190
Impact	25	5	0.833333
Exfiltration	13	1	0.928571
Initial Access	12	0	1
Persistence	55	5	0.916667
Credential Access	104	14	0.881355
Privilege Escalation	107	12	0.899159
<b>Average</b>	<b>816</b>	<b>248</b>	<b>0.766917</b>

Table 6. Zero-shot Detection Task Results On `evtx-atomic-removed`



We now observe outcomes that are more realistic. Though, it appears that the majority of techniques are detected with recall that exceeds expectations for a zero-shot, untrained, and unfinetuned large language model. This heightened performance likely stems from the presence of red teaming tools artifacts (such as direct tool names, specific github repos of known tools etc), which may provide the LLM with significant clues when analyzing the logs. Consequently, our subsequent test focused on a malicious activity detection task using a dataset from which all atomic red team artifacts and red teaming tooling artifacts had been removed. Detailed descriptions of the **evtx-atomic-and-tools-removed** dataset preparation process can be found in the "Dataset Generation" section of this document.

Technique Category	Malicious	Benign	Recall
Collection	19	9	0.678571
Lateral Movement	14	7	0.666667
Command and control	35	18	0.660377
Discovery	81	100	0.447513
Execution	47	14	0.770491
Defense Evasion	267	100	0.72752
Impact	22	8	0.733333
Exfiltration	13	1	0.928571
Initial Access	11	1	0.916667
Persistence	53	7	0.883333
Credential Access	101	17	0.855932
Privilege Escalation	109	10	0.915966
<b>Average</b>	<b>772</b>	<b>292</b>	<b>0.725563</b>

Table 7. Zero-shot Detection Task Results On `evtx-atomic-and-tools-removed`

We can immediately see, that for the purpose of research and experimentation with different prompting techniques, it is best to focus on the **evtx-atomic-and-tools-removed** dataset, as it provides a more balanced and realistic environment for evaluating the capabilities of large language models in detecting malicious activities. By excluding Atomic Red Team artifacts, the dataset minimizes the influence of artificially recognizable patterns, allowing the detection performance to be assessed based on intrinsic behavioral indicators of malicious activities rather than superficial artifacts. Thus, for the remainder of this research, the experiments will be conducted using the **evtx-atomic-and-tools-removed** dataset. Centering the study on this dataset enables an investigation of prompting strategies and the ways in which LLMs interpret context in scenarios that mirror real-world security operations. This approach ensures that the findings are directly applicable and relevant to the cybersecurity field, particularly in leveraging LLMs for practical event log analysis.

## 5.2 Malicious Activity Binary Detection

This section focuses on the results of the LLM being used to analyze the **evtx-atomic-and-tools-removed** dataset and detect malicious activity. Main task is to classify the mixed set of logs containing malicious and benign activity into "malicious" and "benign" categories.

### 5.2.1 Zero-shot

For the zero-shot prompt, the **evtx-atomic-and-tools-removed** dataset was used. The prompt was described in the 4.4.2 Foundational Prompts section of the document. In the context of the Binary Detection task, True Positives (TP) represent cases where malicious instances are correctly classified as malicious, and benign instances are correctly classified as benign. False Negatives (FN) occur when malicious instances are misclassified as benign or vice versa. In the Binary detection task, given the availability of only TP and FN, the sole metric that can be computed on a per-class basis is Recall, which measures the proportion of correctly identified instances within a class. This metric is emphasized throughout the Binary Detection task results because it directly reflects the model's ability to identify each class accurately. In contrast, metrics such as Precision and F1-score, which require the computation of False Positives (FP) in addition to TP and FN, are aggregated and analyzed at the dataset level. These "per-dataset" metrics provide a holistic evaluation of the model's performance by balancing both precision (the ability to avoid false alarms) and recall. This distinction is further elaborated in Section 5.2.5, where the discussion includes Recall, Precision, and F1-score as performance measures for each dataset.

When prompted with a binary detection task, the GPT-4o model produced the following set of results:

<b>Technique Category</b>	<b>Malicious</b>	<b>Benign</b>	<b>Recall per class</b>
Initial Access	11	1	0.916667
Exfiltration	13	1	0.928571
Lateral Movement	14	7	0.666667
Collection	19	9	0.678571
Impact	22	8	0.733333
Command and control	35	18	0.660377
Persistence	53	7	0.883333
Execution	47	14	0.770491
Credential Access	101	17	0.855932
Privilege Escalation	109	10	0.915966
Discovery	81	100	0.447513
Defense Evasion	267	100	0.727527
Benign Activity	4	1060	0.996240

Table 8. Zero-shot Binary Detection Task Results

The findings depicted in Table 11 show the performance of the GPT-4o model in a zero-shot binary detection task. The variability in recall rates across different technique categories shows the LLM's dependency on the specific type of attack technique for distinguishing between malicious and benign activities. Notably, the model excels in identifying "Exfiltration" (recall: 0.928571) and "Initial Access" (recall: 0.916667), indicating a robust capacity for accurately classifying activities associated with these phases of an attack lifecycle. In contrast, the "Discovery" (recall: 0.447513) and "Defense Evasion" (recall: 0.727527) categories demonstrate weaker performance, pointing to challenges in accurately detecting and classifying these techniques due to potentially subtler or more contextually ambiguous activity patterns within event logs. The relatively low recall in categories such as "Lateral Movement" (recall: 0.666667), "Command and Control" (recall: 0.660377), "Collection" (recall: 0.678571), and again "Discovery" suggests that the zero-shot approach might struggle with nuanced attack patterns that depend on behavioral indicators rather than explicit artifacts. These limitations highlight the necessity for more experimenting with more refined prompting strategies (or further model fine-tuning with domain-specific data) to attempt to enhance detection accuracy in real-world applications. Thus, while the results illustrate the promise of large language models for Windows Event Log analysis, they also expose considerable areas for improvement, especially in managing ambiguous or stealthy attack techniques. The zero-shot prompt demonstrates remarkable proficiency in identifying benign activities, achieving a class recall score of 0.9962.

	<b>Malicious (actual)</b>	<b>Benign Activity (actual)</b>
<b>Malicious (predicted)</b>	772	4
<b>Benign Activity (predicted)</b>	292	1060

Table 9. Zero-shot Binary Detection Task Results Summary Table

From the summary table above, we can derive the following:

$$TP = 772 \quad FP = 4 \quad FN = 292 \quad TN = 1060$$

Having the TP, FP, FN and TN, we can then calculate the precision and recall as follows:

$$\text{Precision} = \frac{772}{772 + 4} \approx 0.9948 = 99.48\%$$

$$\text{Recall} = \frac{772}{772 + 292} \approx 0.7255 = 72.55\%$$

$$F1 = \frac{2 \cdot 0.9948 \cdot 0.7255}{0.9948 + 0.7255} \approx 0.8391 \approx 83.91\%$$

Based on the precision and recall, we can see that the system demonstrates a high precision of 99.48%, indicating that nearly all predictions labeled as "malicious" are indeed correct. This highlights its reliability in minimizing false positives (FP = 4). However, the recall of 72.55% reveals that a significant number of malicious cases (FN = 292) were missed, suggesting limitations in identifying all malicious activities. This trade-off between precision and recall suggests the system is optimized for accuracy in positive predictions but at the cost of sensitivity. Such a configuration might be suitable for applications prioritizing low false alarms but less so where capturing all potential threats is critical. The F1 score of 83.91% shows that the system maintains a reasonable balance between precision and recall, but there is room for improvement, especially in increasing recall without significantly compromising precision. The lower recall underscores the need for further refinement in the model's ability to detect all malicious instances.

More detailed discussion and comparison of achieved results is provided in the Discussion subsection

## 5.2.2 Few-Shot

This section discusses the findings of using a few-shot prompts to analyze malicious and benign logs. The primary objective was to distinguish between malicious activities and benign activities within these logs. Initially, the results of a one-shot prompt are presented, followed by those from 3-shot and 5-shot prompts. The results encompass class-specific recall metrics, along with general precision, recall, and F1 scores for the entire data set.

### One-shot Binary Detection

The prompt from the zero-shot binary classification task was enhanced by including examples of both malicious and benign log snippets. For the one-shot prompt, each technique category, or class, was supplemented with a single example snippet. The binary classification results of using the one-shot prompt are presented below.

Technique Category	Malicious	Benign	Recall per class
Initial Access	10	1	0.909090
Exfiltration	11	2	0.846153
Lateral Movement	15	5	0.75
Collection	20	7	0.740740
Impact	19	10	0.655172
Command-and-Control	34	18	0.653846
Persistence	52	7	0.881355
Execution	39	21	0.65
Credential Access	100	17	0.854700
Privilege Escalation	101	17	0.855932
Discovery	84	96	0.466667
Defense Evasion	242	124	0.661202
Benign Activity	2	1061	0.998118

Table 10. One-shot Binary Detection Task Results

	Malicious (actual)	Benign Activity (actual)
Malicious (predicted)	727	2
Benign Activity (predicted)	325	1061

Table 11. One-shot Binary Detection Task Results Summary Table

$$TP = 727 \quad FP = 2 \quad FN = 325 \quad TN = 1061$$

$$\text{Precision} = \frac{727}{727 + 2} \approx 0.9972 = 99.72\%$$

$$\text{Recall} = \frac{727}{727 + 325} \approx 0.6910 = 69.10\%$$

$$\text{F1} = \frac{2 \cdot 0.9972 \cdot 0.6910}{0.9972 + 0.6910} \approx 0.8163 \approx 81.63\%$$

Contrary to the prevalent belief that increasing the number of examples, or "k-shots," invariably enhances outcomes, our findings suggest that one-shot prompting tends to yield marginally lower results. Specifically, the general metrics for zero-shot prompting (Precision=0.9948, Recall=0.7255, F1=0.8391) are slightly superior compared to those achieved with one-shot prompting (Precision=0.9972, Recall=0.6910, F1=0.8163). We can also see that one-shot prompt is slightly better at identifying benign activity with a class recall score of 0.9981 (compared to zero-shot recall score of 0.9962). A further comparison of recall rates by class between zero-shot and one-shot prompts reveals the following:

Class	0-shot	1-shot	Recall Difference
Initial Access	0.9166666667	0.9090909091	-0.007575757576
Exfiltration	0.9285714286	0.8461538462	-0.08241758242
Lateral Movement	0.6666666667	0.75	0.083333333333
Collection	0.6785714286	0.7407407407	0.06216931217
Impact	0.7333333333	0.6551724138	-0.07816091954
Command and control	0.6603773585	0.6538461538	-0.006531204644
Persistence	0.8833333333	0.8813559322	-0.00197740113
Execution	0.7704918033	0.65	-0.1204918033
Credential Access	0.8559322034	0.8547008547	-0.001231348689
Privilege Escalation	0.9159663866	0.8559322034	-0.06003418316
Discovery	0.4475138122	0.4666666667	0.01915285451
Defense Evasion	0.727520436	0.6612021858	-0.06631825017
Benign Activity	0.9962406015	0.9981185325	0.001877930952
<b>Average</b>	<b>0.7831681122</b>	<b>0.7633061876</b>	<b>-0.01986192459</b>

Table 12. Comparison of Recall Scores Between Zero-shot And 1-shot Binary Detection Prompts

We can see that 1-shot prompt performs on average slightly worse than zero-shot. The differences in performance can be attributed to the distinct ways in which the model processes inputs. With zero-shot prompting, the model depends entirely on its pre-trained knowledge and comprehension of context to generate predictions, which might align more closely with the general patterns it has internalized. In contrast, one-shot prompting involves presenting the model with an additional example, which can unintentionally introduce bias or limit its interpretative range, potentially diminishing performance in certain categories.

The variability in performance across different classes may also be influenced by the characteristics of the data and the specific examples utilized in one-shot prompting. For example, if the example in the one-shot prompt poorly corresponds with the patterns of certain classes (such as Exfiltration or Impact), it might misdirect the model's attention, thereby decreasing its recall. On the other hand, in classes where the example closely resembles the test data (such as Lateral Movement or Collection), one-shot prompting could marginally improve recall.

The random selection of k-shots for each class creates a potential bias in LLMs. This is because in our dataset, each class encompasses diverse techniques. For example gaining initial access through Remote Desktop Protocol (RDP) versus creating a Guest account. Limiting the model to a single example may lead it to mistakenly associate initial access more closely with RDP activity than with Guest account creation, despite both methods being commonly employed by adversaries. Therefore, when tasks involve classes with varying internal features, a one-shot prompt might not be the most effective strategy.

### 3-Shot Binary Detection

The prompt from the zero-shot binary classification task was enhanced by including examples of both malicious and benign log snippets. For the 3-shot prompt, each technique category, or class, was supplemented with 3 example snippets. The binary classification results of using the 3-shot prompt are presented below.

Technique Category	Malicious	Benign	Recall per class
Initial Access	8	1	0.888889
Exfiltration	11	0	1
Lateral Movement	14	4	0.777778
Collection	19	6	0.76
Impact	21	6	0.777778
Command-and-Control	34	16	0.68
Persistence	51	6	0.894736
Execution	46	12	0.793103
Credential Access	95	20	0.826086
Privilege Escalation	100	16	0.862068
Discovery	87	91	0.488764
Defense Evasion	245	119	0.673076
Benign Activity	1	1060	0.999057

Table 13. 3-shot Binary Detection Task Results

	Malicious (actual)	Benign Activity (actual)
Malicious (predicted)	731	1
Benign Activity (predicted)	297	1060

Table 14. 3-shot Binary Detection Task Results Summary Table

$$TP = 731 \quad FP = 1 \quad FN = 297 \quad TN = 1060$$

$$\text{Precision} = \frac{731}{731 + 1} \approx 0.9986 = 99.86\%$$

$$\text{Recall} = \frac{731}{731 + 297} \approx 0.7110 = 71.10\%$$

$$F1 = \frac{2 \cdot 0.9986 \cdot 0.7110}{0.9986 + 0.7110} \approx 0.8306 \approx 83.06\%$$



When evaluating the performance metrics for zero-shot prompt (Precision=0.9948, Recall=0.7255, F1=0.8391) in comparison to 3-shot prompt (Precision=0.9986, Recall=0.7110, F1=0.8306), we observe a slight improvement in precision. However, the recall and F1 scores remain inferior to zero-shot prompt. This discrepancy can be attributed in part to the same factors affecting the one-shot prompt. Specifically, incorporating multiple examples per class (k-shots) reduces the diversity available within each class. Given the wide array of individual attack techniques employed across different classes, this narrowing of variability results in diminished effectiveness.

Analyzing per class recall however, we can see that 3-shot prompt improves recall scores across a number of classes:

<b>Class</b>	<b>0-shot</b>	<b>3-shot</b>	<b>Recall Difference</b>
Initial Access	0.9166666667	0.8888888889	-0.0277777778
Exfiltration	0.9285714286	1	0.07142857143
Lateral Movement	0.6666666667	0.7777777778	0.1111111111
Collection	0.6785714286	0.76	0.08142857143
Impact	0.7333333333	0.7777777778	0.0444444444
Command and control	0.6603773585	0.68	0.01962264151
Persistence	0.8833333333	0.8947368421	0.01140350877
Execution	0.7704918033	0.7931034483	0.022611645
Credential Access	0.8559322034	0.8260869565	-0.02984524687
Privilege Escalation	0.9159663866	0.8620689655	-0.05389742104
Discovery	0.4475138122	0.4887640449	0.04125023279
Defense Evasion	0.727520436	0.6730769231	-0.05444351289
Benign Activity	0.9962406015	0.9990574929	0.002816891427
<b>Average</b>	<b>0.7831681122</b>	<b>0.8016414706</b>	<b>0.01847335841</b>

Table 15. Comparison Of Recall Scores Between Zero-shot And 3-shot Binary Detection Prompts

We can observe that the values for Initial Access, Credential Access, Privilege Escalation, and Defense Evasion tend to be slightly lower when using a 3-shot prompt compared to a zero-shot prompt. Specifically, Initial Access decreases by 0.0278, Credential Access by 0.0298, Privilege Escalation by 0.0539, and Defense Evasion by 0.0544. These reductions suggest that for these classes, the inclusion of specific examples in the 3-shot prompt may have limited the model's ability to generalize.

However, other categories show improvement under the 3-shot prompt approach. Lateral Movement improves by 0.1111, Collection by 0.0814, Impact by 0.0444, and Discovery by 0.0413. These gains indicate that the additional contextual examples provided in the 3-shot prompt help the model better capture patterns for these classes, leading to higher recall. Additionally, the overall average recall improves from 0.7832 for zero-shot to 0.8016 for 3-shot, highlighting a slight but meaningful enhancement in overall performance. The 3-shot prompt also exhibits stronger capability of being able to detect benign activity as benign, with a recall score of 0.9991, compared to zero-shot score of 0.9962

It is crucial to highlight that when creating the 3-shot prompt, each example of a malicious log snippet used in the prompt is excluded from the analysis pool. This precaution is taken to avoid a scenario where a specific malicious log is simultaneously part of the prompt as a k-shot example and subject to analysis. This would result 100% detection accuracy for that specific attack technique, which would bias the results and overstate the system's actual performance. By removing these k-shot examples from the analysis, we ensure a more accurate evaluation of the model's ability to analyze and understand malicious and benign instances. This exclusion results in limited available data for certain minority classes: Initial Access has 9 logs, Exfiltration has 11 logs, Lateral Movement has 18 logs, Collection has 25 logs, and Impact has 27 logs. These classes were still decided to be included in the overall research because they represent critical stages in the attack lifecycle, and their inclusion ensures the evaluation covers the full spectrum of potential malicious activities. Even though these classes have fewer examples, their importance in understanding and mitigating threats outweighs the limitations posed by their smaller sample sizes.

### **5-Shot Binary Detection**

The prompt from the zero-shot binary classification task was enhanced by including examples of both malicious and benign log snippets. For the 5-shot prompt, each technique category, or class, was supplemented with 5 example snippets. The binary classification results of using the 5-shot prompt are presented below.

Technique Category	Malicious	Benign	Recall per class
Initial Access	7	0	1
Exfiltration	9	0	1
Lateral Movement	12	4	0.75
Collection	17	6	0.7391304348
Impact	20	5	0.8
Command-and-Control	32	16	0.6666666667
Persistence	48	7	0.8727272727
Execution	42	14	0.75
Credential Access	94	19	0.8318584071
Privilege Escalation	100	14	0.8771929825
Discovery	84	92	0.4772727273
Defense Evasion	234	128	0.6464088398
Benign Activity	1	1058	0.9990557129

Table 16. 5-shot Binary Detection Task Results

	Malicious (actual)	Benign Activity (actual)
Malicious (predicted)	699	1
Benign Activity (predicted)	305	1058

Table 17. 5-shot Binary Detection Task Results Summary Table

$$TP = 699 \quad FP = 1 \quad FN = 305 \quad TN = 1058$$

$$\text{Precision} = \frac{699}{699 + 1} \approx 0.9985 = 99.85\%$$

$$\text{Recall} = \frac{699}{699 + 305} \approx 0.6962 = 69.62\%$$

$$F1 = \frac{2 \cdot 0.9985 \cdot 0.6962}{0.9985 + 0.6962} \approx 0.8203 \approx 82.03\%$$

Examining the performance metrics reveals that, in comparison to the 5-shot prompt (Precision=0.9985, Recall=0.6962, F1=0.8203), the zero-shot prompt exhibits slightly lower precision (Precision=0.9948), yet surpasses it in recall (Recall=0.7255) and F1 score (F1=0.8391). This trend mirrors the results observed with the 3-shot prompt. It is likely that these diminishing returns can be attributed to the relatively small size of minority classes; the examples provided for these classes decrease even further in the 5-shot prompt

setting.

<b>Class</b>	<b>0-shot</b>	<b>5-shot</b>	<b>Recall Difference</b>
Initial Access	0.9166666667	1	0.0833333333
Exfiltration	0.9285714286	1	0.0714285714
Lateral Movement	0.6666666667	0.75	0.0833333333
Collection	0.6785714286	0.7391304348	0.0605590062
Impact	0.7333333333	0.8	0.0666666667
Command and control	0.6603773585	0.6666666667	0.0062893081
Persistence	0.8833333333	0.8727272727	-0.0106060606
Execution	0.7704918033	0.75	-0.0204918032
Credential Access	0.8559322034	0.8318584071	-0.0240737963
Privilege Escalation	0.9159663866	0.8771929825	-0.0387734041
Discovery	0.4475138122	0.4772727273	0.0297589151
Defense Evasion	0.727520436	0.6464088398	-0.0811115961
Benign Activity	0.9962406015	0.9990557129	0.0028151143
<b>Average</b>	<b>0.7831681122</b>	<b>0.8007933111</b>	<b>0.0176251988</b>

Table 18. Comparison Of Recall Scores Between Zero-shot And 5-shot Binary Detection Prompts

Average recall score for 3-shot prompt was 0.8016, while for 5-shot it is 0.8008. Average difference of recall scores for 3-shot prompt was 0.0184 while for 5-shot it is 0.0176. When comparing 5-shot with zero-shot, we can see that 5-shot is better at identifying malicious cases for Initial Access, Exfiltration, Lateral Movement, Collection, Impact and Command and Control classes, however those classes also have relatively small sample sizes as discussed previously. We can see that 5-shot can detect benign activity rather accurately, with a recall score of 0.9991 (very slightly better than 3-shot prompt). More detailed discussion and comparison of achieved results is provided in the Discussion subsection.

### 5.2.3 Chain of Thought

In this section, the focus shifts to Chain-of-Thought prompting technique. Below is a breakdown of binary detection results where CoT prompt was tasked with differentiating between malicious and benign logs. Per class results are as follows:

$$TP = 1024 \quad FP = 66 \quad FN = 40 \quad TN = 998$$

Having the TP, FP, FN and TN, we can then calculate the precision and recall can be

Technique Category	Malicious	Benign	Recall per class
Initial Access	11	1	0.916666
Exfiltration	14	0	1
Lateral Movement	18	3	0.857142
Collection	27	1	0.964285
Impact	29	1	0.966666
Command-and-Control	52	1	0.981132
Persistence	59	1	0.983333
Execution	60	1	0.983606
Credential Access	116	2	0.983050
Privilege Escalation	118	1	0.991596
Discovery	161	20	0.889502
Defense Evasion	359	8	0.978201
Benign Activity	66	998	0.937969

Table 19. Chain-of-Thought Binary Detection Task Results

	Malicious (actual)	Benign Activity (actual)
<b>Malicious (predicted)</b>	1024	66
<b>Benign Activity (predicted)</b>	40	998

Table 20. Chain-of-Thought Binary Detection Task Results Summary Table

calculated as follows:

$$\text{Precision} = \frac{1024}{1024 + 66} \approx 0.9394 = 93.94\%$$

$$\text{Recall} = \frac{1024}{1024 + 40} \approx 0.9624 = 96.24\%$$

$$\text{F1} = \frac{2 \cdot 0.9394 \cdot 0.9624}{0.9394 + 0.9624} \approx 0.9507 \approx 95.07\%$$

Upon analyzing the performance metrics, it becomes evident that the CoT prompt demonstrates better performance with a precision of 0.9394, recall of 0.9624, and an F1 score of 0.9507, compared to zero-shot approach, which yields a precision of 0.9948, recall of 0.7255, and an F1 score of 0.8391. A more detailed analysis of recall scores for individual classes clearly reveals enhanced performance:

The analysis shows that the CoT prompt enhances the recall metric of detection across nearly all categories. The sole exception is the Benign Activity class, which experiences a minor reduction in recall by 0.058. Nevertheless, on average, the CoT prompt increases

Class	0-shot	CoT	Recall Difference
Initial Access	0.9166666667	0.9166666667	0
Exfiltration	0.9285714286	1	0.07142857143
Lateral Movement	0.6666666667	0.8571428571	0.1904761905
Collection	0.6785714286	0.9642857143	0.2857142857
Impact	0.7333333333	0.9666666667	0.2333333333
Command and control	0.6603773585	0.9811320755	0.320754717
Persistence	0.8833333333	0.9833333333	0.1
Execution	0.7704918033	0.9836065574	0.2131147541
Credential Access	0.8559322034	0.9830508475	0.1271186441
Privilege Escalation	0.9159663866	0.9915966387	0.0756302521
Discovery	0.4475138122	0.8895027624	0.4419889503
Defense Evasion	0.727520436	0.9782016349	0.2506811989
Benign Activity	0.9962406015	0.9379699248	-0.05827067669
<b>Average</b>	<b>0.7831681122</b>	<b>0.9563965907</b>	<b>0.1732284785</b>

Table 21. Comparison Of Recall Scores Between Zero-shot And CoT Binary Detection prompts

recall scores by 0.1732, representing a substantial improvement, particularly when contrasted with the 1-shot, 3-shot, and 5-shot prompting strategies. The slight decline in recall for the Benign Activity class can be attributed to the Chain-of-Though prompt’s intensified emphasis on identifying patterns associated with malicious activities. This refinement enhances the model’s capacity to identify malicious activity in most categories. However, it may inadvertently increase sensitivity, leading to a minor increase in false positives within the Benign Activity class. This trade-off arises because the CoT prompt encourages in-depth reasoning, which occasionally results in the misclassification of benign patterns as malicious, due to their resemblance to more complex malicious patterns.

Furthermore, the minor reduction in recall for the Benign Activity class might also result from the reallocation of the model’s representational capacity. By prioritizing the detection of malicious activities, the CoT prompt directs more attention toward identifying usually underrepresented techniques. This reallocation may lead to a deprioritization of benign patterns, which are generally more prevalent and simpler to classify. Despite this slight compromise, the notable improvement in recall across malicious categories underscores the CoT prompt’s efficacy in enhancing threat detection. In the field of cybersecurity, it is generally preferable to encounter a false positive rather than a false negative. More detailed discussion and comparison of achieved results is provided in the Discussion subsection

## 5.2.4 Few-shot + CoT

This section focuses on the results of a prompt, where Chain-of-Thought is combined with 5-shot. The step-by-step thinking and rationalization method of CoT was supplemented with 5 examples per class. The resulting prompt was defined in the

Technique Category	Malicious	Benign	Recall per class
Initial Access	7	0	1
Exfiltration	9	0	1
Lateral Movement	13	3	0.8125
Collection	21	2	0.9130434783
Impact	24	1	0.96
Command-and-Control	42	6	0.875
Persistence	53	2	0.9636363636
Execution	49	7	0.875
Credential Access	109	4	0.9646017699
Privilege Escalation	110	4	0.9649122807
Discovery	135	41	0.7670454545
Defense Evasion	319	43	0.8812154696
Benign Activity	1	1058	0.9990557129

Table 22. Chain-of-Thought + 5-shot Binary Detection Task Results

	Malicious (actual)	Benign (actual)
<b>Malicious (predicted)</b>	891	1
<b>Benign Activity (predicted)</b>	113	1058

Table 23. Chain-of-Thought + 5-shot Binary Detection Task Results Summary Table

$$TP = 891 \quad FP = 1 \quad FN = 113 \quad TN = 1058$$

$$\text{Precision} = \frac{891}{891 + 1} \approx 0.9988 = 99.88\%$$

$$\text{Recall} = \frac{891}{891 + 113} \approx 0.8874 = 88.74\%$$

$$F1 = \frac{2 \cdot 0.9988 \cdot 0.8874}{0.9988 + 0.8874} \approx 0.9398 \approx 93.98\%$$

Upon examining the performance of the CoT + 5-shot prompt, it is evident that there

<b>Technique Category</b>	<b>Malicious</b>	<b>Benign</b>	<b>Recall Difference</b>
Initial Access	0.9166666667	1	0.0833333333
Exfiltration	0.9285714286	1	0.07142857143
Lateral Movement	0.6666666667	0.8125	0.1458333333
Collection	0.6785714286	0.9130434783	0.2344720497
Impact	0.7333333333	0.96	0.2266666667
Command and control	0.6603773585	0.875	0.2146226415
Persistence	0.8833333333	0.9636363636	0.0803030303
Execution	0.7704918033	0.875	0.1045081967
Credential Access	0.8559322034	0.9646017699	0.1086695665
Privilege Escalation	0.9159663866	0.9649122807	0.04894589415
Discovery	0.4475138122	0.7670454545	0.3195316424
Defense Evasion	0.727520436	0.8812154696	0.1536950336
Benign Activity	0.9962406015	0.9990557129	0.002815111433
<b>Average</b>	<b>0.7831681122</b>	<b>0.9212315792</b>	<b>0.138063467</b>

Table 24. Comparison Of Recall Scores Between Zero-shot And CoT + 5-shot Binary Detection Prompts

is a notable improvement in average recall scores across various classes. Notably, the CoT + 5-shot approach enhances the recall of detecting all classes without outliers. For minority classes, the limited size of the sample set likely contributes to the higher recall rates observed. When comparing the 5-shot approach (Precision=0.9985, Recall=0.6962, F1=0.8203) to the CoT + 5-shot (Precision=0.9988, Recall=0.8874, F1=0.9398), as well as their respective average recall score improvement per class (5-shot with an average recall of 0.8007 and an average improvement of 0.0176, compared to CoT + 5-shot with an average recall of 0.9212 and an improvement of 0.1380), it becomes apparent that CoT + 5-shot outperforms the simple 5-shot method. Interestingly, however, the CoT + 5-shot approach underperforms based on recall and F1 in comparison to the standalone CoT method (Precision=0.9394, Recall=0.9624, F1=0.9507). More detailed discussion and comparison of achieved results is provided in the Discussion subsection



### 5.2.5 Binary Detection Discussion

This section summarizes the results from the Binary Detection task across various prompting methods, including zero-shot, one-shot, three-shot, five-shot, Chain-of-Thought (CoT), and CoT combined with five-shot. Presented below is an overview of the Precision, Recall, and F1 scores obtained from each prompt used in the Binary Detection task.

Prompt	Precision	Recall	F1
0-shot	0.9948	0.7255	0.8391
1-shot	0.9972	0.6910	0.8163
3-shot	0.9986	0.7110	0.8306
5-shot	0.9985	0.6962	0.8204
CoT	0.9394	<b>0.9624</b>	<b>0.9507</b>
CoT + 5-shot	<b>0.9988</b>	0.8874	0.9398

Table 25. Comparison of Prompts Precision, Recall and F1 Scores in Binary Detection Task

Reviewing the data presented in the table above, it becomes evident that the CoT + 5-shot method achieved the highest Precision score. In contrast, the CoT approach alone attained the highest scores in both Recall and F1 metrics. This can be explained by the inherent trade-offs between precision and recall when combining Chain-of-Thought (CoT) reasoning with multiple examples per class (5-shot). The CoT + 5-shot approach provides the model with structured reasoning (via CoT) and additional context (via the 5-shot examples), which helps refine its decision boundaries, resulting in the highest precision score of 0.9989. However, the emphasis on precision often comes at the cost of recall, as the model may adopt a stricter classification threshold to minimize false positives, potentially leading to more false negatives.

On the other hand, the CoT approach alone focuses solely on enhancing reasoning without the added contextual examples from k-shot prompting. This allows the model to generalize more effectively across unseen instances, boosting recall to 0.9624 and achieving the highest F1 score (0.9508). The higher recall indicates that CoT alone is better at identifying a broader range of malicious activities, albeit with slightly reduced precision compared to CoT + 5-shot.

Class	# per Class	0-shot	1-shot	3-shot	5-shot	CoT	CoT+5-shot
Initial Access	12	<b>0.9167</b>	0.9091	0.8889	1.0000	<b>0.9167</b>	1.0000
Exfiltration	14	0.9286	0.8462	1.0000	1.0000	1.0000	1.0000
Lateral Movement	21	0.6667	0.7500	0.7778	0.7500	<b>0.8571</b>	0.8125
Collection	28	0.6786	0.7407	0.7600	0.7391	<b>0.9643</b>	0.9130
Impact	30	0.7333	0.6552	0.7778	0.8000	<b>0.9667</b>	0.9600
Command and control	53	0.6604	0.6538	0.6800	0.6667	<b>0.9811</b>	0.8750
Persistence	60	0.8833	0.8814	0.8947	0.8727	<b>0.9833</b>	0.9636
Execution	61	0.7705	0.6500	0.7931	0.7500	<b>0.9836</b>	0.8750
Credential Access	118	0.8559	0.8547	0.8261	0.8319	<b>0.9831</b>	0.9646
Privilege Escalation	119	0.9160	0.8559	0.8621	0.8772	<b>0.9916</b>	0.9649
Discovery	181	0.4475	0.4667	0.4888	0.4773	<b>0.8895</b>	0.7670
Defense Evasion	367	0.7275	0.6612	0.6731	0.6464	<b>0.9782</b>	0.8812
Benign Activity	1064	0.9962	0.9981	<b>0.9991</b>	<b>0.9991</b>	0.9380	<b>0.9991</b>
Average		0.7832	0.7633	0.8016	0.8008	<b>0.9564</b>	0.9212

Table 26. Comparison Of Recall Metrics Across Different Prompts

Analyzing recall metrics per class among different prompts, we observe that minority classes, such as Initial Access (12 examples), Exfiltration (14 examples), Lateral Movement (21 examples), Collection (28 examples), and Impact (30 examples), demonstrate unusually high recall scores under the 3-shot and 5-shot strategies. This phenomenon can be attributed to the limited sample space available for these classes, which potentially leads to overfitting when specific examples are included in the prompt. By incorporating examples directly related to these classes, the model may rely heavily on those patterns, artificially inflating recall scores for the limited dataset.

For instance, Exfiltration achieves perfect recall (1.000) under 3-shot, 5-shot, CoT, and CoT+5-shot prompts, suggesting that the inclusion of a few representative examples significantly enhances detection for this minority class. However, such improvements may not necessarily generalize to unseen data, as the model could be leveraging prompt-specific information rather than learning broader patterns.

In contrast, for majority classes like Benign Activity (1064 examples) or Defense Evasion (367 examples), recall is generally lower with multi-shot prompts, as these classes possess greater variability in their data. The model’s performance on these classes benefits more from reasoning techniques, such as CoT, which consistently shows substantial improvements in recall across diverse and complex classes, reaching 0.9782 for Defense Evasion and 0.8895 for Discovery.

Slightly lower recall metrics are observed for Discovery (0-shot: 0.4475; CoT: 0.8895) and Lateral Movement (0-shot: 0.6667; CoT: 0.8571). This could be, in part, due to

the wide diversity of available malicious techniques. These classes encompass a broad range of techniques, which increases variability in their patterns. For example, Discovery includes various reconnaissance and enumeration activities that can appear in numerous forms, making it challenging for the model to generalize effectively across all possible manifestations. Similarly, Lateral Movement involves diverse strategies, such as credential theft and exploitation, which further complicates accurate detection.

A contributing factor may be the difficulty in distinguishing malicious behaviors from benign activities: Both Discovery and Lateral Movement often overlap with legitimate system behaviors. For instance, administrative tasks or software updates may mimic reconnaissance or lateral access patterns, resulting in higher false negatives as the model struggles to differentiate between benign and malicious intent.

Techniques like Chain-of-Thought prompting, which emphasizes structured reasoning, show promise for addressing some of these difficulties, as seen by the improvements in recall for Discovery (from 0.4475 in 0-shot to 0.8895 in CoT). Future approaches could also involve specialized fine-tuning for high-variance classes and leveraging auxiliary data sources (or RAG approach) to reduce ambiguity in classification.

Overall, the CoT approach demonstrates the highest average recall (0.9564), indicating its ability to generalize effectively across both minority and majority classes. However, the observed recall inflation in smaller classes for 3-shot and 5-shot prompts underscores the importance of considering data distribution and potential overfitting when evaluating the effectiveness of prompting strategies, as well as selecting k-shots.

### 5.3 Malicious Activity Multiclass Classification

This section outlines the results of a multiclass classification task. Various prompting techniques were employed, including Zero-shot, One-shot, 3-shot, 5-shot, Chain-of-Thought, and Chain-of-Thought + 5-shot, to distinguish between malicious and benign activities and categorize them into the correct classes (such as identifying "Collection" as "Collection" and "Lateral Movement" as "Lateral Movement" and others).

#### 5.3.1 Zero-shot

Class	Collection	Lateral Movement	Command And Control	Discovery	Execution	Defense Evasion	Impact	Exfiltration	Initial Access	Persistence	Credential Access	Privilege Escalation	Benign Activity
<b>Collection</b>	<b>7</b>			7							4		
<b>Lateral Movement</b>		<b>5</b>	1		2	2					1		2
<b>Command And Control</b>			<b>9</b>	2	1	4	1			3	8		2
<b>Discovery</b>	10	5	8	<b>124</b>	10	35	1	1	1	3	19	5	16
<b>Execution</b>	2	4	23	27	<b>34</b>	124	4		6	15	25	37	
<b>Defense Evasion</b>	1	1	2	3	5	<b>125</b>	10			9	1	7	75
<b>Impact</b>	2					1	<b>5</b>						3
<b>Exfiltration</b>	5		3	1	1	1	1	<b>11</b>			2		20
<b>Initial Access</b>			1										
<b>Persistence</b>	1	2	2		1	34			1	<b>20</b>		43	3
<b>Credential Access</b>		2	2	7	3	8	2	1	1	2	<b>57</b>	6	
<b>Privilege Escalation</b>		2	1	8	4	30	6	1	3	8	1	<b>21</b>	2
<b>Benign Activity</b>			1	2		3							<b>941</b>
<b>Total</b>	28	21	53	181	61	367	30	14	12	60	118	119	1064

Table 27. Zero-shot Classification Results Table

The table presents the zero-shot classification results for a multiclass task involving a total of 2,128 logs, evenly distributed between 1,064 malicious logs and 1,064 benign logs. Using a zero-shot classification prompt, where the model relied solely on pre-trained knowledge without explicit examples in the prompt, several patterns emerged. The model demonstrated high precision in identifying the majority class for Benign Activity, correctly classifying 941 out of 1,064 logs. This reflects its strong capability to recognize benign behavior in a zero-shot setting. However, performance was moderate for some malicious classes, such as Discovery (124 correct out of 181) and Defense Evasion (125 correct

out of 367), highlighting the model’s partial understanding of these categories but also revealing significant misclassification into other classes, such as Execution.

Malicious classes with fewer samples, such as Lateral Movement (21 logs) and Collection (28 logs), exhibited poor classification accuracy, with frequent misclassifications into related or larger classes like Discovery.

A notable trend was confusion between conceptually similar classes, such as Defense Evasion and Execution, where many Defense Evasion logs were misclassified as Execution in 124 cases. This misclassification likely stems from the theoretical overlap between these attack strategies, as an attacker pursuing Defense Evasion techniques often necessitates achieving some form of Execution to bypass security mechanisms effectively. The close relationship between these two stages of an attack cycle creates challenges for the model in distinguishing them clearly, particularly in a zero-shot setting where contextual nuances are not explicitly provided. Similarly, Persistence techniques were frequently misclassified as Execution (15 occurrences) and Privilege Escalation as Persistence (43 occurrences). These findings suggest that Zero-shot prompting alone is insufficient for nuanced multi-class classification, as it lacks the contextual reasoning needed to effectively differentiate overlapping classes.

	<b>Recall</b>	<b>Precision</b>	<b>F1-score</b>
<b>Collection</b>	0.25	0.3888888889	0.3043478261
<b>Lateral Movement</b>	0.2380952381	0.3846153846	0.2941176471
<b>Command And Control</b>	0.1698113208	0.3	0.2168674699
<b>Discovery</b>	0.6850828729	0.5210084034	0.5918854415
<b>Execution</b>	0.5573770492	0.1129568106	0.1878453039
<b>Defense Evasion</b>	0.340599455	0.5230125523	0.4125412541
<b>Impact</b>	0.1666666667	0.4545454545	0.243902439
<b>Exfiltration</b>	0.7857142857	0.2444444444	0.3728813559
<b>Initial Access</b>	0	0	0
<b>Persistence</b>	0.3333333333	0.1869158879	0.2395209581
<b>Credential Access</b>	0.4830508475	0.6263736264	0.5454545455
<b>Privilege Escalation</b>	0.1764705882	0.2413793103	0.2038834951
<b>Benign Activity</b>	0.8843984962	0.9936642027	0.9358528095

Table 28. Zero-shot Recall, Precision, F1-Score Per Class

Macro Recall = 0.3900461657

Macro Precision = 0.3829080743

Macro F1 = 0.3499308112

Micro F1 = 0.638627

The table above presents the recall, precision, and F1-score metrics for the individual classes in the zero-shot classification task. The data reveals significant variability in performance across different categories. For instance, the Benign Activity class exhibits a high recall of 0.8844, precision of 0.9937, and a strong F1-score of 0.9359, indicating that the model successfully identifies benign activity with high accuracy. In contrast, several malicious activity classes, such as Initial Access, Impact, and Privilege Escalation, demonstrate poor performance, with recall values near zero for Initial Access and low F1-scores for Impact (0.2439) and Privilege Escalation (0.2039). These results further confirm the difficulty the model faces in detecting certain types of malicious activity within a zero-shot setup.

The confusion across certain classes, such as Collection and Lateral Movement, which both have low recall (0.25 and 0.2381, respectively), suggests that these attack techniques are too difficult to classify based on the limited behavioral characteristics available in the logs, making them difficult to distinguish. Additionally, classes like Exfiltration and Discovery show relatively higher recall values, with Exfiltration achieving 0.7857 and Discovery 0.6850, suggesting that the model can better identify these activities within the logs.

On average, the model's Macro Recall (0.3900), Macro Precision (0.3829), and Macro F1 (0.3499) reflect a relatively low performance when considering all classes equally. This suggests that while the model excels at recognizing benign activities, its ability to distinguish various malicious activities is inconsistent.

The micro F1 score of 0.6386, offers a more optimistic view of the model's overall performance. However, the significant performance disparities between individual classes indicate the need for further optimization. Additionally, improving the classification of harder-to-detect classes, such as Execution (F1-score of 0.1878), is crucial for achieving detection capabilities which would be useful in real world scenarios. More detailed discussion and comparison of achieved results is provided in the Discussion subsection.

### 5.3.2 Few-Shot

#### One-shot Classification

Class	Collection	Lateral Movement	Command And Control	Discovery	Execution	Defense Evasion	Impact	Exfiltration	Initial Access	Persistence	Credential Access	Privilege Escalation	Benign Activity
<b>Collection</b>	<b>7</b>			5							3		
<b>Lateral Movement</b>	1	<b>6</b>	2	2	2	4			1				
<b>Command And Control</b>			<b>12</b>	2	1	8			1	1	9	3	1
<b>Discovery</b>	3	3	5	<b>117</b>	11	25	1		1	1	27	8	2
<b>Execution</b>	3	1	16	13	<b>29</b>	98	5		2	8	11		4
<b>Defense Evasion</b>	4	3	2	4	6	<b>168</b>	11			10	2	12	17
<b>Impact</b>	2						<b>9</b>						
<b>Exfiltration</b>	6	1	7		1	1		<b>11</b>			1	24	1
<b>Initial Access</b>					2				<b>2</b>		1		
<b>Persistence</b>			5		1	26			2	<b>26</b>	1	44	1
<b>Credential Access</b>		2	1	15	2	7		1		2	<b>60</b>	8	
<b>Privilege Escalation</b>		3		9	3	15	2	1	2	11	2	<b>19</b>	1
<b>Benign Activity</b>	1	1	2	13	2	14	1						<b>1036</b>
<b>Total</b>	27	20	52	180	60	366	29	13	11	59	117	118	1063

Table 29. One-shot Classification Results Table

With the one-shot classification prompt, we observe similar challenges to the zero-shot setup, where a significant number of classes remain misclassified. However, a few classes show stronger accuracy, notably Discovery (117 correct out of 180), Defense Evasion (168 correct out of 366), and Benign Activity (1036 correct out of 1063). These classes demonstrate a relatively better performance due to their clearer distinguishing features in the data, likely aided by the single example provided in the prompt.

Despite these successes, there are recurring misclassifications. For example, Command and Control (C2C) is often confused with Execution. This is not surprising, as C2C activities commonly involve malicious implants listening for commands and executing them. Therefore, the model may focus more on the execution aspect of the process, leading to this mix-up. Similarly, Collection is frequently misclassified as Exfiltration, which makes sense given that collection is often a precursor to the exfiltration of data. This overlap in behavior results in the model mistakenly categorizing one as the other.

Moreover, Privilege Escalation is often confused with Persistence, which could be attributed to the similarities in the tools used by attackers in these stages. For instance, both privilege escalation and persistence techniques often involve the modification of system settings or use of similar tactics such as scheduled tasks, registry modifications, or process injections. These shared characteristics might lead the language model to struggle in accurately distinguishing between these two classes.

Overall, while the one-shot prompt offers some improvement over the zero-shot setup, the classification issues highlighted indicate that more training examples or advanced prompting techniques could help reduce confusion between conceptually similar attack techniques.

	<b>Recall</b>	<b>Precision</b>	<b>F1-score</b>
<b>Collection</b>	0.2592592593	0.4666666667	0.3333333333
<b>Lateral Movement</b>	0.3	0.3333333333	0.3157894737
<b>Command And Control</b>	0.2307692308	0.3157894737	0.2666666667
<b>Discovery</b>	0.65	0.5735294118	0.609375
<b>Execution</b>	0.4833333333	0.1526315789	0.232
<b>Defense Evasion</b>	0.4590163934	0.7029288703	0.5553719008
<b>Impact</b>	0.3103448276	0.8181818182	0.45
<b>Exfiltration</b>	0.8461538462	0.2075471698	0.3333333333
<b>Initial Access</b>	0.1818181818	0.4	0.25
<b>Persistence</b>	0.4406779661	0.2452830189	0.3151515152
<b>Credential Access</b>	0.5128205128	0.612244898	0.5581395349
<b>Privilege Escalation</b>	0.1610169492	0.2794117647	0.2043010753
<b>Benign Activity</b>	0.9746001881	0.9682242991	0.9714017815

Table 30. One-shot Recall, Precision, F1-Score Per Class

Macro Recall = 0.4469085145

Macro Precision = 0.4673671003

Macro F1 = 0.4149895088

Micro F1 = 0.7101654846

The one-shot prompting results continue to reveal significant variations in performance across classes. Benign Activity continues to perform well with high recall (0.9746), precision (0.9682), and F1-score (0.9714), indicating accurate identification of non-malicious activity. However, several malicious activity classes show poor performance. Initial Access has a low recall (0.1818) and F1-score (0.25), further confirming difficulty in detecting this class. Execution also struggles, with a recall of 0.4833 and F1-score of 0.232, likely due to its complex and context-dependent nature, as well as the diversity of available attack



vectors and attack techniques.

Exfiltration has a high recall (0.8462) but low precision (0.2075), indicating that while it is frequently identified, there are many false positives. Privilege Escalation performs poorly, with a recall of 0.1610 and F1-score of 0.2043, likely due to its overlap with other techniques like Persistence.

Discovery performs better with a recall of 0.65, precision of 0.5735, and F1 of 0.6094, suggesting it is easier to detect than more subtle attack behaviors. Collection, Lateral Movement, and Command and Control are often misclassified, likely due to shared behaviors across classes. Overall, while one-shot prompting improves performance, challenges remain in detecting and distinguishing complex or subtle attack techniques. The macro metrics (recall: 0.4469, precision: 0.4674, F1: 0.415) indicate room for improvement, particularly in balancing recall and precision across classes.

	<b>Macro Recall</b>	<b>Macro Precision</b>	<b>Macro F1</b>	<b>Micro F1</b>
<b>Zero-shot</b>	0.3900461657	0.3829080743	0.3499308112	0.6386278195
<b>One-shot</b>	0.4469085145	0.4673671003	0.4149895088	0.7101654846
<b>Difference</b>	0.05686234884	0.08445902594	0.06505869761	0.07153766508

Table 31. Zero-shot vs One-shot Macro and Micro Metrics

Analyzing the micro and macro metrics, we observe a noticeable improvement in classification performance when moving from zero-shot to one-shot prompts, with the latter outperforming in all four metrics. This indicates that simply adding one example per class via k-shot can boost performance. More detailed discussion and comparison of achieved results is provided in the Discussion subsection.

### 3-Shot Classification

Class	Collection	Lateral Movement	Command And Control	Discovery	Execution	Defense Evasion	Impact	Exfiltration	Initial Access	Persistence	Credential Access	Privilege Escalation	Benign Activity
<b>Collection</b>	7			6		1					5		
<b>Lateral Movement</b>		7	1	1	5	2	2				2	1	
<b>Command And Control</b>			12	7	3	5			1		6	1	2
<b>Discovery</b>	4		8	113	12	26	4				18	4	1
<b>Execution</b>	1	5	19		25	105	3		2	6	15	19	1
<b>Defense Evasion</b>	2	1		3	2	158	7		1	9	2	10	9
<b>Impact</b>	2					1	6			1			
<b>Exfiltration</b>	4	2	3	14	1	2		10					1
<b>Initial Access</b>			1	1	2	2			1			1	
<b>Persistence</b>		1	2		2	25			2	24	2	43	
<b>Credential Access</b>	4			8	1	3				2	59	9	
<b>Privilege Escalation</b>		1		8	5	16	5	1	2	12	5	25	
<b>Benign Activity</b>	1	1	4	17		18				3	1	3	1047
<b>Total</b>	25	18	50	178	58	364	27	11	9	57	115	116	1061

Table 32. 3-shot Classification Results Table

With the 3-shot classification prompt, we observe similar classification trends to the zero-shot and one-shot results. Discovery is correctly classified in 113 out of 178 cases, while Defense Evasion is accurately identified in 158 out of 364 instances, with 105 misclassifications as Execution. This misclassification is likely due to the overlap between Defense Evasion and Execution, as many defense evasion techniques necessitate execution, either directly or via services like scheduled tasks, LNK exploits, or WMI. Consequently, these two attack categories exhibit similar behavioral patterns, leading to frequent confusion.

Impact and Initial Access remain challenging to classify, likely because of their broad and somewhat ambiguous definitions. Despite these challenges, Benign Activity continues to be accurately identified, demonstrating consistent performance in distinguishing non-malicious activity. These trends indicate that while 3-shot prompting improves accuracy over zero-shot and one-shot methods, certain attack categories still present significant challenges due to their inherent complexity or conceptual overlap.

	<b>Recall</b>	<b>Precision</b>	<b>F1-score</b>
<b>Collection</b>	0.28	0.3684210526	0.3181818182
<b>Lateral Movement</b>	0.3888888889	0.3333333333	0.358974359
<b>Command And Control</b>	0.24	0.3243243243	0.275862069
<b>Discovery</b>	0.6348314607	0.5947368421	0.6141304348
<b>Execution</b>	0.4310344828	0.1243781095	0.1930501931
<b>Defense Evasion</b>	0.4340659341	0.7745098039	0.5563380282
<b>Impact</b>	0.2222222222	0.6	0.3243243243
<b>Exfiltration</b>	0.9090909091	0.2702702703	0.4166666667
<b>Initial Access</b>	0.1111111111	0.125	0.1176470588
<b>Persistence</b>	0.4210526316	0.2376237624	0.3037974684
<b>Credential Access</b>	0.5130434783	0.6860465116	0.5870646766
<b>Privilege Escalation</b>	0.2155172414	0.3125	0.2551020408
<b>Benign Activity</b>	0.986804901	0.9561643836	0.9712430427

Table 33. 3-shot Recall, Precision, F1-Score Per Class

Macro Recall = 0.4452048662

Macro Precision = 0.4390237226

Macro F1 = 0.4071063216

Micro F1 = 0.7151747247

	<b>Macro Recall</b>	<b>Macro Precision</b>	<b>Macro F1</b>	<b>Micro F1</b>
<b>Zero-shot</b>	0.3900461657	0.3829080743	0.3499308112	0.6386278195
<b>3-shot</b>	0.4452048662	0.4390237226	0.4071063216	0.7151747247
<b>Difference</b>	0.05515870057	0.05611564827	0.05717551036	0.0765469052

Table 34. Zero-shot vs 3-shot Macro and Micro Metrics

Overall, while some class-specific metrics remain low (e.g., Privilege Escalation with recall = 0.21551, precision = 0.3125, F1 = 0.255; Persistence with recall = 0.4210, precision = 0.2376, F1 = 0.3037; and others like Initial Access, Impact, and Command and Control), examining the macro and micro metrics reveals a clear improvement with the 3-shot prompt. Compared to the Zero-shot approach, the 3-shot method shows an increase in all key metrics: macro recall improves by 0.0551, macro precision by 0.0561, macro F1 by 0.0571, and micro F1 by 0.0765. This indicates that the use of a 3-shot prompt provides a more robust classification performance, especially in handling class misclassifications, improving both overall accuracy and consistency across different attack techniques. More detailed discussion and comparison of achieved results is provided in the Discussion subsection.

## 5-Shot Classification

Class	Collection	Lateral Movement	Command And Control	Discovery	Execution	Defense Evasion	Impact	Exfiltration	Initial Access	Persistence	Credential Access	Privilege Escalation	Benign Activity
<b>Collection</b>	<b>9</b>		1	4		1							
<b>Lateral Movement</b>		<b>11</b>	2	3	3	5					3		
<b>Command And Control</b>	1		<b>13</b>	6	2	7	1				7	2	1
<b>Discovery</b>	4	1	7	<b>108</b>	10	30	1	1		3	10	3	
<b>Execution</b>	4	1	13	18	<b>29</b>	95	3		2	4	15	23	
<b>Defense Evasion</b>	2	1	2	5	4	<b>140</b>	5		1	5	2	12	
<b>Impact</b>	1					1	<b>11</b>						
<b>Exfiltration</b>	1	1	4		1	1		8			1		1
<b>Initial Access</b>					2	1			<b>1</b>		2		
<b>Persistence</b>			2			21			1	<b>29</b>		35	
<b>Credential Access</b>	1			9	1	8			1	1	<b>65</b>	11	
<b>Privilege Escalation</b>			1	8	2	27	3		1	13	7	<b>27</b>	
<b>Benign Activity</b>		1	3	15	2	25	1				1	1	<b>1057</b>
<b>Total</b>	23	16	48	176	56	362	25	9	7	55	113	114	1059

Table 35. 5-shot Classification Results Table

Using a 5-shot classification prompt, the main trends from previous results persist, but with improvements in some classes. Lateral Movement is now identified with greater accuracy, occurring correctly in 11 out of 16 cases, compared to lower performance in earlier prompts. Impact is also more reliably classified, being identified in 11 out of 25 cases. Benign Activity continues to be classified accurately, with only 2 false positives. Additionally, the Defense Evasion continues to be misclassified as Execution frequently. However, the overall performance benefits from the additional context provided by the 5-shot approach.

	<b>Recall</b>	<b>Precision</b>	<b>F1-score</b>
<b>Collection</b>	0.3913043478	0.6	0.4736842105
<b>Lateral Movement</b>	0.6875	0.4074074074	0.511627907
<b>Command And Control</b>	0.2708333333	0.325	0.2954545455
<b>Discovery</b>	0.6136363636	0.606741573	0.6101694915
<b>Execution</b>	0.5178571429	0.1400966184	0.2205323194
<b>Defense Evasion</b>	0.3867403315	0.782122905	0.5175600739
<b>Impact</b>	0.44	0.8461538462	0.5789473684
<b>Exfiltration</b>	0.8888888889	0.4444444444	0.5925925926
<b>Initial Access</b>	0.1428571429	0.1666666667	0.1538461538
<b>Persistence</b>	0.5272727273	0.3295454545	0.4055944056
<b>Credential Access</b>	0.5752212389	0.6701030928	0.619047619
<b>Privilege Escalation</b>	0.2368421053	0.3033707865	0.2660098522
<b>Benign Activity</b>	0.9981114259	0.9556962025	0.976443418

Table 36. 5-shot Recall, Precision, F1-Score Per Class

Macro Recall = 0.5136203883

Macro Precision = 0.5059499229

Macro F1 = 0.478577689

Micro F1 = 0.7309743093

	<b>Macro Recall</b>	<b>Macro Precision</b>	<b>Macro F1</b>	<b>Micro F1</b>
<b>Zero-shot</b>	0.3900461657	0.3829080743	0.3499308112	0.6386278195
<b>5-shot</b>	0.5136203883	0.5059499229	0.478577689	0.7309743093
<b>Difference</b>	0.1235742227	0.1230418486	0.1286468778	0.09234648971

Table 37. Zero-shot vs 5-shot Macro and Micro Metrics

Comparing zero-shot to 5-shot in a multiclass classification task, we can see a significant improvement in all metrics. Macro Recall increases by 0.1236, indicating a better ability to identify different classes across the entire dataset. Macro Precision also improves by 0.1230, suggesting a reduction in false positives. Similarly, Macro F1 shows a notable increase of 0.1286. The Micro F1 score increases by 0.0923, indicating that the 5-shot model is more effective in handling the overall distribution of classes. More detailed discussion and comparison of achieved results is provided in the Discussion subsection

### 5.3.3 Chain of Thought

Class	Collection	Lateral Movement	Command And Control	Discovery	Execution	Defense Evasion	Impact	Exfiltration	Initial Access	Persistence	Credential Access	Privilege Escalation	Benign Activity
<b>Collection</b>	<b>4</b>			3							5		
<b>Lateral Movement</b>		<b>2</b>	2			1						1	
<b>Command And Control</b>	2	1	<b>13</b>	4		1					5	1	
<b>Discovery</b>	5		9	<b>95</b>	10	70	9	2	1	11	13	14	
<b>Execution</b>	7	8	19	33	<b>39</b>	137	6	3	4	25	38	47	10
<b>Defense Evasion</b>	3			4	3	<b>80</b>	8		2	2	3	7	2
<b>Impact</b>	1						<b>2</b>						
<b>Exfiltration</b>		1	2		1	1		<b>5</b>	1		2		
<b>Initial Access</b>						3	1	1				1	2
<b>Persistence</b>						19			2	<b>14</b>	1	23	1
<b>Credential Access</b>	1	2		6		7			2		<b>37</b>	8	
<b>Privilege Escalation</b>	1	3	2	4	1	9	1	1		5	3	<b>7</b>	
<b>Benign Activity</b>	4	4	6	32	7	39	3	2		3	11	10	<b>1049</b>
<b>total</b>	28	21	53	181	61	367	30	14	12	60	118	119	1064

Table 38. Chain-of-Thought Classification Results Table

Based on the table 42, we can observe that using CoT classification prompt, Defense Evasion was classified as Execution 137 times, but only 80 times as Defense Evasion itself. This suggests that the model struggles to distinguish between these two classes, which could be due to their behavioral similarities in some attack scenarios. A similar trend is observed for Persistence, which was correctly identified 14 out of 60 times but misclassified as Execution 25 times. This further indicates challenges in differentiating between techniques that involve similar processes. Likewise, Credential Access was misclassified as Execution 38 times, while being correctly identified as Credential Access only 37 times, suggesting confusion in distinguishing these two concepts. Finally, for Benign Activity, there is a noticeable drop in correct classifications compared to previous results, pointing to potential issues in effectively separating benign and malicious activities, particularly when they share similar characteristics in the system's behavior.

	<b>Recall</b>	<b>Precision</b>	<b>F1-score</b>
<b>Collection</b>	0.1428571429	0.3333333333	0.2
<b>Lateral Movement</b>	0.09523809524	0.3333333333	0.1481481481
<b>Command And Control</b>	0.2452830189	0.4814814815	0.325
<b>Discovery</b>	0.5248618785	0.3974895397	0.4523809524
<b>Execution</b>	0.6393442623	0.1037234043	0.1784897025
<b>Defense Evasion</b>	0.2179836512	0.701754386	0.3326403326
<b>Impact</b>	0.06666666667	0.6666666667	0.1212121212
<b>Exfiltration</b>	0.3571428571	0.3846153846	0.3703703704
<b>Initial Access</b>	0	0	0
<b>Persistence</b>	0.2333333333	0.2333333333	0.2333333333
<b>Credential Access</b>	0.313559322	0.5873015873	0.408839779
<b>Privilege Escalation</b>	0.05882352941	0.1891891892	0.08974358974
<b>Benign Activity</b>	0.9859022556	0.8965811966	0.93912265

Table 39. CoT Recall, Precision, F1-Score Per Class

Macro Recall = 0.2985381549

Macro Precision = 0.4083694489

Macro F1 = 0.292252383

Micro F1 = 0.6329887218

	<b>Macro Recall</b>	<b>Macro Precision</b>	<b>Macro F1</b>	<b>Micro F1</b>
<b>Zero-shot</b>	0.3900461657	0.3829080743	0.3499308112	0.6386278195
<b>CoT</b>	0.2985381549	0.4083694489	0.292252383	0.6329887218
<b>Difference</b>	<b>-0.09150801081</b>	<b>0.02546137459</b>	<b>-0.05767842819</b>	<b>-0.005639097744</b>

Table 40. Zero-shot vs CoT Macro and Micro Metrics

One possible explanation for this is that while Chain of Thought (CoT) allows the model to reason through more complex relationships and processes, it may also introduce unnecessary complexity that detracts from the model’s performance in a task like this. The added reasoning steps could cause the model to misinterpret certain attack techniques or their relationships, leading to a drop in classification accuracy. For instance, when the model attempts to break down a classification task into multiple steps or logical deductions, it might lose focus on key features or attributes of a class, resulting in a decrease in both recall and precision. Additionally, CoT might introduce more room for error when distinguishing between similar classes or when context is critical, as seen in the misclassifications of Defense Evasion and Persistence as Execution. Thus, the reasoning process introduced by CoT could potentially increase misclassifications and reduce the model’s overall efficiency in the multiclass classification task. More detailed discussion and comparison of achieved results is provided in the Discussion subsection

### 5.3.4 Few-shot + CoT

Class	Collection	Lateral Movement	Command And Control	Discovery	Execution	Defense Evasion	Impact	Exfiltration	Initial Access	Persistence	Credential Access	Privilege Escalation	Benign Activity
<b>Collection</b>	<b>5</b>			3		1					3		
<b>Lateral Movement</b>		<b>5</b>	1	1	2	4							
<b>Command And Control</b>	1		<b>11</b>	1		4				2	6	2	
<b>Discovery</b>	5	1	5	<b>91</b>	7	63	5	1		2	16	8	
<b>Execution</b>	4	3	19	26	<b>41</b>	104	5	1	<b>1</b>	5	26	27	
<b>Defense Evasion</b>	1		1	1	1	<b>88</b>			<b>1</b>	7		5	
<b>Impact</b>							<b>6</b>				1		
<b>Exfiltration</b>		1	3	1	1			<b>7</b>				1	
<b>Initial Access</b>				1	1		1		4	2		2	
<b>Persistence</b>		1				14			1	<b>20</b>	1	34	
<b>Credential Access</b>		2		8	1	7	2				<b>44</b>	9	1
<b>Privilege Escalation</b>	3	1	2	8		19	2			9	7	<b>13</b>	
<b>Benign Activity</b>	4	2	6	35	2	58	4			8	9	13	<b>1058</b>
<b>total</b>	23	16	48	176	56	362	25	9	7	55	113	114	1059

Table 41. Chain-of-Thought + 5-shot Classification Results Table

By examining the results from the CoT + 5-shot classification, we observe that the existing trends largely persist. The Discovery category stands out with a high classification accuracy (albeit still mostly misclassified as Execution), identified 91 times. Likewise, the Defense Evasion category also shows strong results, being correctly classified 88 times. However, there remains a challenge as Defense Evasion is still frequently misclassified as Execution, even with the addition of k-shots and a clarifying CoT prompt explaining their differences. The Persistence category shows more consistency, with correct classification in 20 out of 55 instances. Meanwhile, benign activities continue to be correctly identified as benign in strong majority of cases.



	<b>Recall</b>	<b>Precision</b>	<b>F1-score</b>
<b>Collection</b>	0.2173913043	0.4166666667	0.2857142857
<b>Lateral Movement</b>	0.3125	0.3846153846	0.3448275862
<b>Command And Control</b>	0.2291666667	0.4074074074	0.2933333333
<b>Discovery</b>	0.5170454545	0.4460784314	0.4789473684
<b>Execution</b>	0.7321428571	0.1564885496	0.2578616352
<b>Defense Evasion</b>	0.2430939227	0.8380952381	<b>0.3768736617</b>
<b>Impact</b>	0.24	0.8571428571	0.375
<b>Exfiltration</b>	0.7777777778	0.5	0.6086956522
<b>Initial Access</b>	0.5714285714	0.3636363636	0.4444444444
<b>Persistence</b>	0.3636363636	0.2816901408	0.3174603175
<b>Credential Access</b>	0.389380531	0.5945945946	0.4705882353
<b>Privilege Escalation</b>	0.1140350877	0.203125	0.1460674157
<b>Benign Activity</b>	0.9990557129	0.8824020017	0.9371124889

Table 42. CoT + 5-shot Recall, Precision, F1-Score Per Class

Macro Recall = 0.4389734038

Macro Precision = 0.4870725104

Macro F1 = 0.4105328019

Micro F1 = 0.6752302472

	<b>Macro Recall</b>	<b>Macro Precision</b>	<b>Macro F1</b>	<b>Micro F1</b>
<b>Zero-shot</b>	0.3900461657	0.3829080743	0.3499308112	0.6386278195
<b>CoT+5-shot</b>	0.4389734038	0.4870725104	0.4105328019	0.6752302472
<b>Difference</b>	<b>0.04892723817</b>	<b>0.1041644361</b>	<b>0.06060199068</b>	<b>0.03660242766</b>

Table 43. Zero-shot vs CoT Macro and Micro Metrics

The comparison between the Zero-shot and CoT + 5-shot methods reveals improvements in the classification performance when using the latter approach. Specifically, the Macro Recall increases from 0.390 to 0.439, reflecting a gain of approximately 4.89%. This suggests that CoT + 5-shot improves the model’s ability to correctly identify instances across different classes. Moreover, Macro Precision demonstrates a more significant improvement, rising from 0.383 to 0.487 (an increase of 10.41%), indicating better accuracy in predicting positive instances without misclassifying them as other classes.

The Macro F1 score also shows an increase from 0.350 to 0.411 (a 6.06% improvement). The Micro F1 score increases from 0.639 to 0.675, reflecting an overall enhancement in the model’s classification capabilities. More detailed discussion and comparison of achieved results is provided in the Discussion subsection.

### 5.3.5 Multiclass Classification Discussion

This section summarizes the results from the Multiclass Classification Detection task across various prompting methods, including zero-shot, one-shot, 3-shot, 5-shot, Chain-of-Thought (CoT), and CoT combined with 5-shot. Below are presented micro and macro metrics from the utilized prompting techniques

Prompt	Macro Recall	Macro Precision	Macro F1	Micro F1
Zero-shot	0.3900	0.3829	0.3499	0.6386
One-shot	0.4469	0.4673	0.4149	0.7101
3-shot	0.4452	0.4390	0.4071	0.7151
5-shot	<b>0.5136</b>	<b>0.5059</b>	<b>0.4785</b>	<b>0.7309</b>
CoT	0.2985	0.4083	0.2922	0.6329
CoT+5-shot	0.4389	0.4870	0.4105	0.6752

Table 44. Comparison of Prompts Micro and Macro Metrics in Classification Task

In the multiclass classification task, 5-shot prompting emerged as the most effective approach, achieving the highest micro F1 score of 0.7309. This performance highlights the importance of providing multiple examples to the model, particularly in the analysis Windows logs, where understanding normal system behavior plays a critical role in successful classification. However, the performance on other metrics, such as macro recall and macro precision, is less impressive. This discrepancy likely arises due to the wide diversity of tactics, many of which exploit unique and highly varied attack vectors.

The limitations of 5-shot prompting are partly attributable to the fixed context length, which restricts the number of relevant examples that can be included in a single prompt. Consequently, while the approach excels in identifying common or well-represented patterns, it struggles with classes that require nuanced differentiation or have less representation in the models training data.

Class	Zero-shot	One-shot	3-shot	5-shot	CoT	CoT + 5-shot
Collection	0.2500	0.2593	0.2800	<b>0.3913</b>	0.1429	0.2174
Lateral Movement	0.2381	0.3000	0.3889	<b>0.6875</b>	0.0952	0.3125
Command And Control	0.1698	0.2308	0.2400	<b>0.2708</b>	0.2453	0.2292
Discovery	<b>0.6851</b>	0.6500	0.6348	0.6136	0.5249	0.5170
Execution	0.5574	0.4833	0.4310	0.5179	0.6393	<b>0.7321</b>
Defense Evasion	0.3406	<b>0.4590</b>	0.4341	0.3867	0.2180	0.2431
Impact	0.1667	0.3103	0.2222	<b>0.4400</b>	0.0667	0.2400
Exfiltration	0.7857	0.8462	<b>0.9091</b>	0.8889	0.3571	0.7778
Initial Access	0.0000	0.1818	0.1111	0.1429	0.0000	<b>0.5714</b>
Persistence	0.3333	0.4407	0.4211	<b>0.5273</b>	0.2333	0.3636
Credential Access	0.4831	0.5128	0.5130	<b>0.5752</b>	0.3136	0.3894
Privilege Escalation	0.1765	0.1610	0.2155	<b>0.2368</b>	0.0588	0.1140
Benign Activity	0.8844	0.9746	0.9868	0.9981	0.9859	<b>0.9991</b>

Table 45. Comparison Of Per-class Recall in Classification Task

Analyzing the per-class recall, it is evident that 5-shot prompting generally achieves the highest scores across most classes, though the results remain moderate for several categories. Notable improvements with 5-shot are observed in classes such as Lateral Movement (0.6875), Impact (0.4400), and Persistence (0.5273). This suggests that providing additional examples often enhances the model’s understanding, particularly for more nuanced activities.

However, the results also indicate diminishing returns or even potential confusion in certain cases. For instance, in the Discovery class, recall drops slightly from the highest zero-shot score of 0.6851 to 0.6136 in 5-shot. Similarly, for Defense Evasion, recall declines from its peak of 0.4590 (one-shot) to 0.3867 in 5-shot. This might be due to the increased context introducing ambiguity or overlapping features between classes, especially for conceptually similar tactics.

Moreover, certain classes such as Exfiltration (0.9091 in 3-shot) and Benign Activity (0.9991 in CoT + 5-shot) demonstrate that the approach performs exceptionally well for more distinct or well-defined behaviors. Yet, for inherently ambiguous categories like Command and Control or Privilege Escalation, even 5-shot does not drastically improve recall, possibly reflecting a need for more context beyond the fixed prompt size.

Overall, while 5-shot prompting provides measurable gains in some cases, it also underscores challenges in multiclass classification when faced with overlapping tactics or limited capacity for nuanced differentiation.

## 6. Discussion

This section presents the results of utilizing various prompting methods for Binary Detection and Multiclass Classification of both malicious and benign Windows Event logs.

### Binary Detection Task Discussion

The Binary Detection task results reveal key insights into the effectiveness of various prompting techniques: zero-shot, one-shot, three-shot, five-shot, Chain-of-Thought (CoT), and CoT + 5-shot. Each method demonstrates unique strengths and limitations.

The CoT + 5-shot method achieves the highest Precision score (0.9988), driven by structured reasoning and detailed context from 5-shot examples. This combination sharpens decision boundaries, enhancing precision but lowering recall due to stricter thresholds that exclude some true positives, increasing false negatives. Conversely, standalone CoT excels in Recall (CoT Recall of 0.9624 vs CoT+5-shot Recall of 0.8874) and F1 score (CoT F1 of 0.9507 vs CoT+5-shot F1 of 0.9398), relying on reasoning rather than contextual examples. This flexibility helps it identify a broader range of incidents, though at a slight precision tradeoff compared to CoT + 5-shot.

Minority classes like Initial Access and Exfiltration show inflated recall under 3-shot and 5-shot prompts, indicating overfitting to narrow patterns due to limited data. While these methods improve recall for smaller classes, they risk sacrificing generalization for unseen examples. For majority classes, such as Benign Activity and Defense Evasion, CoT delivers strong recall by leveraging nuanced reasoning to capture diverse behaviors. However, lower recall for Discovery and Lateral Movement highlights challenges in generalizing across their wide technique range and overlap with benign actions.

Overall, CoT's strong recall across varied classes underscores its potential, though overfitting in smaller classes with 3-shot and 5-shot methods stresses the need for balanced data distribution and prompt selection to enhance model reliability.

### Multiclass Classification Task Discussion

The multiclass classification analysis highlights both the strengths and limitations of various prompting methods, with 5-shot prompting standing out in certain areas. Achieving the highest micro F1 score of 0.7309, it demonstrates the effectiveness of providing multiple examples, particularly in the context of complex tasks like analyzing Windows logs, where

distinguishing between diverse system behaviors is crucial. This suggests that carefully selected examples enhance the model's ability to generalize across recurring patterns.

However, the metrics such as macro recall and macro precision, reveal the constraints of 5-shot prompting in handling the diversity and complexity of attack tactics. The limited prompt size struggles to encompass the wide range of behaviors, particularly for underrepresented or highly variable tactics requiring more intricate differentiation.

Per-class recall analysis shows significant improvements for classes like Lateral Movement, Impact, and Persistence, where additional examples enrich the model's contextual understanding. Yet, declines in recall for Discovery and Defense Evasion indicate challenges arising from overlapping features or ambiguities introduced by constrained prompt sizes.

Exceptional recall scores in Exfiltration (0.9091) and Benign Activity (0.9991) reflect the method's ability to identify distinct and well-defined patterns. Conversely, performance for classes like Command and Control and Privilege Escalation remains less effective, indicating difficulties in accommodating complex contexts within fixed prompts.

In summary, while 5-shot prompting enhances classification for specific classes, it underscores persistent challenges in differentiating overlapping tactics and managing prompt size constraints. Future research should explore dynamic context incorporation and adaptive prompt strategies to build more robust models for incident response and digital forensics using LLMs.

Overall, this research demonstrates that the choice of prompting method should align with the complexity of the task - binary detection versus more detailed activity classification. Cyberattack activities often blur the lines between categories, as their classification depends heavily on context, including preceding and subsequent actions. For instance, creating a scheduled task to execute a malicious file at system boot would typically be classified as a persistence technique. However, if the attacker initially escalated privileges to gain access necessary to configure the scheduled task, the same action might also be considered privilege escalation. Similarly, during intra-network lateral movement, the preceding activity often involves enumeration and analysis of available hosts on the network, which would typically be classified as discovery. However, if the attacker subsequently leverages this information to move laterally across the network, the activity shifts classification to lateral movement - even though the initial steps are more closely aligned with discovery.

As such, accurately classifying activity without understanding the full picture is incredibly challenging. This contextual dependence makes it unsurprising that the LLMs struggled to

achieve consistently strong scores. One key observation from this research is that the Chain of Thought prompting strategy performed the best in binary classification tasks, achieving the highest F1-score of 95.07%. In contrast, all prompting strategies yielded relatively low F1-scores in multiclass scenarios. This suggests that binary classification may be more appropriate for practical use, as identifying an issue from the logs can be much more challenging for human analysts, while classifying an already identified log fragment is a more straightforward task. The CoT strategy, which excels in binary classification, stands out because it does not rely on a set of pre-existing examples, distinguishing it from other strategies that use previously created examples in the prompt. In the context of LLMs, such approaches can be referred to as unsupervised LLM-based methods.

## 6.1 Limitations

The study is subject to several limitations. First, the experimentation with LLM, specifically GPT-4o, was cost-constrained, allowing each prompt to be executed only once. This is a significant limitation, as LLMs are non-deterministic, and their outputs can vary across different executions. Ideally, multiple executions per prompt would provide a more accurate assessment of the variability in the detection metrics and results. Another limitation is the use of public LLMs, which, in the context of log analysis, could lead to sensitive log data being shared with a third-party service provider. In a real-world scenario, it would be preferable to use a self-hosted model to ensure greater data privacy and security.

Secondly, the study's focus was limited to Windows 10, despite the existence of numerous other operating systems, including legacy Windows (XP, 7 etc.) versions, various Linux distributions (Ubuntu, Fedora etc.), and cloud-based environments (AWS, GCP, Azure etc.), which are also frequent targets of cyber attacks. This narrow focus restricts the generalizability of the findings.

Thirdly, the dataset used in the study was generated using more aggressive simulated attack tactics, suitable for detecting "smash and grab" style attacks but inadequate for analyzing Advanced Persistent Threats style of attacks that employ more strategic, prolonged methodologies frequently spanning weeks or even months. The current dataset and the context window size limitations of LLMs do not support such analysis.

Additionally, the malicious dataset was generated using Windows virtual machines, while the benign dataset was sourced from the author's physical, non-virtualized PC. Although efforts were made to standardize the logging configuration across both datasets, the difference in the underlying environments likely introduced inconsistencies and slight structural differences. Ideally, both datasets would have been generated on the same

machine or host to minimize such discrepancies and ensure greater consistency between the datasets.

Lastly, the rapidly evolving landscape of LLMs presents another limitation. The study exclusively used GPT-4o, without exploring models which are more oriented towards technical content such as DeepSeek or employing fine-tuning techniques that could enhance performance (Guo et al., 2024). This reliance on a single, non-finetuned model further constrains the study's scope and applicability.

## **6.2 Future Work**

The future of research on the application of large language models for log analysis in incident response and digital forensics offers numerous avenues to explore. It is based on addressing key areas of development and innovation to improve the effectiveness and reliability of LLMs and related advances.

Enhancing dataset diversity remains a critical priority. Future studies should integrate a broader range of operating systems and environments, encompassing legacy Windows versions, diverse Linux distributions, and cloud-based platforms. This expansion has the potential to improve the generalizability of findings and also provide a more nuanced understanding of LLM applicability across heterogeneous systems and contexts.

Another possible avenue could be the inclusion of Advanced Persistent Threat attack scenarios (e.g., based on known APT TTPs) or simply more advanced attack types in datasets. APTs frequently have more sophisticated and delicate tactics & techniques, and hence demand datasets that authentically reflect such activities. Incorporating these scenarios would enable researchers to better evaluate LLM capabilities to identify complex and stealthy attack patterns over extended timeframes.

To address the inherent non-deterministic nature of LLMs, future work should involve conducting multiple execution trials for each prompt. This practice would help quantify variability in detection metrics, thus ensuring a more robust and reliable evaluation of LLM performance across different scenarios.

Exploring alternative LLM models, such as DeepSeek, alongside fine-tuning techniques is another promising avenue. These efforts could potentially enhance models specialization and adaptability to cybersecurity-specific contexts, potentially uncovering new capabilities for detecting and mitigating threats in a targeted and efficient manner.

Additionally, research could focus on dynamic context incorporation and adaptive prompt engineering. These techniques could help with tackling problems encountered in this research, such as overlapping attack tactics and constrained prompt sizes.

Achieving balanced data representation is equally important. Ensuring equitable distribution across different simulated attack type classes, particularly for minority attack vectors, will help mitigate overfitting and provide further insight into the detection capability of the underrepresented classes. This step is essential for assessing the accuracy of detections, especially with less frequent but highly significant cyber threats.



## 7. Conclusion

This thesis makes several important contributions to the existing body of knowledge and addresses a significant gap in current research. Specifically, it offers a comprehensive examination of the capabilities of Large Language Models in cyber threat detection - an area that has received relatively little attention in prior studies.

The research highlights that binary classification may be a more practical application of LLMs in this domain. While human analysts often find it challenging to identify issues directly from raw logs, classifying pre-identified log fragments is a comparatively simpler and more effective task. Among the various strategies analyzed, the Chain-of-Thought (CoT) approach emerges as particularly effective for binary classification. Unlike other methods that rely on pre-existing examples embedded within the prompt, the CoT strategy operates independently of such examples, offering both flexibility and adaptability across diverse operational environments. Within the context of LLMs, this approach can be characterized as an unsupervised LLM-based detection method.

Another major accomplishment of this study is the creation of a new dataset consisting of 1064 unique attack technique logs. Each attack log contains a detailed set of Windows Event logs, which include Sysmon, Security, System, Application, and PowerShell logs. This dataset stands out due to its breadth and pertinence, overcoming the constraints of existing host-intrusion datasets that are either outdated or narrowly focused on certain types of attacks. This dataset covers 13 different categories from the MITRE ATT&CK framework and includes a range of attacks from simple to complex, making it a valuable resource for further study and experimentation. The accompanying dataset generation scripts bring further value to the field, allowing users to customize the dataset creation process. Instead of relying solely on the default Windows 10 setup, users can replace it with their own system snapshot. This flexibility makes it possible to simulate attacks tailored to specific research needs or organizational setups, enabling researchers and practitioners to create scenarios that better reflect their unique environments. The dataset can be accessed from the following [Atomic EVTX GitHub Repository](#). The dataset generation script is available in the appendices of this document.

In addressing the main research question, whether Large Language Models can enhance the operational efficiency of incident response and digital forensic processes, this thesis unveils promising insights and identifies areas where further development is required.

Preliminary findings indicate that LLMs indeed have the potential to augment the detection of cyberattacks through the analysis of Windows logs. In regard to **RQ1**, the study established that LLMs, when employing Chain-of-Thought (CoT) or CoT combined with a 5-shot prompt strategy, perform exceptionally well in binary detection of malicious activity. Pure CoT achieved Precision of 0.9394, Recall of 0.9624, F1 of 0.9507 while CoT+5-shot achieved Precision of 0.9988, Recall of 0.8874, F1 of 0.9398. This demonstrates a substantial advancement in detection capabilities, showcasing LLMs as viable tools in cyber threat detection tasks.

For **RQ2**, the potential of LLMs in classifying cyberattacks was validated, although the results suggest that the current model configuration necessitates additional refinement. The 5-shot prompting strategy yielded the highest scores with Micro F1 score of 0.7309, Macro F1 score of 0.4785, Macro Precision of 0.5059 and Macro Recall of 0.5136. The complexity of multiclass classification appears to require further fine-tuning of the model parameters to reach optimal efficacy or more refined prompting strategy. This reveals a path for future research, aimed at enhancing the adaptability and specificity of LLMs in nuanced forensic scenarios.

Lastly, **RQ3** explored optimal prompting strategies, concluding that CoT and CoT with a 5-shot strategy are most effective for binary detection tasks, while 5-shot prompts excel in multiclass classification contexts. These findings provide a systematic approach to employing LLMs efficiently, offering a tactical advantage in real-world incident response and digital forensic applications.

Overall, the study affirms the transformative potential of LLMs and their potential value in incident response and digital forensics. Nonetheless, more research is required to ensure that these LLMs can be leveraged to their fullest potential in the rapidly evolving landscape of cybersecurity. Through continued exploration and refinement, LLMs are poised to become indispensable assets in the digital forensic and incident response toolkit.

## References

- Bayer, M., Kuehn, P., Shanehsaz, R., & Reuter, C. (2024). Cysecbert: A domain-adapted language model for the cybersecurity domain. *ACM Transactions on Privacy and Security*, 27(2), 1–20.
- Bridge. (2024). Event logging security - win32 apps — learn.microsoft.com [[Accessed 08-12-2024]].
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodi, D. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165. <https://arxiv.org/abs/2005.14165>
- CAIDA. (2007). The caida "ddos attack 2007" dataset — caida.org [[Accessed 08-12-2024]].
- Canary. (2024). Github - invoke-atomicredteam - powershell module to execute tests [[Accessed 08-12-2024]].
- Canary, R. (2024). Atomic red team. *o. D.[Online]*. Available: <https://github.com/redcanaryco/atomic-red-team>.
- Casey, E. (2009). *Handbook of digital forensics and investigation*. Academic Press.
- Catak, F. O., Ahmed, J., Sahinbas, K., & Khand, Z. H. (2021). Data augmentation based malware detection using convolutional neural networks. *PeerJ Computer Science*, 7, e346. <https://doi.org/10.7717/peerj-cs.346>
- Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., et al. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3), 1–45.
- Cheshkov, A., Zadorozhny, P., & Levichev, R. (2023). Evaluation of chatgpt model for vulnerability detection. *arXiv preprint arXiv:2304.07232*.
- Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012, August). *Computer security incident handling guide: Recommendations of the national institute of standards and technology*. <https://doi.org/10.6028/nist.sp.800-61r2>
- Clairoux-Trepanier, V., Beauchamp, I.-M., Ruellan, E., Paquet-Clouston, M., Paquette, S.-O., & Clay, E. (2024). The use of large language models (llm) for cyber threat intelligence (cti) in cybercrime forums. *arXiv preprint arXiv:2408.03354*.
- Cranford. (2023). What is Red Teaming? | CrowdStrike — crowdstrike.com [[Accessed 08-12-2024]].

- Creech, G. (2014). *Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks* [Doctoral dissertation]. UNSW Sydney. <https://doi.org/10.26190/UNSWORKS/16615>
- CrowdStrike. (2024). CrowdStrike [[Accessed 08-12-2024]].
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. N. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. <https://arxiv.org/abs/1810.04805>
- Ferrag, M. A., Ndhlovu, M., Tihanyi, N., Cordeiro, L. C., Debbah, M., & Lestable, T. (2023). Revolutionizing cyber threat detection with large language models. *arXiv preprint arXiv:2306.14263*.
- Garfinkel, S. L. (2010). Digital forensics research: The next 10 years [The Proceedings of the Tenth Annual DFRWS Conference]. *Digital Investigation*, 7, S64–S73. <https://doi.org/https://doi.org/10.1016/j.diin.2010.05.009>
- Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y., et al. (2024). Deepseek-coder: When the large language model meets programming—the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.
- Hadi, M. U., tashi qasem al, q. a., Qureshi, R., Shah, A., muneer amgad, a., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., & Mirjalili, S. (2023). A survey on large language models: Applications, challenges, limitations, and practical usage. <https://doi.org/10.36227/techrxiv.23589741.v1>
- Handa, A., Sharma, A., & Shukla, S. K. (2019). Machine learning in cybersecurity: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4), e1306.
- Hays, S., & White, J. (2024). Employing llms for incident response planning and review. *arXiv preprint arXiv:2403.01271*.
- Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., et al. (2023). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.
- Jiang, Z., Liu, J., Chen, Z., Li, Y., Huang, J., Huo, Y., He, P., Gu, J., & Lyu, M. R. (2024). Lilac: Log parsing using llms with adaptive parsing cache. *Proceedings of the ACM on Software Engineering*, 1(FSE), 137–160.
- Kaminsky. (2024). Ways to detect and curb Living off the Land (LotL) attacks — kaspersky.com [[Accessed 08-12-2024]].
- Khan, S., Parkinson, S., & Murphy, C. (2023). Context-based irregular activity detection in event logs for forensic investigations: An itemset mining approach. *Expert Systems with Applications*, 233, 120991. <https://doi.org/https://doi.org/10.1016/j.eswa.2023.120991>

- Le, V.-H., & Zhang, H. (2023). Log parsing with prompt-based few-shot learning. *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2438–2449.
- Ma, Z., Chen, A. R., Kim, D. J., Chen, T.-H., & Wang, S. (2024). LImparser: An exploratory study on using large language models for log parsing. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 1–13.
- markruss. (2024). Sysmon - sysinternals — learn.microsoft.com [[Accessed 08-12-2024]].
- Martínez Torres, J., Iglesias Comesaña, C., & García-Nieto, P. J. (2019). Machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics*, 10(10), 2823–2836.
- Metz. (2024). Libevtx/documentation/windows xml event log (evtx).asciidoc at main · libyal/libevtx — github.com [[Accessed 08-12-2024]].
- Microsoft. (2024). About logging<sub>windows</sub> – powershell – learn.microsoft.com [[Accessed 08-12-2024]].
- Microsoft. (2022). Appendix 1 - events to monitor — learn.microsoft.com [[Accessed 08-12-2024]].
- Nalpas. (2024). Understand llm sizes | articles | web.dev — web.dev [[Accessed 08-12-2024]].
- Narang. (2024). Pathways language model (palm): Scaling to 540 billion parameters for breakthrough — research.google [[Accessed 08-12-2024]].
- NIST. (2015). Red Team - Glossary | CSRC — csrc.nist.gov [[Accessed 08-12-2024]].
- OpenAI. (2024a). Github - openai/simple-evals — github.com [[Accessed 08-12-2024]].
- OpenAI. (2024b). Hello gpt-4o [[Accessed 08-12-2024]].
- OpenAI. (2024c). Openai o1 systemcard [[Accessed 08-12-2024]].
- OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., . . . Zoph, B. (2023). Gpt-4 technical report. <https://doi.org/10.48550/ARXIV.2303.08774>
- Oracle. (2024). Chapter 8: Virtualbox networking [VirtualBox User Manual].
- Picus. (2023). What is adversary emulation? — picussecurity.com [[Accessed 08-12-2024]].
- Purba, M. D., Ghosh, A., Radford, B. J., & Chu, B. (2023). Software vulnerability detection using large language models. *2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 112–119.
- Scanlon, M., Breiting, F., Hargreaves, C., Hilgert, J.-N., & Sheppard, J. (2023). Chatgpt for digital forensic investigation: The good, the bad, and the unknown. *Forensic Science International: Digital Investigation*, 46, 301609. <https://doi.org/https://doi.org/10.1016/j.fsidi.2023.301609>

- Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., Gupta, A., Han, H., Schulhoff, S., Dulepet, P. S., Vidyadhara, S., Ki, D., Agrawal, S., Pham, C., Kroiz, G., Li, F., Tao, H., Srivastava, A., . . . Resnik, P. (2024). The prompt report: A systematic survey of prompting techniques. <https://arxiv.org/abs/2406.06608>
- Song, C., Ma, L., Zheng, J., Liao, J., Kuang, H., & Yang, L. (2024). Audit-llm: Multi-agent collaboration for log-based insider threat detection. *arXiv preprint arXiv:2408.08902*.
- Statcounter. (2024). Operating system market share worldwide | statcounter Global Stats — [gs.statcounter.com](https://gs.statcounter.com) [[Accessed 08-12-2024]].
- Stratosphere. (2015). Stratosphere laboratory datasets [Retrieved March 13, 2020, from <https://www.stratosphereips.org/datasets-overview>].
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. <https://doi.org/10.48550/ARXIV.1706.03762>
- Vellum. (2024). Llm leaderboard 2024 — [vellum.ai](https://vellum.ai) [[Accessed 08-12-2024]].
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.
- Xu, J., Yang, R., Huo, Y., Zhang, C., & He, P. (2024). Divlog: Log parsing with prompt enhanced in-context learning. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 1–12.
- Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.
- Zimmerman. (2023). MDwiki — [ericzimmerman.github.io](https://ericzimmerman.github.io) [[Accessed 08-12-2024]].

# Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis<sup>1</sup>

I ARTUR NIKITCHUK

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Advancing Cyber Threat Detection Through Host-Based Security Log Analysis with Large Language Models”, supervised by Prof. Hayretdin Bahsi
  - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
  - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons’ intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

12.01.2025

---

<sup>1</sup>The non-exclusive licence is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

## Appendix 2 - VM Orchestration and Attack Simulation

### Powershell Script

GITHUB LINK HERE

```
1
2 # Start the script block
3 $startTime = Get-Date
4 Set-Location "D:\Virtual Box\"
5
6 # Define the array of TTP IDs
7 $ttpArray = @("T1562.002-1", "T1562.002-2", "T1562.002-3", ...)
8
9 # VM management functions
10 function Start-TargetVM {
11     param (
12         [int]$Timeout = 300,
13         [int]$Interval = 5
14     )
15     .\VBoxManage.exe startvm VM-WIN10-TRGT --type headless
16     $elapsedTime = 0
17     while (-not (Test-VMReady)) {
18         if ($elapsedTime -ge $Timeout) {
19             Write-Host "Timed out waiting for the VM to boot." -
ForegroundColors Red
20             exit 1
21         }
22         Write-Host "    [?] Waiting for the VM to boot..." -
ForegroundColors Yellow
23         Start-Sleep -Seconds $Interval
24         $elapsedTime += $Interval
25     }
26     Write-Host "[+] Target Machine has fully booted" -ForegroundColor
Green
27 }
28
29 function Test-VMReady {
30     $result = .\VBoxManage.exe guestproperty get "VM-WIN10-TRGT" "/
VirtualBox/GuestInfo/OS/LoggedInUsers"
31     return ($result -notmatch "No value set!")
32 }
33
34 function Stop-TargetVM {
```



```

35 $vmState = Get-VMState
36 switch -Regex ($vmState) {
37     'running|paused|starting' {
38         Write-Host "[+] VM is $vmState. Powering off..." -
ForegroundColors Yellow
39         .\VBoxManage.exe controlvm VM-WIN10-TRGT poweroff
40     }
41     'saved' {
42         Write-Host "[+] VM is in saved state. Discarding..." -
ForegroundColors Yellow
43         .\VBoxManage.exe discardstate VM-WIN10-TRGT
44     }
45 }
46 Wait-For-VMState -TargetState 'poweroff'
47 Write-Host "[+] VM is powered off and ready to restore the snapshot
." -ForegroundColor Green
48 }
49
50 function Get-VMState {
51     return (.\VBoxManage.exe showvminfo "VM-WIN10-TRGT" --
machinereadable | Select-String "VMState=").ToString().Split('=')
[1].Trim(' ')
52 }
53
54 function Wait-For-VMState {
55     param (
56         [string]$TargetState,
57         [int]$Timeout = 60,
58         [int]$Interval = 3
59     )
60     $elapsedTime = 0
61     do {
62         Start-Sleep -Seconds $Interval
63         $currentState = Get-VMState
64         $elapsedTime += $Interval
65         if ($elapsedTime -ge $Timeout) {
66             Write-Host "Timed out waiting for VM state $TargetState" -
ForegroundColors Red
67             exit 1
68         }
69     } while ($currentState -ne $TargetState)
70 }
71
72 function New-RetryPSSession {
73     param (
74         [string]$ComputerName,
75         [PSCredential]$Credential,

```

```

76     [int]$MaxAttempts = 5,
77     [int]$RetryIntervalSeconds = 10
78 )
79 Add-Type -AssemblyName System.Windows.Forms
80 for ($attempt = 1; $attempt -le $MaxAttempts; $attempt++) {
81     try {
82         Write-Host "Attempt $attempt to establish remote session
... " -ForegroundColor Yellow
83         $session = New-PSSession -ComputerName $ComputerName -
Credential $Credential -ErrorAction Stop
84         Write-Host "Remote session established successfully." -
ForegroundColor Green
85         return $session
86     }
87     catch {
88         Write-Host "Failed to establish remote session (Attempt
$attempt of $MaxAttempts)" -ForegroundColor Red
89         Write-Host "Error: $_" -ForegroundColor Red
90         if ($attempt -lt $MaxAttempts) {
91             Write-Host "Retrying in $RetryIntervalSeconds seconds
..." -ForegroundColor Yellow
92             Start-Sleep -Seconds $RetryIntervalSeconds
93         } else {
94             Write-Host "Max attempts reached. Unable to establish
remote session." -ForegroundColor Red
95             # Create and show Windows Notification
96             $notification = New-Object System.Windows.Forms.
NotifyIcon
97             $notification.Icon = [System.Drawing.SystemIcons]::
Error
98             $notification.BalloonTipIcon = [System.Windows.Forms.
ToolTipIcon]::Error
99             $notification.BalloonTipTitle = "Remote Session
Connection Failed"
100            $notification.BalloonTipText = "Failed to establish a
remote session with $ComputerName after $MaxAttempts attempts."
101            $notification.Visible = $true
102            $notification.ShowBalloonTip(5000)
103            # Exit the script
104            exit
105        }
106    }
107 }
108 }
109
110 # Preliminary configurations
111 Enable-PSRemoting -SkipNetworkProfileCheck -Force

```

```

112 winrm set winrm/config/client/auth '@{Basic="true"}'
113 winrm set winrm/config/service/auth '@{Basic="true"}'
114 winrm set winrm/config/client '@{AllowUnencrypted="true"}'
115 winrm set winrm/config/service '@{AllowUnencrypted="true"}'
116 Set-NetFirewallRule -Name 'WINRM-HTTP-In-TCP' -RemoteAddress Any
117 Set-Item WSMAN:\localhost\Client\TrustedHosts -Value '*' -Force
118
119 Import-Module "C:\AtomicRedTeam\invoke-atomicredteam\Invoke-
    AtomicRedTeam.psd1" -Force
120 $PSDefaultParameterValues = @{"Invoke-AtomicTest:PathToAtomicsFolder"="
    C:\AtomicRedTeam\atomics"}
121
122 $targetComputer = "Server002"
123 $credential = New-Object System.Management.Automation.PSCredential ("
    admin_test", (ConvertTo-SecureString "123123" -AsPlainText -Force))
124 foreach ($ttp in $ttpArray) {
125     $startTime_ttp = Get-Date
126     $outputFolder = "D:\atomic_results\$ttp"
127     Write-Host "[*] Processing TTP: $ttp" -ForegroundColor Cyan
128     Start-TargetVM
129     Start-Sleep -Seconds 30
130     $session = New-RetryPSSession -ComputerName $targetComputer -
    Credential $credential
131     if ($null -eq $session) {
132         Write-Host "Skipping TTP $ttp due to connection failure." -
    ForegroundColor Red
133         continue
134     }
135     New-Item -Path $outputFolder -ItemType Directory -Force | Out-Null
136     Write-Host "Output folder created at $outputFolder" -
    ForegroundColor Green
137     Write-Host "[*] Executing Pre-req installation" -ForegroundColor
    Yellow
138     Invoke-AtomicTest $ttp -GetPrereqs -Session $session -
    ExecutionLogPath "$outputFolder\$ttp.csv" *>&1 | Tee-Object "
    $outputFolder\$ttp.txt" -Append
139     Write-Host "[*] Clearing pre-attack logs" -ForegroundColor Yellow
140     Invoke-Command -Session $session -ScriptBlock {
141         @("Microsoft-Windows-Sysmon/Operational", "Application", "
    System", "Security", "Windows PowerShell") | ForEach-Object {
142             Write-Host "Clearing $_ logs..."
143             wevtutil cl $_
144         }
145     }
146     Write-Host "[*] Executing TTP" -ForegroundColor Yellow
147     Invoke-AtomicTest $ttp -Session $session -ExecutionLogPath "
    $outputFolder\$ttp.csv" *>&1 | Tee-Object "$outputFolder\$ttp.txt"

```

```

-Append
148 Start-Sleep -Seconds 30
149 Write-Host "[*] Exporting logs" -ForegroundColor Yellow
150 @"Microsoft-Windows-Sysmon/Operational", "Application", "System",
"Security", "Windows PowerShell") | ForEach-Object {
151     $logName = $_
152     $outputFile = Join-Path $outputFolder "$ttp'_$($logName -
replace '/' , '_').evtx"
153     try {
154         Invoke-Command -Session $session -ScriptBlock {
155             param($LogName)
156             $tempPath = Join-Path $env:TEMP "TempLog.evtx"
157             wevtutil epl $LogName $tempPath
158             Get-Content -Path $tempPath -Raw -Encoding Byte
159             Remove-Item -Path $tempPath -Force
160         } -ArgumentList $logName | Set-Content -Path $outputFile -
Encoding Byte
161         Write-Host "Log $logName exported successfully" -
ForegroundColor Green
162     }
163     catch {
164         Write-Host "Error exporting $logName : $_" -ForegroundColor
Red
165     }
166 }
167 Remove-PSSession $session
168 Stop-TargetVM
169 Write-Host "[*] Restoring backup" -ForegroundColor Yellow
170 .\VBoxManage.exe snapshot VM-WIN10-TRGT restorecurrent
171 $executionTime_ttp = (Get-Date) - $startTime_ttp
172 Write-Host "TTP $ttp completed. Execution time: $executionTime_ttp"
-ForegroundColor Cyan
173 Write-Host "-----" -
ForegroundColor Cyan
174 }
175 $totalExecutionTime = (Get-Date) - $startTime
176 Write-Host "Total script execution time: $totalExecutionTime" -
ForegroundColor Green

```

Listing 1. PowerShell Script For Dataset Generation