

Abstract

The escalating volume and sophistication of cyber threats underscore the critical need for advanced Network Intrusion Detection Systems (NIDSs). Cybersecurity analysts, however, frequently experience “alert fatigue” due to the sheer number of alerts, highlighting a gap between current NIDS capabilities and practical operational needs. Recent advancements in Large Language Models (LLMs) present a potential avenue for enhancing NIDS efficacy. This thesis investigates the effectiveness of general-purpose LLMs in the multi-class classification of network traffic flows.

To address this, our methodology involved extending a state-of-the-art three-stage NIDS pipeline with a fourth stage powered by LLMs. We systematically evaluated three LLM families (DeepSeek-R1, Gemma-3, Qwen2.5-Coder) across various model sizes (1B to 32B parameters) using the CIC-IDS2017 dataset. The investigation encompassed three distinct input data representations (**raw**, statistically contextualized, and summarized-contextualized), five advanced prompting strategies (including few-shot, Chain-of-Thought, and Tree-of-Thought), and Low-Rank Adaptation (LoRA) fine-tuning. Weighted F_1 -score and balanced accuracy served as primary performance metrics to evaluate the performance on the unbalanced testing data, that results from passing through the baseline NIDS stages.

The findings indicate that the evaluated general-purpose LLMs, within the implemented framework, are not currently suitable for replacing or significantly enhancing established machine learning techniques for direct network flow classification. The LLM-augmented system did not achieve performance comparable to the baseline state-of-the-art NIDS and exhibited poor classification capability, often near random chance. Critical reliability issues, such as malformed outputs and single-class prediction biases, were prevalent and were not overcome by variations in model scale, input representation, prompting, or fine-tuning. This outcome, while negative, provides a valuable scientific contribution by highlighting current limitations and guiding future research. It suggests a fundamental misalignment between current general-purpose LLMs and the task of low-level network flow data classification. Future efforts should prioritize exploring hybrid architectures, where LLMs support Natural Language Processing (NLP)-centric auxiliary tasks in cybersecurity, or focus on developing novel data representations and LLM adaptation techniques more attuned to numerical data.

Contents

List of Figures	ix
List of Tables	x
List of Abbreviations	xi
List of Listings	xiii
1 Introduction	1
1.1 Objectives	3
1.2 Methodology Overview	3
1.3 Thesis Structure	5
2 Related Work	6
2.1 Intrusion Detection Systems Evolution	6
2.2 Large Language Models For Textual And Numerical Predictive Tasks	8
2.3 Research Gap	9
3 Background	11
3.1 Cybersecurity Foundations	11
3.1.1 Network Security Architecture and Key Protocols	12
Overview of Network Security Layers (OSI and TCP/IP)	12
Key Network Security Protocols	13
Intrusion Detection and Prevention Systems (IDPS)	16
3.1.2 Types of Network Attacks	18
Brute Force Attacks	19
Denial of Service Attacks	20
Heartbleed Attack	20
Web-Based Attacks	21
3.2 Fundamentals of Large Language Models	22
3.2.1 Transformer Architecture	26
Attention Mechanism	27
Transformer Architecture vs. Traditional Methods	30
Relevance to LLMs	31
3.2.2 Pre-training and Fine-tuning	32
3.2.3 Capabilities of LLMs	35
Natural Language Understanding and Generation	36
Information Retrieval and Summarization	37

	Coding Assistance and Automation	37
	Interdisciplinary Communication and Collaboration	38
	Reasoning	39
3.2.4	Limitations and Challenges	39
	Explainability	39
	Reproducibility	40
	Privacy Concerns	41
	Misinformation and Hallucinations	41
4	LLM Agents in Intrusion Detection	42
4.1	Study Methodology	42
4.1.1	Experimental Design	43
	Performance Metrics	44
	Formal Hypotheses	46
	Model Adaptation: Input Format, Instruction Tuning, and Fine-tuning	46
	Auxiliary Experiments	50
4.1.2	CIC-IDS2017 Dataset	52
	Dataset Composition.	53
	Feature Extraction and Labeling	53
4.1.3	Baseline: Multi-Stage Hierarchical IDS by Verkerken et al. . .	54
	Proposed Methodology	54
	Experimental Evaluation and Results	56
	Reimplementation and Reproduced Results	57
4.1.4	Limitations and Assumptions	59
4.2	System Architecture and Implementation	61
4.2.1	Core Components	62
4.2.2	Dataset Splitting and Stage Distribution	65
4.2.3	Hardware Setup	68
4.3	Performance Evaluation	68
4.3.1	System Performance by Model Variant	70
4.3.2	System Performance by Prompting Technique	80
4.3.3	System Performance Under Fine-Tuning	83
4.3.4	Auxiliary Analyses	85
5	Discussion	92
5.1	Analysis of the Results	92
5.1.1	Overall Classification Performance	92
5.1.2	Model Variant Analysis	94
5.1.3	Prompt Engineering and Input Format	95
5.1.4	Fine-Tuning Effects	96
5.2	Answering the Research Questions	96
5.3	Implications for Practice and Research	97

6 Conclusion and Future Directions	100
Appendix A. Further Details on Configurations, Datasets, Functions and Prompts	115
Appendix B. Full Results	132
Appendix C. Illustrative Examples of AI Assistance in the Writing Process	140

List of Figures

1	Overview of our proposed multi-stage IDS with LLM integration . . .	4
2	Cybersecurity Awareness Overview model	11
3	The OSI model and the TCP/IP model in contrast	12
4	IDS Taxonomy	16
5	MITRE ATT&CK stages	19
6	Transformer Architecture	27
7	Attention Mechanism	28
8	Baseline System Design for the Multi-Stage IDS	56
9	Reported Baseline Results for the Full System	57
10	Reproduction Confusion Matrix for the Full System	58
11	Stages of our custom multi-stage Intrusion Detection System (IDS) .	62
12	Component diagram of the classifier implementation	63
13	Component diagram of the aggregator implementation	64
14	Data sampling and distribution	65
15	Partitioning of the CIC-IDS2017 dataset	67
16	End-to-End Model Family Evaluation: Weighted F_1 -Score	72
17	Stage-Specific Model Family Evaluation: Weighted F_1 -Score	73
18	End-to-End Model Family Evaluation: Balanced Accuracy	75
19	Stage-Specific Model Family Evaluation: Balanced Accuracy	76
20	End-to-End Model Family Evaluation: Best Configuration (F1) . . .	77
21	Stage-Specific Model Family Evaluation: Best Configuration (F1) . .	78
22	Stage-Specific Model Family Evaluation: Best Configuration (b. Acc.)	80
23	Valid Stage-Specific Model Family Evaluation: Best Configuration (b. Acc.)	81
24	Stage-Specific Prompting Technique Evaluation: Weighted F_1 -Score .	84
25	Stage-Specific Prompting Technique Evaluation: Best Configuration (F1)	84
26	Stage-Specific Fine-Tuning Evaluation: Best Configuration (F1) . . .	86
27	End-to-End Auxiliary Evaluation: Balanced Test Sets Confusion Matrices	88
28	Stage-Specific Auxiliary Evaluation: Percentage of Malformatted Out- puts	89
29	Label distribution of the CIC-IDS2017 dataset	131

List of Tables

1	TCP/IP Model Security Layers with Protocols and Common Cyber-security Tools	14
2	Definitions and domains of symbols used in NLP	26
3	Prompting Techniques Used	49
4	Comparison of classification performance in baseline and reproduced results	58
5	Hardware and software specifications of the experimental system . . .	68
6	Figure Caption Prefix Guide	69
7	Figure Map by Scope and Ablation Variable	70
8	Stage-Specific Fine-Tuning Evaluation: Full Results	85
9	End-to-End Auxiliary Evaluation: Balanced (Full) Test Set Results .	87
10	Stage-Specific Auxiliary Evaluation: Stratified Random Guessing Baseline Performance	90
11	Model Configurations	115
12	Network Traffic Features	116
13	End-to-End Model Family and Input Format Evaluation: All Classification Metrics	132
14	Stage-Specific Model Family and Input Format Evaluation: All Classification Metrics	134
15	Valid Stage-Specific Model Family and Input Format Evaluation: All Classification Metrics	135
16	Stage-Specific Prompting Technique Evaluation: All Classification Metrics	138

List of Listings

1	Brief prompt example	4
2	Defined format for passing raw input data	47
3	Defined format for the contextualization function	48
4	Defined format for the summarization function	48
5	Stage-Specific Input Format: Prompting Example	71
6	Stage-Specific Input Format and Prompting Technique Evaluation: Example with ToT-V Prompting	83
7	Example for a raw prompt without prompting technique	122
8	Example for a ctx prompt without prompting technique	125
9	Example for a sumctx prompt without prompting technique	128
10	Full example for a raw query with ToT-V prompting	130
11	Example of AI assistance in the writing process: Grammar, Spelling and Punctuation	140
12	Example of AI assistance in the writing process: Translation	141
13	Example of AI assistance in the writing process: Tone & Style	141

1 Introduction

Network Intrusions are an all too common occurrence in the corporate world today with cybercrime growing rapidly. Between 2024 and 2029, global cybercrime costs are expected to rise by \$6.4 trillion, an increase of 69.41% [42]. A first step to combat this rise in crime is to detect attacks as early as possible. By advancing the research into new tools and techniques for Network Intrusion Detection System (NIDS), we can lower false positive rates and increase the automation of the detection of attacks, while reducing the need for human intervention.

While current research solutions exhibit high accuracy, in practice many cybersecurity analysts are struggling to keep up with the volume of alerts, completing only about half of their expected daily workload [17]. This divide between research and practice might have two causes: either the classification performance of the current State-of-the-Art (SOTA) models is not high enough, especially in a multi-class setting with highly unbalanced data, or the tools do not convey the necessary information to the analysts effectively. Thus, increasing either the effectiveness of current models or the usability of the available tools could help to bridge this gap.

With the global adoption of Large Language Models (LLMs) for a variety of tasks it comes as no surprise that LLMs have already been used in cybersecurity, as seen in tools like Microsoft Copilot for Security¹. LLMs can process alert data in natural language, providing human-readable summaries and contextual threat intelligence. This can reduce the time analysts spend researching each alert, thus directly alleviating alert fatigue and streamlining the triage process. However, to the best of our knowledge, there are no well-evaluated open-source solutions that test the effectiveness of LLMs in the context of Network Intrusion Detection, even though research in the broader area of LLMs in cybersecurity is beginning to gain momentum [6, 25, 31]. Interest in decoder-only LLMs for security, for example, has exploded (92 papers by 2024 [102]), yet their efficacy on flow-based NIDS remains largely untested. More generally, the efficacy of LLMs in processing and reasoning over primarily numerical and tabular data is largely untested. Unlike unstructured log messages where LLMs excel, flow records are predominantly composed of numeric features with no inherent linguistic semantics, making their interpretation by LLMs an open challenge.

This research investigated the first steps in developing an open-source LLM-based IDS for Security Operations Centers (SOCs), by taking a closer look at the effectiveness of LLMs in the context of NIDS. Through a comprehensive implementation

¹<https://www.microsoft.com/security/business/ai-machine-learning/microsoft-copilot-security>, accessed July 9, 2025

and evaluation, this thesis tested the effectiveness of LLMs in Network Intrusion Detection, ultimately supporting them to better combat the ever-evolving cyber threats, including novel, unknown attacks. This study is significant because even a negative result regarding LLMs’s direct applicability to raw numerical flow data provides a valuable scientific contribution by highlighting limitations and guiding future research towards hybrid approaches or more suitable data representations.

NIDS-related Security Operations Center Challenges. Today’s security analysts in SOC’s are increasingly confronted with the challenge of discerning legitimate threats from a deluge of false positives generated by automated alert systems [17], while more sophisticated attacks can go unnoticed [27].

This situation, often termed “alert fatigue”, not only demands a significant portion of their time but also contributes to professional burnout [3], given that many IDS alerts stem from benign activities flagged as threats [85]. A more detailed review of these challenges, including the nuanced distinction between “false alarms” and “benign triggers” [3], is provided in Chapter 2. The core issue remains the need for more effective tools and processes to support analysts.

Large Language Model Opportunities. LLMs have shown unprecedented success across diverse fields [56], primarily due to the transformer architecture [91], which excels in capturing complex patterns and long-range dependencies in sequential data. While their application to natural language is well-established, their potential for structured, numerical data, such as network flows, is an active area of investigation. The ability of transformers to detect subtle and complex patterns, without extensive domain-specific feature engineering [48], is crucial for enhancing NIDS performance against sophisticated attacks. For instance, LLMs have shown promise in security-centric tasks like log parsing [62, 86].

If we can effectively incorporate structured numerical information into LLM-based systems while preserving the natural language interface, we may unlock powerful synergies and alleviate the burdens placed on security analysts by enabling them to integrate their prior knowledge into predictive models through natural language. On one side of the spectrum, rule-based or classical Machine Learning (ML) approaches are often explainable but rigid. On the other, LLMs are highly flexible and capable of processing large volumes of data, yet potentially unreliable or opaque in their reasoning. By combining these paradigms, it becomes possible to leverage both the interpretability of traditional methods and the usability and scalability of LLMs. Motivated by this interplay between interpretability, usability, and computational capacity, we see the need for a thorough investigation of the current state of the predictive capabilities of LLMs on intrusion detection data.

1.1 Objectives

The primary focus of this study was to support security analysts in distinguishing between real threats and false alarms. This study tests whether LLM-enhanced systems can classify malicious network traffic at least as well as a state-of-the-art multistage IDS. To evaluate this, the study compared the efficacy of LLM-based classification with the current state of the art in NIDS. The findings of this study are expected to contribute to the ongoing discourse on the role of AI in enhancing cybersecurity practices and provide practical insights for SOCs. In summary, the research addresses the following research questions:

RQ1 Are LLMs effective for classifying malicious network traffic?

- (a) Do LLMs improve classification performance compared to a multi-stage SOTA baseline model?
- (b) Do LLMs perform better than random guessing in classifying malicious network traffic?

To rigorously test these questions, the following null hypotheses were formulated:

$$\begin{aligned} H_0^1 : \mathcal{P}_{\text{LLM-IDS}} &\leq \mathcal{P}_{\text{baseline}} \\ H_0^2 : \mathcal{P}_{\text{LLM-IDS}} &\leq \mathcal{P}_{\text{random}} \end{aligned}$$

where \mathcal{P} denotes the classification performance.

The practical results that test our hypotheses enable us to also discuss the more general question if LLMs can add value for security analysts in the identification and response to cyber threats.

1.2 Methodology Overview

This section provides a high-level overview of our methodological approach. For a detailed account of the experimental design, dataset usage, model adaptation procedures, and evaluation strategy, please refer to Section 4.1.

Our methodology involves extending the three-stage intrusion detection pipeline by Verkerken et al. [92] with a novel fourth stage powered by LLMs. Within this new stage, we systematically explore the impact of various LLM configurations, including different model families and sizes, diverse input data representations (raw, contextualized, and context-summarized), and multiple adaptation techniques such as advanced prompting strategies and Low-Rank Adaptation (LoRA) fine-tuning. We evaluate this methodology using the CIC-IDS2017 dataset, employing a hold-out test set of 59,435 flows. After these flows are processed through the initial three stages of the pipeline, an average of about 500 samples remain for evaluation by the LLM-based Stage 4. Weighted F_1 -score and balanced accuracy serve as our primary performance metrics. Figure 1 provides an overview of the proposed multi-stage

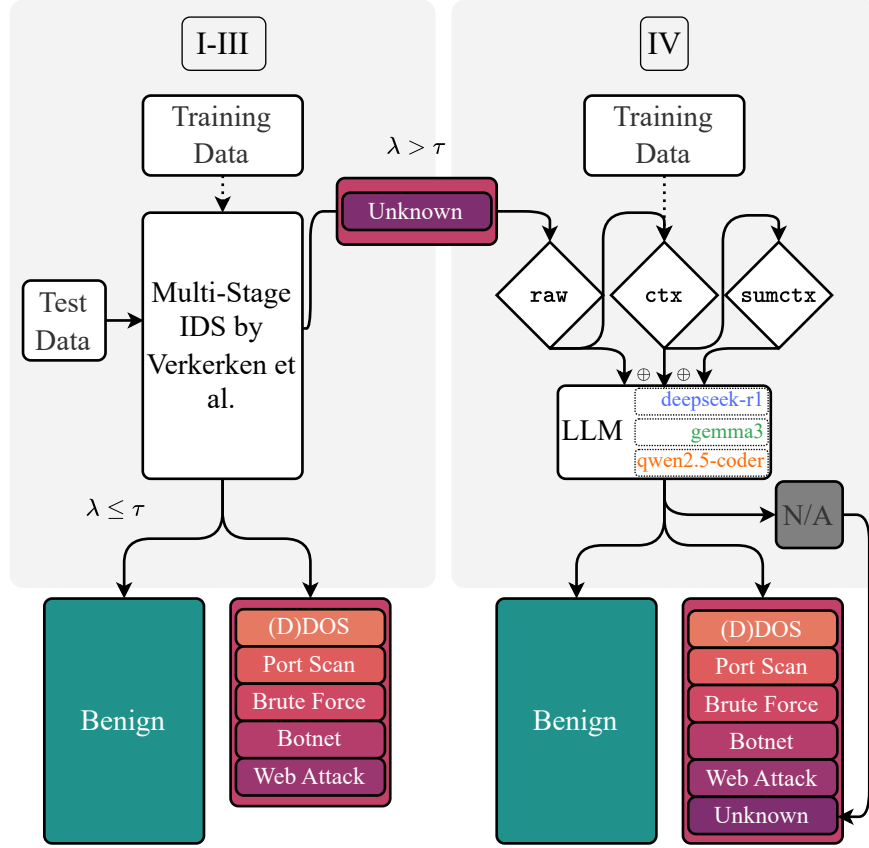


Figure 1: Overview of our proposed multi-stage IDS with LLM integration. The first three stages are the same as in the work by Verkerken et al. [92]. Test samples classified with certainty λ that pass the baseline threshold τ are passed to the LLM for final classification.

IDS with LLM integration, with Listing 1 providing an example of a prompt used to classify a network flow.

Query
[[## flow_parameters ##]] 6, 8686549, 2, 0, 12, 0, 6, 6, ...
LLM Output
[[## Label ##]] (D)DOS [[## completed ##]]

Listing 1: Abbreviated prompt example that demonstrates one method of our overall approach in letting an LLM classify network traffic. We omitted System Message, Reasoning and further details for brevity.

We tested three different LLM families (with single model sizes ranging from 1B to 32B parameters) (i) **deepseek-r1** (ii) **gemma3** (iii) **qwen2.5-coder** on three different input representations (i) **raw** (comma-separated values) (ii) **ctx** (statistically contextualized values) (iii) **sumctx** (summarized contextualized values) with five different prompting paradigms (i) - (no improvement) (ii) **inst** (automatic instruction tuning) (iii) **6** (six-shot prompting) (iv) **CoT-6** (six-shot Chain-of-Thought (CoT) prompting) (v) **ToT-V** (Tree-of-Thought prompting with five experts).

In summary, this thesis makes the following primary contributions:

- The first evaluation of three LLM families on flow-level intrusion detection classification.
- A systematic comparison of three distinct input representations, five prompting paradigms, and the impact of LoRA fine-tuning for this task.

1.3 Thesis Structure

This thesis is organized to provide a comprehensive and critical analysis of the application of LLMs in network intrusion detection.

Chapter 2 presents an exhaustive review of the relevant literature, tracing the evolution of IDSs, the integration of deep learning and transformer-based models in cybersecurity, and the emerging role of LLMs for both textual and numerical predictive tasks.

Chapter 3 revisits foundational concepts in cybersecurity and LLMs, including network security architecture, key protocols, types of network attacks, and the core principles underlying modern language models.

Chapter 4 details the methodology and implementation of the proposed LLM-augmented IDS. It describes the experimental setup, the integration of LLMs into a multi-stage detection pipeline, the design of input representations and prompting strategies, and the evaluation framework. The chapter concludes with a thorough performance analysis of various LLM configurations and adaptation techniques.

Chapter 5 provides a critical discussion and interpretation of the experimental results. It analyzes the classification performance of LLM-based systems, examines the impact of model variants, input formats, and prompt engineering, and addresses the research questions posed at the outset. The broader implications for both practical cybersecurity operations and future research are also discussed.

Chapter 6 summarizes the key findings of the thesis, highlighting the main conclusions regarding the suitability of LLMs for flow-based intrusion detection.

The appendices, Appendix A, Appendix B, and Appendix C, provide supplementary material. Appendix A contains further details on experimental configurations, datasets, and prompt examples. Appendix B presents comprehensive results and performance metrics for all evaluated model configurations. Appendix C documents illustrative examples of AI assistance in the writing process.