Technology
Arts Sciences
TH Köln

Evaluation of Flow-Based Network Traffic Analysis with LLMs for Intrusion Detection

Research Project Report

Master of Science

in the degree program Communication Systems and Networks

at the Faculty of Information, Media and Electrical Engineering

at Technische Hochschule Köln (Cologne University of Applied Sciences)

Submitted by:	Md Biplob Hosen
Matrikel-Nr.:	1114 6024
E-mail:	md_biplob.hosen@smail.th-koeln.de

Submitted to: Prof. Dr. Andreas Grebe

Köln, 26.08.2026

Abstract

The research project examines the application of a Large Language Model (LLM) to flow-based network traffic analysis by comparing its findings with those of Suricata, a traditional rule-based Intrusion Detection System (IDS). In order to do this, it makes use of five days from the CIC-IDS2017 dataset and designs an experimental workflow that includes bias control, so that the information from Suricata-generated alerts and the baseline labels is not available to the LLM when it is performing inference.

This project used four SQLite databases to separate the raw Suricata EVE events, the reconstructed Suricata baseline data, the LLM predictions, and the final comparison results. A two-stage process based on the meta-llama-3.1-8b-instruct model hosted on KISSKI was put in place. In the first stage all the reconstructed flows were broadly categorised as benign, noisy, suspicious or malicious, and in the second stage flows that had been initially classified as suspicious or malicious were analysed in more depth. Before accepting any model predictions, structured validation, canonicalization, detection of missing flows, and retry procedures were carried out.

In the five pilots, a total of 1,967,638 flows were processed. In Stage-1, 22,455 flows were chosen for further examination in Stage-2. Following a more in-depth analysis, 654 flows were found to be suspicious and 410 were identified as malicious. The final LLM result included 1,930,912 benign, 35,662 noisy, 654 suspicious, and 410 malicious predictions. The agreement between Suricata and the LLM in their triage decisions differed greatly among the pilots, ranging from 53.76% to 85.35%. A major cause of disagreement was that many flows which Suricata had marked as noisy or suspicious were classified as benign by the LLM.

The findings show that the LLM working within the bias-controlled workflow did not merely replicate the Suricata baseline but instead arrived at different flow-level interpretations. Yet since Suricata is used as a point of comparison rather than as the verified ground truth, the results indicate the areas in which the two methods agree and those in which they disagree rather than providing a direct measure of classification accuracy.

Keywords: LLM, IDS, Suricata, Network Traffic Analysis, CIC-IDS2017, Flow-Based Analysis.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Background

Network security monitoring of modern computer networks, especially enterprise networks, is very important from a cybersecurity perspective. For this purpose, companies use Intrusion Detection Systems (IDS). These IDSs monitor network traffic flows to detect unusual or abnormal behaviour, identify attacks, and help security systems and security analysts support incident response. IDS detect illegal access attempts, impersonation, penetration by legitimate users, denial-of-service (DoS), and other malicious behaviours via Anomaly-Based Detection (AD), Stateful Protocol Analysis (SPA), and Signature-Based Detection (SD) [1].

Suricata is an open-source IDS, IPS and network security monitoring engine [2]. It can use existing free rulesets [3], and analyse protocols such as HTTP, HTTP/2, TLS, DNS, and SMB [4]. Suricata can provide protocol-specific information, produce structured EVE JSON logs [5] and generate alerts via syslog [6].

Rule-based IDSs can detect abnormal traffic flows and attack patterns; however, analysts must interpret additional information when traffic is noisy or ambiguous [1]. Network flows may represent ambiguous, noisy, or actual attack activity. In this study, flows are divided into four security categories: benign, noisy, suspicious, and malicious. Security analysts are required to classify flows into these categories using available protocol-level evidence, which demands significant effort. This context presents an opportunity to evaluate whether Large Language Models can assist with structured analysis of evidence at the network-flow and protocol levels.

Large Language Models are capable of processing structured text, extracting patterns, and generating outputs in structured JSON or plain language. LLMs can support flow-level reasoning, log interpretation, alert summarization, and analyst assistance in cybersecurity [7], [8]. To obtain meaningful results from LLMs, experimental design must prevent the influence of existing IDS decisions [7], [9]. If the LLM receives decision information such as triage labels, attack-class labels, severity, alerts, or signatures from the IDS (Suricata), this information could bias the LLM and reduce the independence and reliability of its analysis. Therefore, bias control is a critical requirement in this study.

1.2 Problem Statement

Conventional IDS tools such as Suricata generate structured alerts and event logs based on predefined rules and protocol analysis [3], [4], [5]. However, these outputs can be numerous, noisy, and difficult to interpret at scale [10]. LLMs, by contrast, offer new possibilities for processing structured security data [7], [8]. Still, their use in IDS research can be biased if the model is exposed to existing IDS labels or alert metadata.

This research project investigates the design and evaluation of a bias-controlled input for an LLM-based intrusion analysis workflow using Suricata EVE logs. In this study, the LLM operates on bias-controlled inputs, with Suricata-defined alerts, signatures, baseline labels, and attack class information filtered and excluded. The LLM generates predictions, which are subsequently validated through a dedicated validation script. The validated LLM predictions are then compared with the Suricata baseline.

1.3 Research Aim

The aim of this research is to investigate and compare the performance of the Suricata baseline with bias-controlled LLM predictions using the CIC-IDS2017 dataset.

1.4 Research Objectives

The objectives of this project are as follows:

1. Generating Suricata EVE JSON logs from multiple days (Monday to Friday) of CIC-IDS2017 as pilots.
2. Building four structured databases for each pilot to store raw events, Suricata baseline outputs, LLM prediction results, and final comparison results.
3. Constructing bias-controlled LLM inputs by excluding Suricata alerts, signatures, severities, flow labels, and attack class information.
4. Splitting input texts into chunks for sending to the LLM.
5. Triaging all flows using the LLM in Stage-1.
6. Reviewing the suspicious or malicious candidates from Stage-1 outputs in Stage-2.
7. Retrying incomplete, missing, and malformed LLM-processed flows of Stage-1 and Stage-2.
8. Canonicalizing and validating Stage-1 and Stage-2 outputs before inserting into the LLM prediction database.
9. Analyzing triage-label and attack-class distribution of Suricata baseline labels and LLM prediction results.
10. Comparing triage-label and attack-class agreement rates across five pilots between Suricata baseline labels and LLM prediction results.

1.5 Research Questions

The study addresses the following research questions:

- How can LLM input be constructed as bias-controlled to exclude decision information from Suricata alerts and labels?
- What is the agreement rate of the triage label and attack class between Suricata baseline results and bias-controlled LLM prediction results?

- How does the LLM classify flows that Suricata marks as noisy, suspicious, or malicious?
- What are the limitations of using LLMs for flow-level intrusion detection and in reasoning from protocol-level evidence?

1.6 Scope of the Study

This study utilizes five days of PCAP data from the CIC-IDS2017 dataset, which was developed by a team at the Canadian Institute of Cybersecurity and contains both benign traffic and common attack scenarios for intrusion detection evaluation [11], [12]. Suricata-generated EVE JSON logs, designated as Pilot4, Pilot5, Pilot6, Pilot7, and Pilot8, capture data from Wednesday, Monday, Tuesday, Thursday, and Friday, respectively. The official CIC-IDS2017 labels were not mapped to the Suricata-reconstructed flow records identified by flow_id, and the LLM does not use the full packet payload as input. The study compares bias-controlled LLM predictions with an independent Suricata baseline, rather than verified ground truth. Analyses include triage-label and attack-class distributions across pilots, the number and rate of triage-label agreement flows, and the rate of attack-class agreement between the Suricata baseline and LLM predictions. Pilots 1-3, which were used as learning experiments on the CIC-DDoS2019 dataset to develop the project workflow, are not included in the final evaluation.

2 Literature Review

2.1 Intrusion Detection Systems

Intrusion Detection Systems are security mechanisms that monitor computer systems or networks and can analyze and identify intrusions such as suspicious or malicious activity [1]. NIDS monitor network traffic and detect suspicious activities [13]. IDS detection methods can be signature-based, anomaly-based, or stateful protocol analysis-based [1].

Signature-based (SD) detection is a pattern- or string-dependent method used to detect known attacks; anomaly-based (AD) detection is a behavior-dependent method that monitors and compares expected behavior and normal activities to detect unusual activity; and stateful protocol analysis (SPA) detection is a specification-based method that depends on vendor-specific profiles [1]. SD and AD have concerns about unknown attacks and false-positive alerts, respectively [1]. Signature-based IDS is also called rule-based IDS [14].

2.2 Suricata as a Rule-Based IDS

Suricata is an open-source IDS and network monitoring tool [2]. It supports rule-based detection, protocol detection, flow tracking, and structured event output [3], [4], [5]. Suricata generates EVE JSON logs for later analysis [5]. Suricata was used as the IDS in this study, and its independent baseline output was compared with LLM prediction results instead of the official CIC-IDS2017 ground truth.

2.3 CIC-IDS2017 Dataset

CIC-IDS2017 is an intrusion detection evaluation dataset developed by a research team at Canadian Institute for Cybersecurity. It contains benign traffic and common attacks in a realistic network environment [11], [12]. This project uses CIC-IDS2017 pilot datasets and processes Suricata EVE JSON logs derived from those traffic captures. The selected pilots represent different days and traffic conditions, allowing cross-pilot comparison [12].

2.4 Large Language Models in Security Analysis

Large Language Models (LLMs) can process structured text and produce structured outputs, such as JSON. In cybersecurity, they can be used for log analysis, explaining alerts, summarising events, and supporting analysts [7], [8]. However, it is necessary to design the prompt carefully, construct bias-controlled inputs, and validate outputs for analyzing LLM-based security events [7], [9] .

A major risk is so-called label leakage. If a large language model is given the alert signature, its severity, the attack class, or a label assigned by an Intrusion Detection System (IDS), it may repeat this information rather than independently analyzing the traffic evidence. Therefore, it is essential to construct the inputs to control for bias.

2.5 Research Gap

Many IDS experiments focus on machine-learning classifiers using labeled datasets [8]. However, this project investigates a different problem: comparing a rule-based IDS baseline with a bias-controlled LLM interpretation of flow-level evidence. According to the reviewed literature, limited attention has been given to constructing bias-controlled LLM inputs and building structured, reproducible workflows that strictly separate bias-controlled LLM inputs from conventional IDS baseline outputs and compare only after validating LLM predictions [7], [8].

This project designed a workflow that uses four separate databases and a two-stage LLM analysis process.

3 Methodology

3.1 Research Design

This research uses an experimental workflow. The project uses PCAP files from CIC-IDS2017 to generate Suricata EVE JSON logs. The experimental pipeline processes the Suricata EVE JSON logs into structured raw events and stores them in a separate database. The Suricata baseline output and LLM predictions are also stored separately. The LLM receives only selected flow and protocol-level details. After LLM prediction and validation, the LLM prediction results are compared with the Suricata baseline outputs.

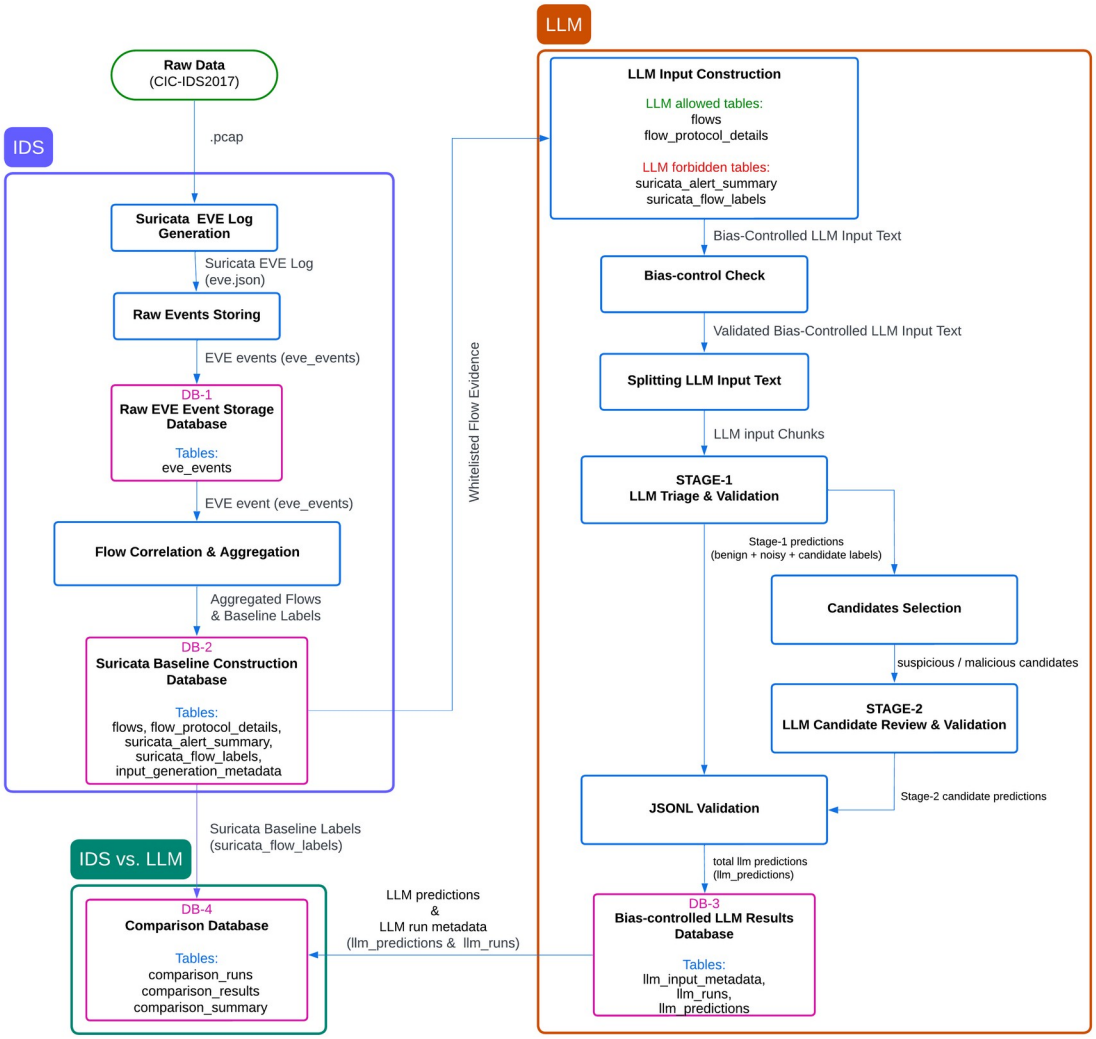

Figure 3.1: Workflow of Suricata Baseline and Bias-Controlled LLM-Based Intrusion Analysis

The figure above (Figure 3.1) shows the experimental workflow for Suricata baseline and bias-controlled LLM-based intrusion analysis in this research project. The workflow has been divided into four layers: the Raw Data source, the IDS pipeline, the LLM pipeline, and the IDS-vs-LLM comparison pipeline. The Raw Data source is the data source (consisting of PCAP files) from CIC-IDS2017. Those raw packet captures are transformed into Suricata EVE JSON logs [5], and those EVE events are stored in DB-1 (Raw EVE Event Storage Database, Table 3.2) in the IDS pipeline. The logs are correlated and aggregated into flows and stored in DB-2 (Suricata Baseline Construction Database, Table 3.2) in the same IDS pipeline, which contains flow identity, flow-level protocol details, Suricata alert summaries, Suricata baseline labels, and input generation metadata. In the first step of the LLM pipeline, the LLM input construction mechanism constructs bias-controlled LLM inputs using whitelisted flow evidence and avoids Suricata baseline decision information such as Suricata alert summaries, Suricata flow labels, categories, and severity-related information. The LLM pipeline checks and splits the constructed LLM inputs before LLM analysis. The LLM analysis is accomplished in two stages: Stage-1 and Stage-2. In Stage-1, flows are triaged; in Stage-2, suspicious or malicious candidates are reviewed for attack classification, confidence, evidence, and reasoning. After analyzing those flows, a validation mechanism validates the predictions to check for missing or duplicate results, flow_id, canonicalization, etc. In the last step of the LLM pipeline, the validated LLM predictions are stored in DB-3 (Bias-controlled LLM Results Database, Table 3.2). Finally, in the comparison pipeline, Suricata baseline labels and final LLM predictions are compared and stored in DB-4 (Comparison Database, Table 3.2).

3.2 Dataset Selection

This research project uses five pilot datasets from CIC-IDS2017:

Pilot	**CIC-IDS2017 Day**
Pilot4	Wednesday
Pilot5	Monday
Pilot6	Tuesday
Pilot7	Thursday
Pilot8	Friday

Table 3.1: Experimental Pilot Names and Corresponding Dataset Days from CIC-IDS2017

This study used five experimental pilot names with corresponding dataset days from CIC-IDS2017. These dataset days have different traffic sizes and characteristics [11], [12], providing an opportunity to compare cross-pilot triage-label and attack-class distributions between Suricata baseline results and LLM prediction results.

3.3 Database Architecture

In this project, separate databases serve individual purposes. Every pilot has a similar set of four databases. This separation was essential for event-level raw data storage, flow-level aggregated data storage, bias-control LLM input construction, preventing mixing Suricata decision information with the LLM pipeline by whitelisting, and comparing Suricata baseline outputs with LLM prediction results. Furthermore, this separation also provides traceability and reproducibility throughout the experimental pipeline and pilots.

Database	Purpose
DB-1	Stores raw Suricata EVE events
DB-2	Stores Suricata baseline flows, protocol details, alert summaries, Suricata flow labels, and input generation metadata.
DB-3	Stores LLM input metadata, LLM runs and LLM predictions
DB-4	Stores final Suricata-vs-LLM comparison results

Table 3.2: Names of databases and their purposes

This project has four SQLite databases per pilot: DB-1, DB-2, DB-3, and DB-4. DB-1 and DB-2 belong to the IDS pipeline and store EVE events and aggregated flows, respectively. On the other hand, DB-3 stores LLM prediction results, and DB-4 stores comparison results of Suricata baseline outputs and LLM prediction outputs.

3.4 DB-1: Raw EVE Event Storage

In this experiment, Suricata-generated event logs are stored in DB-1. Later, DB-1 provides raw events so DB-2 can aggregate and correlate flows. DB-1 is an SQLite database with a single table called eve_events. The eve_events table contains several event types, including flow, alert, anomaly, file information, and protocol-specific information such as application protocol names like DNS, TLS, HTTP, and HTTP2 [5].

DB-1 keeps raw events as evidence for future traceability and reproducibility before any aggregation or processing by the LLM pipeline. Each pilot has its own DB-1 database, which preserves the corresponding source day from the official CIC-IDS2017. DB-1 does not store aggregated flow labels, LLM prediction results, or comparison results of Suricata baseline and LLM predictions.

3.5 DB-2: The construction of the Suricata baseline

DB-2 builds and stores the Suricata baseline at the flow level, and aggregates the event-level data collected from DB-1 using the Suricata flow identifier so that two or

more events from the same network flow can be expressed as a single flow-level record.

The database consists of five tables: flows, flow_protocol_details, suricata_alert_summary, suricata_flow_labels, and input_generation_metadata. The flows table includes the reconstructed flow information; the flow_protocol_details table contains protocol-specific details; the suricata_alert_summary table contains summaries of Suricata alerts and anomalies; the suricata_flow_labels table contains the flow labels derived by Suricata; and the input_generation_metadata table holds the input generation metadata which is used for tracing the downstream LLM input generation. The alert and anomaly events linked to each flow are each summarised separately in order that it may assess the presence of Suricata detections without having to process the raw EVE event records repeatedly.

Table	Purpose
flows	Stores one row per flow
flow_protocol_details	Stores protocol-specific details linked to flows
suricata_alert_summary	Stores alert/anomaly summaries
suricata_flow_labels	Stores Suricata baseline triage labels and attack classes
input_generation_metadata	Stores metadata related to input preparation and generation of downstream LLM input

Table 3.3: Structure and Purpose of Tables in DB-2

The primary task of DB-2 is to store the reconstructed flow-level Suricata baseline and the metadata required to trace subsequent LLM input generation. The input_generation_metadata DB-2 table links each flow to its generated input representation through information such as the input file, chunk file, line range, prompt version, input hash, and token estimate. It does not contain LLM predictions or LLM-derived classifications. The LLM process ignores Suricata-defined decision information, such as Suricata-derived alert signatures, categories, severities, and baseline labels.

However, the Suricata flow-level baseline is compared with the final LLM prediction results in the final steps. In this study, Suricata’s detection results have been used as a baseline rather than as a ground-truth reference for comparison.

3.6 The construction Bias-Controlled LLM Input

The LLM pipeline created the LLM input from DB-2 using allowed flow evidence. Some selected flow-level and protocol-level evidence from the flows and flow_protocol_details tables is allowed for LLM analysis. This restriction allows the LLM

to investigate network-flow evidence without access to Suricata-defined decision information.

Suricata-defined information may reveal the baseline classification or could influence the LLM prediction results. For this reason, these are ignored in the LLM input. Those forbidden fields are alert signatures, alert categories, severity values, Suricata flow labels, and other baseline-derived classification information. Therefore, suricata_alert_summary and suricata_flow_labels are not used as evidence during LLM input construction.

The input_generation_metadata table in DB-2 is used only to trace the generated input. It records information such as the associated flow identifier, input and chunk files, line ranges, prompt version, input hash, and token estimate. These metadata fields are not supplied to the LLM as network evidence.

Input source	**LLM access**	**Purpose**
flows	Allowed - selected fields only	Flow-level traffic evidence
flow_protocol_details	Allowed - selected fields only	Protocol-specific evidence
suricata_alert_summary	Forbidden	Contains Suricata detection information
suricata_flow_labels	Forbidden	Contains Suricata baseline labels
input_generation_metadata	Not used as evidence	Traceability of input generation

Table 3.4: Allowed and Forbidden DB-2 Sources for Bias-Controlled LLM Input Construction

The bias-controlled LLM input is constructed as a compact TXT file, which then goes through a bias-control check mechanism before splitting.

3.7 Bias-Control Check

After the step of constructing the input for the LLM, there is a bias-control check which ensures that any information excluded by the input-construction mechanism has not been included in the evidence of the LLM input. This bias-control check functions as a validation boundary between the IDS pipeline and the LLM pipeline.

The bias-control check verifies that Suricata-derived decision information, including alert signatures, alert categories, severity values, Suricata baseline labels, and other fields that could influence LLM prediction results, is excluded. Previous LLM prediction

results are also excluded so that new prediction results cannot be influenced by previous predictions.

Only flow-level and protocol-level evidence selected through the whitelist may pass this stage. If the input satisfies the bias-control requirements, it is accepted as validated bias-controlled LLM input text and can proceed to splitting into chunks and Stage-1 analysis. The validation process also generates a bias-control report that provides an auditable record of the check.

3.8 Stage-1 LLM Triage

In this stage, the LLM starts analyzing the split chunks after the input is validated through the bias-control process. In the first step of Stage-1, the desired chunk is sent to the LLM with a predefined system prompt. Stage-1 sends only the permitted bias-controlled evidence but no Suricata baseline decision fields.

This project has four LLM triage labels. These are benign, noisy, suspicious, and malicious. The benign label is defined as normal traffic flow without unusual flow evidence or suspicious behavior. The noisy label is defined as a flow that has unusual or possibly irrelevant activities, but the available evidence doesn’t present as an attack. The suspicious label defines flows that contain indicators that need further investigation. The most undesirable triage label is malicious, which contains strong evidence that indicates an attack.

Triage label	**Interpretation**	**Stage-2**
benign	No suspicious evidence identified	No
noisy	Irregular or ambiguous activity with insufficient evidence of an attack	No
suspicious	Requires deeper review	Yes
malicious	Stronger evidence of malicious behavior	Yes

Table 3.5: Stage-1 LLM Triage Labels and Stage-2 Candidate Selection

Stage-1 is designed as a comprehensive screening stage rather than to classify the final attack-class. The benign and noisy triage labels are retained as Stage-1 results, and those flows don’t proceed to further investigation in LLM analysis. But flows with suspicious and malicious labels are selected as candidates for review in Stage-2. This stage identifies the benign triage label as BENIGN attack-class and noisy, suspicious , and malicious triage labels as Unknown attack-class, because this stage doesn’t classify the final attack-class.

Those Stage-1 results are validated through a validation process before the results are accepted as canonical results. The validation process checks whether the LLM prediction results are valid JSON, include flow_id, include all expected fields, have no

duplicate records or missing flows, have valid confidence values, and don’t have malformed outputs that may need a retry. After validation, LLM predictions of Stage-1 are used to select Stage-2 candidates.

3.9 Stage-2 Candidate Review

In Stage-2, the LLM reviews and further investigates in depth those flows that are identified as either suspicious or malicious triage-label from Stage-1. Only flows classified as suspicious or malicious in Stage-1 are selected as Stage-2 candidates. Flows classified as benign or noisy do not proceed to this stage. This candidate-selection strategy limits more detailed analysis to flows with stronger indicators of potentially suspicious or malicious activity.

The candidates of Stage-2 are investigated for attack-class using the bias-controlled flow and protocol evidence. The LLM evaluates each candidate flow and produces a structured prediction containing the flow_id, triage-label, attack-class, confidence, supporting evidence, and a short reasoning statement.

Output Field	**Purpose**
flow_id	Identifies the reviewed flow
triage_label	Final triage interpretation
attack_class	More specific attack-class prediction
confidence	LLM confidence value
evidence	Flow/protocol evidence supporting the decision
reason	Short explanation of the classification

Table 3.6: Stage-2 LLM Candidate Review Output Fields

The Stage-2 attack classes represent more specific interpretations of potentially malicious behavior. Depending on the evidence available in a candidate flow, classes may include categories such as FTP-Patator, SSH-Patator, Bot, Infiltration, Web Attack - Brute Force, Web Attack - SQL Injection, PortScan, DDoS, and Heartbleed. An attack class is assigned only on the basis of the evidence provided to the LLM; Suricata alert signatures, baseline labels, and other forbidden information remain unavailable during Stage-2 analysis.

Stage-2 predictions are subsequently subjected to output validation before being accepted into DB-3, the LLM results database. For flows that undergo Stage-2 review, the validated Stage-2 prediction becomes the final LLM result used in the later comparison stage. For all other flows, the validated Stage-1 prediction remains the final LLM result.

3.10 Validation of Output and Retry

Before being accepted as final predictions and being inserted into DB-3, the outputs of the LLM are checked to ensure their validity. This validation procedure is applied to the outputs from both Stage-1 and Stage-2, and it makes sure that the outputs are complete and structurally correct, for example that the prediction results from both Stage-1 and Stage-2 are valid JSON, that all the flows include the required output fields, and that the flow_id values correspond to the flows that are expected in the input chunk submitted and that match the allowed schema.

After that, the validated outputs are canonicalised into a consistent field structure and inserted into DB-3 (LLM result database) as a last task of the LLM pipeline. This process ensures the outputs are stored with the same field structure and consistent values, reducing inconsistencies caused by variations in LLM formatting.

Validation Check	**Purpose**
JSON parsing	Confirms that the response is structurally valid
Schema validation	Verifies required fields
flow_id matching	Confirms that predictions correspond to submitted flows
Duplicate removal	Prevents multiple predictions for the same flow
Out-of-scope removal	Rejects predictions for flows not present in the input chunk
Confidence validation	Checks allowed confidence values
Missing-flow detection	Identifies expected predictions that were not returned
Canonicalization	Converts accepted predictions into a consistent representation
Retry	Reprocesses missing or invalid predictions

Table 3.7: The validation, canonicalization and retry checks for LLM output

If one or more expected flow predictions are missing, malformed, or invalid, the affected records are identified and retried rather than accepting an incomplete chunk. The retry process is repeated until the required predictions are successfully returned and pass validation, or until the configured retry procedure is exhausted. This prevents incomplete or structurally invalid LLM responses from being silently incorporated into the experimental results.

Only predictions of this project that pass the validation and canonicalization stages are treated as valid LLM outputs for subsequent comparison with the Suricata baseline.

3.11 Final Comparison

The final comparison is performed in DB-4 after the Suricata baseline and LLM analysis paths have been completed independently. Suricata-derived baseline labels come from DB-2, while validated LLM predictions and associated run information come from DB-3. The two result sets are linked at the flow level using the common flow_id.

For flows that do not undergo Stage-2 review, the validated Stage-1 prediction is used as the final LLM result. For flows selected for Stage-2, the validated Stage-2 prediction overrides the corresponding Stage-1 result for the final comparison. This produces one final LLM classification for each evaluated flow.

The comparison evaluates agreement between the Suricata baseline and the final LLM prediction. The main measures were triage_label agreement, triage_label agreement rate, and attack_class agreement rate, Stage-1-only triage-label and attack-class counts, Stage-2 override counts, and cross-classification distributions. Results are calculated at both per-pilot and cross-pilot levels.

Comparison Measure	**Purpose**
triage_label agreement	Measures whether Suricata and LLM triage interpretations match
attack_class agreement	Measures whether attack class interpretations match
triage_label disagreement	Identifies flows with different triage interpretations
attack_class disagreement	Identifies flows with different attack-class interpretations
Stage-1-only count	Counts flows whose final LLM result comes from Stage-1
Stage-2 override count	Counts flows whose Stage-1 result was replaced by Stage-2
Cross-classification matrix	Summarizes cross-classification patterns between Suricata and LLM

Table 3.8: Measures Used for Suricata–LLM Final Comparison

The comparison results are stored in DB-4 separately from both the Suricata baseline database (DB-2) and the LLM results database (DB-3). The comparison does not treat Suricata output as verified ground truth. Therefore, agreement indicates that Suricata and the LLM produced canonical labels for a flow, while disagreement indicates a difference between the two analysis approaches. The evaluation therefore

characterizes agreement and disagreement between the two analysis approaches rather than treating either system as a verified measure of detection accuracy.

4 Implementation

4.1 Experimental Environment

The experimental pipeline was implemented in a Linux-based environment using Python and SQLite [15]. Suricata processed the CIC-IDS2017 pcap files and generated EVE JSON event logs [5]. ChatGPT-assisted Python scripts parsed the EVE output, reconstructed it as flow-level records by aggregating and correlating, generated the Suricata baseline, constructed bias-controlled LLM inputs, split the validated bias-controlled inputs into chunks, sent chunks to the LLM using API access, validated LLM responses, and performed the final Suricata-versus-LLM comparison.

SQLite was used as four separate databases for every pilot: DB-1, DB-2, DB-3, and DB-4. DB-1 and DB-2 belong to the IDS pipeline and store EVE events and aggregated flows, respectively. DB-3 stores LLM prediction results, and DB-4 stores comparison results of Suricata baseline outputs and LLM prediction outputs. Separate database files were maintained for each pilot in order to preserve pilot-level traceability throughout the processing pipeline.

Component	**Configuration**
Operating System	Kali GNU/Linux Rolling 2026.3
Python	3.13.14
Suricata	8.0.6 RELEASE
SQLite	3.46.1
Dataset	CIC-IDS2017
Evaluated Pilots	Pilot4–Pilot8
LLM Provider	KISSKI
LLM Model	meta-llama-3.1-8b-instruct
Code & AI Assistance	OpenAI GPT5.5 & GPT-5.6 Sol [16]
Temperature	0
LLM Access	OpenAI-compatible REST API

Table 4.1: Experimental Software and Processing Environment

LLM inference was performed through the configured KISSKI API. The model was accessed programmatically from the Python-based processing scripts using the configured API base URL, model identifier, and authentication key. Deterministic inference settings were used where applicable, including a temperature of 0, in order to reduce unnecessary variation between repeated prediction requests.

The implementation consisted of a series of specific processing scripts. At each stage explicit intermediate files or database entries were produced which could be checked before the next stage was carried out. The modular design enabled debugging, validation, the handling of retries, and reproducibility in the five CIC-IDS2017 pilots.

4.2 Processing Pipeline

A sequence of processing stages was put into place, the output from each stage being used as the input for the following one. The processing was carried out separately for each of the CIC-IDS2017 pilot studies, which allowed all the intermediate files, database records, the LLM predictions, and the final comparison results to be traced back to their specific pilots.

Implementation stage	**Main script**
Raw EVE storage	0_store_eve_events.py
Flow correlation and aggregation	1_build_suricata_baseline_from_eve_db.py
Bias-controlled LLM input generation	2_make_llm_input_txt_compact_from_baseline_db.py
Bias-control validation	7_bias_control_check.py
LLM input splitting	4_split_llm_input_txt_by_size.py
Stage-1 LLM processing	6_send_txt_chunks_as_batches_to_llm.py
Stage-2 candidate selection	8_stage2_candidates_from_stage1.py
Stage-2 LLM processing	6b_send_stage2_candidates_to_llm.py
LLM JSONL validation	9_validate_llm_jsonl_before_db.py
DB-3 LLM results insertion	10_insert_llm_results_to_db3.py
DB-4 comparison construction	11_build_comparison_db4.py

Table 4.2: The main processing stages and the implementation scripts

The EVE JSON event records generated by Suricata were first stored in DB-1; the events were then correlated using the flow_id and aggregated into flow-level records in DB-2. At this stage, reconstructed flows, protocol-specific details, summaries of Suricata alerts, Suricata-derived flow labels, and input-generation metadata were produced for further processing.

Subsequently, bias-controlled LLM input was generated from selected fields within the flows and flow_protocol_details tables. Suricata alert summaries and baseline labels were excluded from the LLM inputs. Prior to model submission, a bias-control validation step ensured that prohibited Suricata-derived information was not present in the LLM input.

After validation, the LLM input was divided into processing chunks and submitted to the KISSKI-hosted LLM for Stage-1 triage. Stage-1 responses were parsed, validated, canonicalized, and checked for missing or invalid predictions. Flows identified as suspicious or malicious were selected as Stage-2 candidates and submitted for further LLM review.

Stage-2 responses underwent the same validation and canonicalization procedures before the final LLM predictions were inserted into DB-3. For flows reviewed in Stage-2, the validated Stage-2 result replaced the corresponding Stage-1 result during final result construction. For all other flows, the validated Stage-1 prediction was retained as the final LLM result.

Finally, DB-4 stored the comparison result of the independently generated Suricata baseline from DB-2 with the validated LLM results from DB-3 using flow_id. This comparison database was used to generate per-pilot and cross-pilot agreement and disagreement metrics, Stage-2 override counts, and cross-classification results.

4.3 Pilot Processing

For five of the CIC-IDS2017 pilots, which corresponded to different dataset days, the implementation was carried out independently. Each of the pilots went through the same pipeline, starting with the storage of Suricata EVE events and flow reconstruction and then proceeding to bias-controlled LLM analysis before ending with the comparison between Suricata and the LLM.

The number of flows that were reconstructed differed from one pilot to another, this being due to the different traffic captured on the various days of the CIC-IDS2017 dataset.

In the five pilots altogether 1,967,638 flow records were processed, the highest number of flows being in Pilot8 with 524,374 records and the lowest in Pilot6 with 294,900. Each of the pilots was processed separately by means of its individual DB-1, DB-2, DB-3, and DB-4 database files, thus maintaining pilot-level traceability during the experiment.

4.4 LLM Input Splitting

To decide on the final chunk configuration, preliminary implementation tests were carried out using Pilot6. The tests evaluated various input sizes and the number of flows per request in order to find a suitable configuration for LLM processing. These experiments looked at different chunk sizes and flow numbers, as well as different environments for accessing the LLM, such as OpenRouter, Ollama, and KISSKI.

As a result of these implementation tests, the KISSKI system together with the meta-llama-3.1-8b-instruct model was chosen for use in the final processing pipeline. The final setup employed 100 flows per Stage-1 chunk/API request and 50 candidate flows per Stage-2 chunk/API request. This configuration was selected on the basis of what

was observed during the initial testing, as it represented a practical compromise between request size, response completeness, processing stability, and the level of detail needed from the model.

In the case of Stage-1, the validated input was therefore split up into batches of 100 flows. Since Stage-1 carries out a general triage and needs relatively short outputs for each flow, the arrangement with 100 flows enabled multiple flows to be processed efficiently while at the same time keeping the responses structured.

In Stage-2 the chunks used were smaller and contained 50 candidate flows. Since Stage-2 called for more detailed outputs—such as attack-class interpretation, confidence levels, supporting evidence, and the reasoning—reducing the number of flows per request gave additional response capacity for this in-depth analysis.

Processing Stage	**Chunk Size per API Request**	**Purpose**
Stage-1	100 flows	Broad triage with efficient batch processing
Stage-2	50 flows	Deeper candidate analysis with more detailed output

Table 4.3: Final LLM Input Chunk Sizes Used for Stage-1 and Stage-2 Processing

The input-splitting procedure kept the flow_id of each record so that the predictions could later be verified against the flows included in the relevant input chunk. It also allowed for the detection of missing predictions and for the selective retry of incomplete or invalid responses from the LLM.

4.5 Stage-1 Implementation

Stage-1 LLM processing was performed using the 6_send_txt_chunks_as_batches_to_llm.py script. The validated bias-controlled input chunks generated in the previous processing stages were submitted sequentially to the KISSKI-hosted meta-llama-3.1-8b-instruct model through the configured OpenAI-compatible REST API. Each Stage-1 request contained up to 100 flow records.

The model was told in the Stage-1 prompt to assess each flow by referring only to the flow level and the protocol level evidence found in the submitted chunk. Since the Suricata alert signatures, alert categories, severities, and the Suricata baseline labels were not part of the input given to the model, it had to carry out the classification of the flows without any information about the Suricata detection result.

For each flow that was submitted, the model had to provide a structured prediction which included the flow_id, triage_label, attack_class, and confidence fields. The allowed values for triage_label were benign, noisy, suspicious, and malicious. Detailed prediction of the attack class was deliberately limited during Stage-1. Flows that were

triaged as benign used BENIGN as the attack class, while flows classified as noisy, suspicious, or malicious retained Unknown until deeper Stage-2 analysis.

The Stage-1 responses that were returned were parsed, checked for validity and put into their standard form before being accepted as usable predictions. The processing scripts verified the structured output format, the expected flow_id values, the allowed labels, the presence of duplicate records, and the presence of missing predictions; any invalid or missing predictions were flagged for retry and were not accepted silently.

The validated Stage-1 results were kept for all the flows that had been processed. Those flows which were classified as benign or noisy kept the validated Stage-1 prediction as their final result in the LLM. Flows that were classified as suspicious or malicious were extracted by the Stage-2 candidate selection process and then went on to Stage-2 review.

4.6 Stage-2 Implementation

The Stage-2 LLM processing was carried out by means of the 6b_send_stage2_candidates_to_llm.py script. Only those flows which had been identified as suspicious or malicious in the validated Stage-1 processing were selected as Stage-2 candidates. The candidate flows were split into smaller chunks, each containing no more than 50 flows, and then sent to the same KISSKI-hosted meta-llama-3.1-8b-instruct model via the OpenAI-compatible REST API.

Stage-2 made use of the same bias-controlled flow and protocol evidence that was available in Stage-1. As in the case of Stage-1, Suricata alert signatures, alert categories, severity information, the flow labels derived by Suricata, and the Suricata baseline classifications were still left out of the model input. Although Stage-1 had decided which flows were to be reviewed, the Suricata results did not have any effect on the Stage-2 classification process.

In contrast to Stage-1 triage, the Stage-2 prompt asked for a more detailed analysis of each candidate flow. The model had to give a structured prediction including the flow_id, triage_label, attack_class, confidence, supporting evidence and a brief reasoning statement. The aim of this more thorough examination was to improve the interpretation of the flows that had been identified in Stage-1 as needing further investigation.

The returned Stage-2 responses were parsed, validated and canonicalized before being accepted. The validation process checked the expected flow_id values, the required output structure, the presence of duplicate records, the absence of predictions, and other malformed responses. Those predictions which were invalid or incomplete were isolated and then retried instead of being included directly in the final result set.

The Stage-2 predictions that had been validated and canonicalized were taken as the final LLM results for the relevant candidate flows, and when the final results were being constructed the validated Stage-2 prediction replaced the earlier Stage-1 prediction for the same flow_id; the flows which were not chosen for Stage-2 kept the validated Stage-1 predictions as their final LLM results.

4.7 Output Validation and Retry Handling

The responses generated by both Stage-1 and Stage-2 were validated before being included in the database. The primary method used for this validation was the 9_validate_llm_jsonl_before_db.py script, since API responses might include malformed JSON, missing predictions, duplicate records, unexpected flow identifiers, or values that do not comply with the required output schema.

The predictions returned for each chunk were verified against the relevant input records. As part of the validation procedure, it was confirmed that each expected flow_id was included, that no flow identifiers outside the specified scope had been introduced, and that no duplicate predictions had been kept. Moreover, the required fields and categorical values were checked to make certain that the triage_label, attack_class, the confidence value, and the other fields specific to the stage adhered to the anticipated structure.

Before any further processing, the valid predictions were converted into a consistent format. This involved standardising the values and making sure that the structured output could be consistently processed across different chunks and pilots. Instead of being quietly included in the experiment, those predictions which failed parsing or schema validation were excluded from the final set of accepted results.

The procedure used to verify the results also detected any missing flows by comparing the set of expected flow_id values in each input chunk with the flow_id identifiers provided by the model. Those predictions which were missing, malformed, or otherwise invalid were separated out so that they could be retried selectively. In this way only the records in question were resubmitted rather than having the entire processing of a previously successful chunk repeated.

The validation and retry process was carried out separately for the outputs of Stage-1 and Stage-2. The Stage-1 results were not regarded as complete until the necessary flow predictions had successfully passed validation, and the Stage-2 results likewise had to go through the same validation procedure before they could be accepted as valid candidate-review predictions. The final selection of Stage-2 over Stage-1 was made later on during the construction of DB-4. After that, only those LLM results which had been validated and canonicalized were accepted for storage in DB-3.

4.8 DB-3 LLM Results Insertion

The Stage-1 and Stage-2 LLM outputs which had been validated and canonicalized were inserted into DB-3 by means of the 10_insert_llm_results_to_db3.py script. For each pilot there was a separate DB-3 database, the name of the database file being pilot*_llm_unbiased_results.db. Even though the name of the implementation file includes the word unbiased, DB-3 is called the bias-controlled LLM results in this study. The reason is that the experimental design prevents information leakage by properly constructing and validating the inputs rather than by assuming that there is no bias at all.

The three main tables in DB-3 are llm_input_metadata, llm_runs, and llm_predictions. The llm_input_metadata table enables traceability between the flows that have been processed and the LLM inputs that they generate. The llm_runs table keeps records of execution metadata including the pilot, provider, model, prompt version, temperature, token configuration, input mode, and the processing directories. The llm_predictions table holds the validated predictions linked to each flow_id.

The Stage-1 and Stage-2 predictions were stored as individual LLM executions instead of being combined when inserting the data into the database. Stage-1 includes a validated prediction for each flow that has been processed, whereas Stage-2 has extra predictions only for those flows which were identified as suspicious or malicious in Stage-1. Each prediction is associated with the run_id of the corresponding run, enabling the processing stage from which it came to be determined.

When the data is being inserted, the Stage-1 records keep the triage label, the attack class, and the confidence information; furthermore, the Stage-2 records also retain the more detailed evidence and reasoning which is returned during the candidate review. The insertion process ensures that there is a relationship maintained between the prediction, its flow_id, the relevant LLM run, and the input metadata.

The only results which had successfully gone through the validation, canonicalization, and the required retry procedures described in Section 4.7 were included as accepted predictions. The Suricata baseline labels and the information derived from Suricata alerts stayed outside DB-3 and continued to be stored separately in DB-2.

The selection between the Stage-1 and Stage-2 predictions was not made at the time of inserting DB-3. Rather, it was made when constructing DB-4: if there was a valid Stage-2 prediction for a given flow_id, then this prediction was used to replace the corresponding Stage-1 prediction for the final comparison between Suricata and the LLM; if there was no such valid Stage-2 prediction, the validated Stage-1 prediction was kept as the final result of the LLM.

4.9 DB-4 Comparison Implementation

The final comparison database, DB-4, was created using the 11_build_comparison_db4.py script after the Suricata baseline and the LLM processing paths had been completed. For each pilot, DB-4 was constructed separately and the information from DB-2 and DB-3 was only combined at the final stage of the comparison. In this way, the distinction between the Suricata baseline and the bias-controlled LLM analysis was maintained throughout the prediction process.

The comparison process began by loading the validated Stage-1 and Stage-2 predictions from DB-3, using the Stage-1 predictions as the initial set of results from the LLM; and where a validated Stage-2 prediction was available for the same flow_id, the Stage-2 prediction replaces the Stage-1 prediction for the final comparison; as a result, each flow had a single final LLM output when DB-4 was being constructed, even though the original Stage-1 and Stage-2 records were still kept separately in DB-3.

The Suricata baseline records taken from the suricata_flow_labels table in DB-2 were matched with the final LLM results using the exact flow_id. For each flow that was matched, DB-4 stored the Suricata triage label, attack class, attack subtype and reason along with the corresponding LLM triage label, attack class, attack subtype, confidence and rationale. The implementation also kept a record of whether the LLM result had come solely from Stage-1 or was a Stage-2 override.

The agreement was assessed at several levels. The agreement regarding the triage label was calculated after adjusting for differences in text case and whitespace. When comparing attack classes, normalization was also used so that equivalent benign labels such as Benign and BENIGN were handled in a consistent manner. Agreement with respect to attack subtype was examined on its own. In the case of flows for which no final prediction from the LLM was available, a specific comparison status was assigned rather than leaving them out silently.

The results of the comparison were saved in the comparison_results table, with the run-level provenance being recorded in comparison_runs. Aggregate statistics were written to comparison_summary and these included the number of Stage-1 predictions, the number of Stage-2 predictions, the number of Stage-1-only results, the number of Stage-2 overrides, the number of missing predictions, the agreement counts, the triage label distributions, the attack-class distributions, and the Suricata versus LLM agreement rate.

DB-4 is therefore the point at which the independently produced Suricata baseline and the LLM outputs were combined. The measurements obtained do describe the agreement, disagreement, and cross-classification behaviour between the two detection methods. However, they are not regarded as direct accuracy measurements since the Suricata baseline is used as a point of comparison rather than as confirmed ground truth.

5 Results

5.1 Results Overview

The experimental pipeline has been completed for all five of the selected CIC-IDS2017 pilots, each of them being processed separately via both the Suricata baseline path and the bias-controlled LLM analysis path before the results were combined in DB-4 for the final comparison.

In the case of the five pilots, 1,967,638 flow records were processed. For each flow record that was processed, Stage-1 LLM triage generated one valid prediction, and those flows identified as suspicious or malicious were then sent to Stage-2 for more in-depth candidate examination.

The number of Stage-2 candidates differed from one pilot to another: Pilot4 generated 4,343 Stage-2 predictions, Pilot5 produced 3,464, Pilot6 produced 5,893, Pilot7 produced 3,801, and Pilot8 produced 4,954; the total number of flows that had Stage-2 review was 22,455.

Pilot	Dataset Day	Total Flows	Stage-2 Candidates
Pilot4	Wednesday	470,977	4,343
Pilot5	Monday	342,997	3,464
Pilot6	Tuesday	294,900	5,893
Pilot7	Thursday	334,390	3,801
Pilot8	Friday	524,374	4,954
Total	**-**	**1,967,638**	**22,455**

Table 5.1: The number of processed flows and the number of stage-2 candidate entries in the five pilots

The findings described in this chapter are given in three separate stages. Firstly, the Suricata baseline results obtained from DB-2 are stated on their own. Second, the Stage-1 and Stage-2 LLM results kept in DB-3 are shown without referring to the Suricata baseline. Lastly, the two separately produced sets of results are combined using DB-4 in order to look at instances of agreement and disagreement, Stage-2 overrides, and cross-classification behaviour.

5.2 Suricata Baseline Results

The baseline results for Suricata were obtained separately by referring to the suricata_flow_labels table in DB-2 for each pilot; the number of flows labelled at the baseline level was the same as the total number of reconstructed flows in each pilot, which gave a total of 1,967,638 Suricata-labelled flows for the entire experiment.

The Suricata triage labels varied greatly among the five pilots. In all the pilots combined, 1,471,443 flows were labelled as benign, 470,891 as noisy, 25,153 as suspicious and 151 as malicious. This represents about 74.78% benign, 23.93% noisy, 1.28% suspicious and 0.01% malicious traffic in the Suricata baseline.

Pilot4 had the highest percentage of noisy traffic, consisting of 207,126 out of 470,977 flows that were classified as noisy, the remaining 259,361 being classified as benign. Pilot5 had the greatest number of suspicious flows, with 9,042 flows given the suspicious designation. Pilot7 generated the largest number of malicious baseline classifications, having 141 malicious flows, while the other pilots had between one and six flows classified as malicious.

Table 5.2 gives a summary of the distribution of the four Suricata triage labels for each pilot.

Pilot	Benign	Noisy	Suspicious	Malicious	Total
Pilot4	259,361	207,126	4,489	1	470,977
Pilot5	284,637	49,316	9,042	2	342,997
Pilot6	245,859	45,148	3,892	1	294,900
Pilot7	292,804	37,594	3,851	141	334,390
Pilot8	388,782	131,707	3,879	6	524,374
Total	**1,471,443**	**470,891**	**25,153**	**151**	**1,967,638**

Table 5.2: Suricata Triage-Label Distribution Across the Pilots

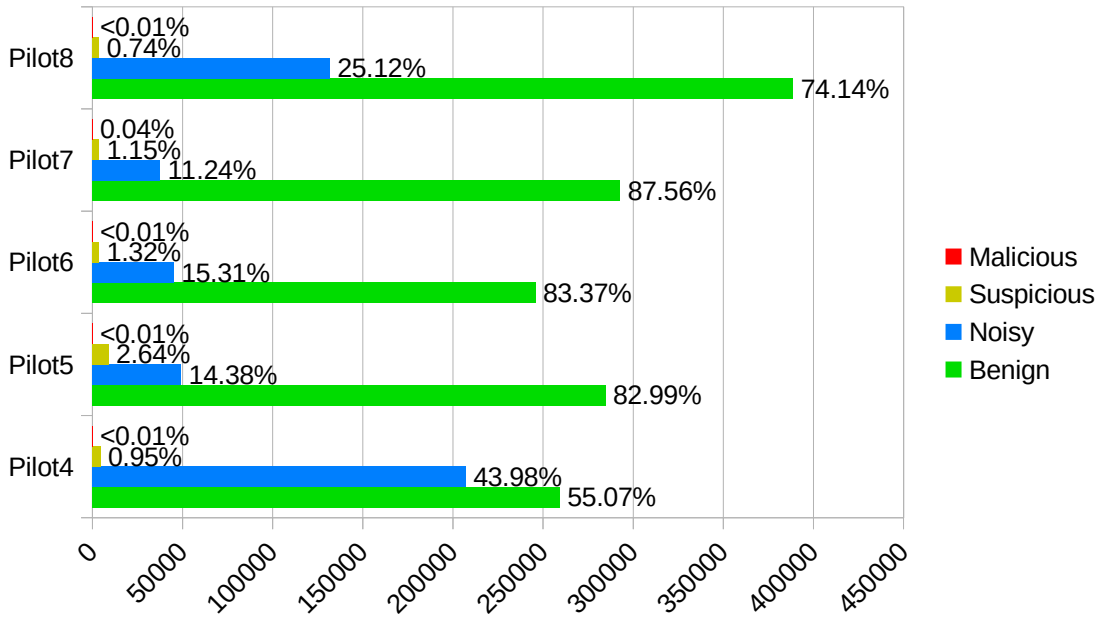

Figure 5.1: The distribution of Suricata Triage labels across the pilots

At the attack-class level, the majority of the Suricata baseline classifications were given the label Benign or were classified as Unknown. Of the flows examined by the five pilots, 1,471,443 were labelled Benign and 496,044 were assigned Unknown status. Flows that were categorised as noisy or suspicious were those assigned to the Unknown class, meaning that the baseline construction procedure had not assigned them a particular attack class.

Just 151 of the flows were subjected to a particular malicious attack category. These included 92 PortScan flows, 55 Bot flows, and four DoS flows. The variety of specific attack types varied from pilot to pilot. Pilots 4, 5, and 6 had only Bot classifications among the malicious flows that were specifically identified. Pilot 7 had the greatest number of specific types, with 88 PortScan, 49 Bot, and four DoS flows. Pilot 8 had four PortScan and two Bot classifications.

A summary of the Suricata attack-class distribution among the five pilots is given in Table 5.3.

Pilot	Benign	Unknown	Bot	PortScan	DoS	Total
Pilot4	259,361	211,615	1	0	0	470,977
Pilot5	284,637	58,358	2	0	0	342,997
Pilot6	245,859	49,040	1	0	0	294,900
Pilot7	292,804	41,445	49	88	4	334,390
Pilot8	388,782	135,586	2	4	0	524,374
Total	**1,471,443**	**496,044**	**55**	**92**	**4**	**1,967,638**

Table 5.3: Suricata Attack-Class Distribution Across the Pilots

The distribution of the attack-class results was the same. Benign flows were assigned to the None subtype, and the majority of the flows which did not have a specific attack identified were recorded as Unknown. For the Bot and PortScan cases, specific subtype assignments were noted, which is in line with the findings at the attack-class level. The four flows that were included in the DoS attack class in Pilot7 were still classified as Unknown at the subtype level. These Suricata baseline results are kept as a separate reference and are only compared with the LLM results in the later comparison sections.

5.3 Stage-1 LLM Triage Results

In the five pilots, the Stage-1 LLM triage produced one validated prediction for each flow that was processed, giving a total of 1,967,638 Stage-1 predictions. These predictions were made independently of the Suricata baseline and were given one of four triage labels: benign, noisy, suspicious, or malicious.

In the five pilots the LLM identified 1,910,554 flows as benign, 34,629 as noisy, 19,558 as suspicious, and 2,897 as malicious. This represents about 97.10% benign, 1.76% noisy, 0.99% suspicious, and 0.15% malicious predictions at Stage-1.

The number of flows needing further review differed from pilot to pilot. Pilot6 generated the highest number of such flows, including 4,742 suspicious and 1,151 malicious predictions. These 5,893 flows accounted for about 2.00% of the traffic in Pilot6. On the other hand, the proportion of Stage-2 candidates in the other pilots was between about 0.92% and 1.14%.

The distribution of the Stage-1 triage labels is given in Table 5.4.

Pilot	Benign	Noisy	Suspicious	Malicious	Total
Pilot4	458,758	7,876	3,919	424	470,977
Pilot5	333,272	6,261	3,054	410	342,997
Pilot6	282,749	6,258	4,742	1,151	294,900
Pilot7	323,328	7,261	3,423	378	334,390
Pilot8	512,447	6,973	4,420	534	524,374
Total	**1,910,554**	**34,629**	**19,558**	**2,897**	**1,967,638**

Table 5.4: Stage-1 LLM Triage-Label Distribution Across the Five Pilots

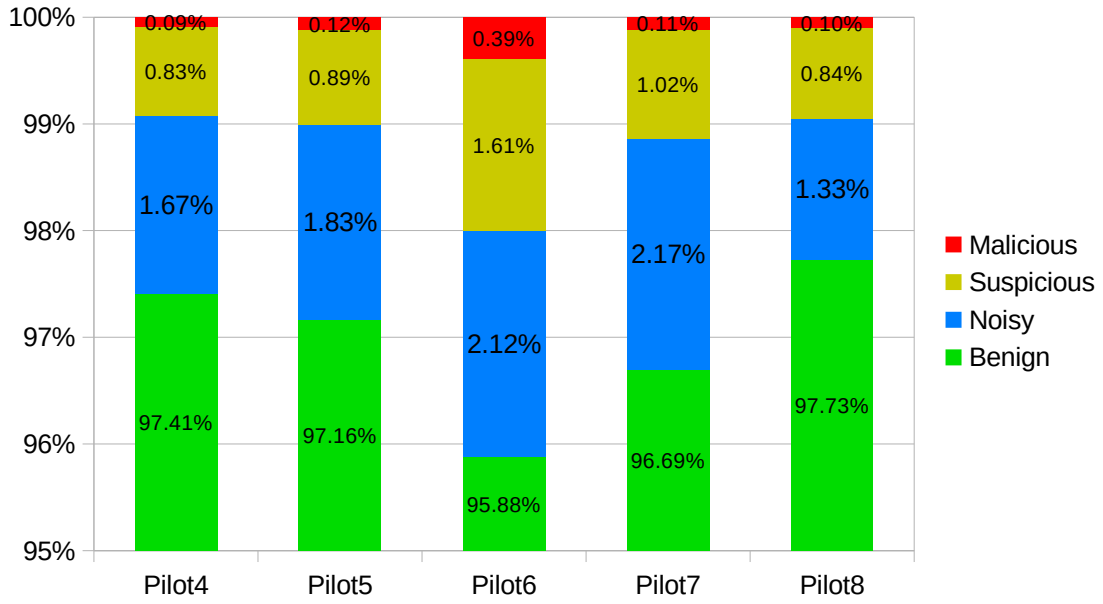

Figure 5.2: Percentage Distribution of Stage-1 LLM Triage Labels Across the Pilots

5.4 Stage-2 Candidate Review Results

The Stage-2 analysis was carried out only for the 22,455 flows which had been identified as suspicious or malicious in Stage-1. The candidate flows were then re-assessed using the more detailed Stage-2 prompt, and this resulted in the production of refined triage labels, attack-class interpretations, confidence values, supporting evidence, and reasoning.

After the Stage-2 review, 20,358 of the 22,455 candidate flows were classified as benign, 1,033 as noisy, 654 as suspicious, and 410 as malicious. This represents about 90.66% benign, 4.60% noisy, 2.91% suspicious, and 1.83% malicious in the group of candidate flows at the Stage-2 level.

The distribution of the Stage-2 triage varied in the five pilots, and Pilot6 had the highest proportion of higher-risk classifications after the more detailed review; 285 of the

candidate flows were still suspicious, and 251 were classified as malicious, which amounted to approximately 4.84% and 4.26% of its Stage-2 candidates, respectively. In contrast, more than 92% of Stage-2 candidates in Pilot4 and Pilot5 were reclassified as benign. Pilot8 had the highest benign proportion at approximately 93.10%. The Stage-2 triage-label distribution is summarized in Table 5.5.

Pilot	Benign	Noisy	Suspicious	Malicious	Total Candidates
Pilot4	3,996	210	100	37	4,343
Pilot5	3,215	148	71	30	3,464
Pilot6	5,163	194	285	251	5,893
Pilot7	3,372	286	90	53	3,801
Pilot8	4,612	195	108	39	4,954
Total	**20,358**	**1,033**	**654**	**410**	**22,455**

Table 5.5: Stage-2 LLM Triage-Label Distribution Across the Pilots

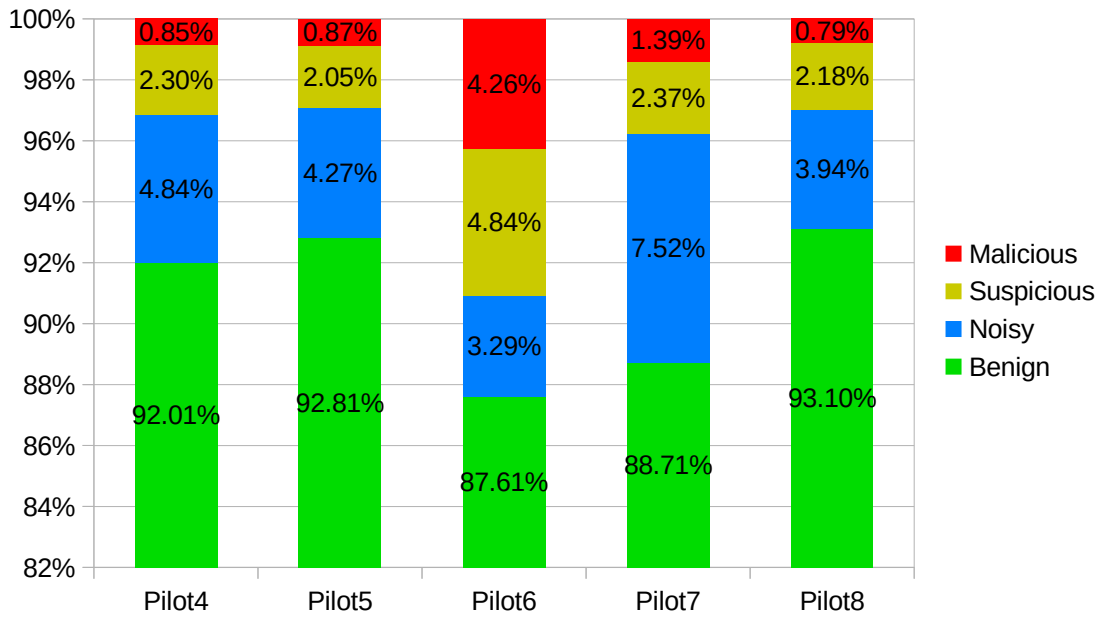

Figure 5.3: The percentage distribution of the stage-2 LLM triage labels among the pilots

For Stage-2, more detailed attack-class assignments were given to the candidate flows. Most of the candidates were assigned either BENIGN or Unknown. In all the pilots, 20,358 Stage-2 predictions were classified as BENIGN and 1,680 were given the

label Unknown. The other 417 predictions were assigned a specific attack-class interpretation.

The most commonly occurring type of specific attack was FTP-Patator, accounting for 236 predictions, and then came Web Attack - Brute Force with 115 predictions. Other attack types were Web Attack - Sql Injection (21), Bot (19), DDoS (6), Infiltration (6), Heartbleed (5), SSHPatator (3), PortScan (2), LDAP (2), SMB (1), and DoS slowloris (1).

Stage-2 Attack Class	**Count**
BENIGN	20,358
Unknown	1,680
FTP-Patator	236
Web Attack - Brute Force	115
Web Attack - Sql Injection	21
Bot	19
DDoS	6
Infiltration	6
Heartbleed	5
SSH-Patator	3
PortScan	2
LDAP	2
SMB	1
DoS slowloris	1
Total	**22,455**

Table 5.6: The distribution of the attack classes for the Stage-2

High confidence was also mainly given to the stage-2 predictions. Of the 22,455 candidate predictions, 20,392 were given high confidence, 1,761 were given low confidence, and 302 received medium confidence. This represents about 90.81%, 7.84%, and 1.34% of the stage-2 predictions respectively. The LLM predictions showed LDAP and SMB as attack-class values, though the official CIC-IDS2017 dataset doesn’t have those attack classes, and LDAP and SMB were not included in the Stage-2 prompt's allowed attack_class list.

5.5 The Final Prediction Results of the LLM

After completion of Stage-2 review, the final LLM result set was constructed by retaining the validated Stage-1 prediction for flows that did not proceed to Stage-2 and replacing the Stage-1 prediction with the validated Stage-2 prediction where a Stage-2 result existed for the same flow_id.

In the five pilots the final set of LLM predictions totaled 1,967,638. Of these, 1,930,912 were classified as benign, 35,662 as noisy, 654 as suspicious, and 410 as malicious. This corresponds to approximately 98.13% benign, 1.81% noisy, 0.03% suspicious, and 0.02% malicious traffic in the final LLM output.

The impact of the Stage-2 review was therefore significant for the set of candidates. Even though 22,455 flows reached Stage-2 having been classified as suspicious or malicious in Stage-1, only 1,064 of them stayed in the suspicious or malicious categories following the more in-depth review. The other candidates who proceeded to Stage-2 were mostly reclassified as benign or noisy. A summary of the final LLM triage-label distribution among the five pilots is given in Table 5.7.

Pilot	**Benign**	**Noisy**	**Suspicious**	**Malicious**	**Total**
Pilot4	462,754	8,086	100	37	470,977
Pilot5	336,487	6,409	71	30	342,997
Pilot6	287,912	6,452	285	251	294,900
Pilot7	326,700	7,547	90	53	334,390
Pilot8	517,059	7,168	108	39	524,374
Total	**1,930,912**	**35,662**	**654**	**410**	**1,967,638**

Table 5.7: The distribution of the LLM triage labels among the pilots

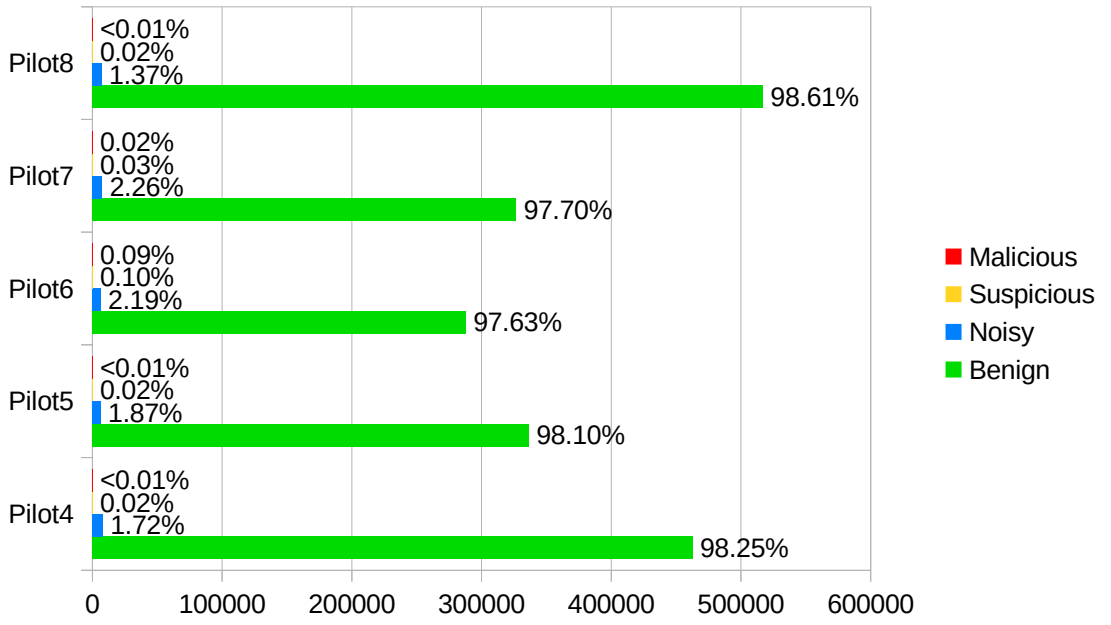

Figure 5.4: The percentage distribution of the final LLM triage labels among the pilots

Pilot6 had the highest number of final suspicious and malicious predictions, with 285 and 251 flows in each case. The other four pilots had considerably fewer final high-risk predictions. These results show the full LLM analysis path and are used in the following section for comparison with the independently generated Suricata baseline.

5.6 Suricata versus LLM Comparison

The Suricata baseline and the final LLM results were each generated independently, after which the two result sets were compared in DB-4 by means of the common flow_id. In the case of flows that were reviewed in Stage-2, the validated Stage-2 prediction was taken as the final LLM result; in all other cases, the validated Stage-1 prediction was kept. The comparison thus assesses the final two-stage LLM output against the independently constructed Suricata baseline.

Agreement on the triage labels differed greatly among the five pilots. Pilot4 had the lowest agreement rate (53.76%), whereas Pilot7 had the highest (85.35%). The agreement rates for Pilot5 and Pilot6 were similar, being 81.16% and 81.13%, respectively, and Pilot8 obtained 73.00% agreement. The agreement for the attack class also followed the same pattern, varying from 53.79% in Pilot4 to 85.39% in Pilot7. A summary of the comparison results is given in Table 5.8.

Pilot	Total Flows	Triage-Label Agreement	Triage-Label Agreement Rate	Attack-Class Agreement Rate
Pilot4	470,977	253,218	53.76%	53.79%
Pilot5	342,997	278,386	81.16%	81.19%
Pilot6	294,900	239,266	81.13%	81.17%
Pilot7	334,390	285,413	85.35%	85.39%
Pilot8	524,374	382,776	73.00%	73.02%

Table 5.8: The agreement between Suricata and the LLM

The results of the triage cross-classification indicated that a major cause of disagreement was the fact that flows which the Suricata baseline had labelled as noisy or suspicious were classified as benign by the final LLM. This was especially noticeable in Pilot4, since 206,092 flows which had been marked as noisy by Suricata and 4,406 flows which had been marked as suspicious by Suricata were regarded as benign by the LLM.

This pattern was also seen in the other pilots. In Pilot5, the LLM regarded 49,190 Suricata-noisy flows and 8,986 Suricata-suspicious flows as benign; in Pilot6, the corresponding figures were 44,971 and 3,817; in Pilot7, they were 37,464 and 3,773; and in Pilot8, they were 131,058 and 3,810. The fact that a large number of the Suricata-noisy flows were classified as benign by the LLM played a major role in the observed disagreement rates.

The disagreement was not one-way. There were instances where flows deemed benign by Suricata were given non-benign final labels by the LLM. For example, in Pilot6, 6,246 flows which were classified as benign by Suricata were labelled as noisy by the LLM, 249 as suspicious and 241 as malicious. Pilot8 also included 6,516 flows that were regarded as benign by Suricata and were classified as noisy by the LLM, 58 as suspicious and 23 as malicious. Similar but smaller cross-classification patterns were found in the other pilots.

At the attack-class level the main agreement was between Suricata Benign and LLM BENIGN; however, a large number of flows which had been assigned Unknown by the Suricata baseline were classified as BENIGN by the LLM. This was the case with 210,498 flows in Pilot4, 58,176 in Pilot5, 48,788 in Pilot6, 41,237 in Pilot7 and 134,868 in Pilot8.

The comparison between the different class levels also revealed differences in the way the attacks were interpreted. In Pilot7, of the 88 flows which Suricata had identified as PortScan, 73 were classified as BENIGN by the LLM and the other 15 were classified

as Unknown. All 49 of the Suricata Bot flows in Pilot7 were classified as BENIGN. In Pilot8, all four of the Suricata PortScan flows and both of the Suricata Bot flows were classified as BENIGN by the final LLM.

On the other hand, the LLM allocated a number of specific attack types to flows that Suricata had classified as either Benign or Unknown. In the various pilots, these types were FTP-Patator, Web Attack - Brute Force, Web Attack - Sql Injection, Bot, DDoS, Heartbleed, Infiltration, SSH-Patator, PortScan, and some other less frequent categories. The fact that such cross-classification occurred shows that the two methods did not always understand the same flow evidence in the same way.

Therefore, the comparison results show agreement for many flows along with systematic differences in attack classes and triage labels. Because the Suricata outputs are used as an independent baseline rather than as verified ground truth, the reported values represent agreement, disagreement, and cross-classification between the two approaches and five pilots. So the comparison result is not interpreted as direct measures of LLM classification accuracy.

5.7 Cross-Pilot Results Summary

In the five CIC-IDS2017 pilots the full experimental procedure handled 1,967,638 flows. The Suricata baseline identified 1,471,443 of these as benign, 470,891 as noisy, 25,153 as suspicious and 151 as malicious. The distribution differed greatly from one pilot to another, there being the highest proportion of noisy traffic in Pilot4 and the greatest number of classifications as malicious by Suricata in Pilot7.

The independent Stage-1 LLM analysis generated one validated prediction for each flow that was processed. Stage-1 determined that 1,910,554 flows were benign, 34,629 were noisy, 19,558 were suspicious and 2,897 were malicious. A total of 22,455 flows which had been classified as suspicious or malicious were sent on to Stage-2 for more thorough examination.

Stage 2 considerably reduced the number of candidate flows. From the 22,455 flows that were examined, 20,358 were deemed to be benign, 1,033 were considered noisy, 654 were classified as suspicious, and 410 were identified as malicious. As a result, only 1,064 flows remained in the two higher-risk triage categories after the more in-depth review. Stage 2 also generated a variety of specific attack-class assignments, with FTP-Patator and Web Attack Brute Force being the most commonly assigned specific attack classes.

Following the application of the Stage-2 override rule, the final set of LLM predictions included 1,930,912 benign, 35,662 noisy, 654 suspicious, and 410 malicious cases. The final LLM output was thus largely composed of benign cases in all five pilots. The comparison between Suricata and the LLM showed that the level of agreement was not the same across the different dataset days. The rate of agreement on triage labels

varied from 53.76% in Pilot4 to 85.35% in Pilot7; Pilots 5 and 6 achieved similar agreement rates of about 81%, while Pilot8 had an agreement rate of about 73%. The pattern of agreement for attack classes was very similar.

A cross-classification analysis revealed that a common cause of disagreement was the way in which the Suricata-noisy and Suricata-suspicious flows were treated. A large number of these flows were classified as benign by the final LLM. At the same time, a smaller number of Suricata-benign flows were given noisy, suspicious, malicious, or specific attack-class interpretations by the LLM. These findings show that the two analysis methods produced overlapping but different interpretations of the same reconstructed flow evidence.

The results given in this chapter form the empirical foundation for the discussion in the following chapter. Chapter 6 looks at the patterns of agreement and disagreement observed, the effect of the two-stage LLM design, the implications of the bias-controlled input strategy, and the limitations of using Suricata as a comparison baseline rather than verified ground truth.

6 Discussion

6.1 Interpretation of Main Findings

The experimental results indicate that the Suricata baseline and the bias-controlled LLM analysis arrived at different interpretations of the same reconstructed network flows. In the five CIC-IDS2017 pilots, a total of 1,967,638 flows were analysed by both methods. While a significant number of the final classifications were in agreement, the rate of agreement differed greatly from pilot to pilot, going from 53.76% in Pilot4 to 85.35% in Pilot7. This difference shows that the relationship between the two methods depended on the traffic characteristics of the individual dataset days and was not constant throughout the entire experiment.

There was a significant difference in the way the triage labels were distributed. While the Suricata baseline rated 74.78% of all the flows as benign and 23.93% as noisy, the final LLM output classified about 98.13% of the flows as benign and just 1.81% as noisy. This contrast was especially clear in Pilot4, since many of the flows that had been marked as noisy by Suricata were now classified as benign by the LLM. Likewise, though to a smaller extent, similar patterns were seen in the other pilots. The discrepancy thus did not stem mainly from individual decisions regarding attack classes, but from a more general difference in how the two methods interpreted traffic that Suricata had regarded as noisy or suspicious.

The impact of the two-stage LLM design on the final result distribution was also significant. In the first stage, 22,455 flows were identified as suspicious or malicious and were then passed on for further examination. Following the second stage of analysis, 20,358 of the flows that had been flagged were classified as benign, 1,033 were deemed noisy, 654 remained suspicious and 410 remained malicious. This indicates that stage one acted as a broad screening stage, while stage two carried out a more selective reevaluation of the candidate flows before the final LLM result was arrived at.

The Stage-2 results also showed that the LLM could generate more specific interpretations of the attack types when further analysis was asked for. Interpretations of the types such as FTP-Patator, Web Attack Brute Force, Web Attack Sql Injection, Bot, DDoS, Heartbleed, and Infiltration appeared in the Stage-2 output. However, these assignments should be regarded as security interpretations produced by the model rather than as confirmed identifications of attacks, since the experiment had not used an independent flow-level ground-truth mapping to verify each prediction.

The cross-classification results also indicated that disagreement happened in both directions: a large number of flows identified as Suricata-noisy or Suricata-suspicious were regarded as benign by the LLM, while some flows classified as Suricata-benign were given by the LLM predictions marking them as noisy, suspicious, malicious, or as belonging to a specific attack class. Therefore, the LLM didn't just reproduce the

Suricata baseline; rather, the two systems arrived at overlapping but different interpretations of the flow-level evidence.

The results support the main aim of the experimental design, which was to compare a conventional rule-based IDS baseline with an independently generated LLM interpretation without exposing the model to Suricata-derived labels when making predictions. The patterns of agreement and disagreement thus serve as evidence that the bias-controlled workflow had allowed for a meaningful comparison between the two methods of analysis. It must, however, be regarded as showing only comparative behaviour and not as proof that one of the methods is more accurate than the other.

6.2 Suricata versus LLM Agreement and Disagreement

When the Suricata-versus-LLM comparison was carried out, it was found that the degree of agreement varied according to the type of traffic in each pilot. The rate of agreement for the triage labels varied from 53.76% in Pilot4 to 85.35% in Pilot7; Pilots 5 and 6 had similar agreement levels of about 81%, while Pilot8 had about 73%. The rates of agreement for the attack classes followed a very similar pattern, which shows that the main differences between the two methods were already apparent at the higher triage level.

In Pilot4 the lowest level of agreement was achieved since Suricata deemed a large number of the flows to be noisy. Of the 207,126 flows which Suricata had classified as noisy in this pilot, 206,092 were regarded as benign by the final LLM. Furthermore, out of the 4,489 Suricata-suspicious flows, 4,406 were classified as benign. These cross-classification patterns account for a great deal of the relatively low agreement seen in Pilot4 and show that the two methods had applied considerably different interpretations of the traffic that Suricata had regarded as non-benign but not strongly malicious.

The same pattern was found in the other pilots, even though the extent of it varied. In Pilot5 there were 49,190 flows classified as LLM-benign among those identified as Suricata-noisy, 44,971 in Pilot6, 37,464 in Pilot7 and 131,058 in Pilot8. A similar number of flows classified as LLM-benign were also amongst those classified as Suricata-suspicious, namely 8,986 in Pilot5, 3,817 in Pilot6, 3,773 in Pilot7 and 3,810 in Pilot8. This consistent pattern indicates that the LLM was considerably more likely to give a benign rating to traffic which the Suricata baseline had labelled as noisy or suspicious.

Disagreement also took place in the other direction. A number of flows which Suricata had classified as benign were given non-benign labels by the LLM. Pilot6 gives the most clear-cut example, since 6,246 flows that were deemed benign by Suricata were classified as noisy, 249 as suspicious, and 241 as malicious. In Pilot8, 6,516 flows that were considered benign by Suricata were classified as noisy, 58 as suspicious, and 23 as malicious. It is important to note that these cases demonstrate the LLM did not merely decrease the number of non-benign classifications; it also identified a smaller

number of flows as possibly concerning even though the Suricata baseline had labelled them as benign.

The attack-class comparison showed the same asymmetric behavior. Large numbers of Suricata-Unknown flows were classified as BENIGN by the LLM, including 210,498 flows in Pilot4, 58,176 in Pilot5, 48,788 in Pilot6, 41,237 in Pilot7, and 134,868 in Pilot8. At the same time, the LLM assigned specific attack classes such as FTPPatator, Web Attack - Brute Force, Web Attack - Sql Injection, DDoS, Heartbleed, and Bot to some flows that Suricata had categorized as either Benign or Unknown.

A similar situation was seen in the case of particular Suricata attack types. In Pilot7, 73 out of the 88 Suricata PortScan flows were regarded as BENIGN by the final LLM, the other 15 being labelled Unknown. All 49 of the Suricata Bot flows in Pilot7 were classified as BENIGN. In Pilot8, all four of the Suricata PortScan flows and both of the Suricata Bot flows were considered BENIGN. These examples show a considerable amount of disagreement at the class level for some low-frequency attack categories.

The differences that have been observed should not be taken as direct proof that either of the systems was correct or wrong. Although Suricata was used as a baseline for rule-based comparison, it was not regarded as the established truth, and the LLM outputs were interpretations generated by the model based on the bias-controlled flow evidence. The agreement figures thus reflect the degree of similarity in behaviour between the two methods, while the disagreement patterns show the areas in which their decision mechanisms differed.

From an intrusion-detection perspective, the most important finding is that agreement alone does not fully characterize the relationship between the two systems. Two pilots with similar overall agreement rates may still exhibit different types of disagreement, such as Suricata-noisy to LLM-benign, Suricata-benign to LLM-non-benign, or disagreement over specific attack classes. Consequently, the cross-classification patterns provide a more informative view of system behavior than a single agreement percentage.

6.3 The Effect of the Two-Stage LLM Design

The two-stage LLM architecture had a major impact on the final classification result. In Stage 1, a broad screening was carried out and 22,455 flows were identified as suspicious or malicious during the five pilot studies. These flows were then passed on to Stage 2 for more in-depth analysis by means of smaller request chunks and a more detailed output structure.

Stage-2 significantly reduced the number of flows remaining in the higher-risk triage categories. Of the 22,455 candidate flows, 20,358 were reclassified as benign and 1,033 as noisy. Only 654 remained suspicious and 410 remained malicious. Therefore,

only 1,064 Stage-2 candidates, approximately 4.74% of the candidate set, remained in the two higher-risk categories after deeper review.

This change indicates that Stage-1 operated as a broad screening mechanism, while Stage-2 acted as a refinement stage that reassessed the initially flagged flows using more detailed reasoning requirements. Rather than treating every Stage-1 suspicious or malicious prediction as final, the architecture allowed the model to reconsider those cases before they contributed to the final LLM result set.

The effect of Stage-2 differed across pilots. Pilot6 retained the largest number of higher-risk predictions after review, with 285 suspicious and 251 malicious flows. In the other pilots, most Stage-1 candidates were reclassified to lower-risk categories. This cross-pilot variation suggests that the Stage-2 refinement was influenced by the characteristics of the candidate traffic rather than applying a uniform reduction across all dataset days.

The Stage-2 process also enabled more specific attack-class interpretation. While Stage-1 intentionally restricted detailed attack classification, Stage-2 produced classes such as FTP-Patator, Web Attack - Brute Force, Web Attack - Sql Injection, Bot, DDoS, Heartbleed, and Infiltration. This shows that the two-stage design separated broad triage from more detailed semantic interpretation, allowing the initial screening stage to remain compact while reserving richer output for a smaller subset of candidate flows.

From an implementation perspective, this design also reduced the amount of traffic requiring detailed LLM analysis. Only 22,455 of 1,967,638 total flows, approximately 1.14%, proceeded to Stage-2. This reduced the number of high-detail requests and concentrated the more expensive analysis on flows that had already been identified as potentially concerning during Stage-1.

However, the Stage-2 refinement should not be interpreted as proving that the Stage-1 predictions were incorrect. Stage-2 represents a second model-based assessment using a more detailed prompt and output requirement, not an independent ground-truth verification step. The large number of candidates reclassified as benign or noisy therefore demonstrates the effect of the two-stage decision process rather than a measured Stage-1 error rate.

Overall, the two-stage design provided a practical separation between broad screening and detailed candidate review. It reduced the number of final higher-risk predictions, enabled more specific attack-class interpretation, and limited deeper LLM processing to a small fraction of the complete flow population.

6.4 Effect of Bias-Controlled Input Construction

A central methodological objective of this study was to prevent the LLM from receiving Suricata-derived labels or alert information during prediction. The LLM input was therefore constructed only from selected flow-level and protocol-level evidence, while

Suricata alert signatures, categories, severities, baseline labels, and previous LLM predictions were excluded. A dedicated bias-control check was applied before the input was submitted to the model.

This separation is important for interpreting the comparison results. If Suricata labels had been included in the LLM input, agreement between the two systems could have resulted partly from information leakage rather than independent analysis. By excluding those fields, the final comparison reflects two separately generated interpretations of the reconstructed flow evidence: a rule-based Suricata baseline and an LLM-based interpretation produced from the controlled input representation.

The observed disagreement patterns provide practical support for the effectiveness of this separation. The LLM did not simply reproduce the Suricata baseline. Large numbers of Suricata-noisy and Suricata-suspicious flows were classified as benign by the LLM, while some Suricata-benign flows received non-benign LLM classifications. Specific attack-class assignments produced during Stage-2 also differed from Suricata attack-class assignments in a number of cases. These differences are consistent with the two analysis paths operating independently rather than the LLM copying Suricata-derived labels.

The bias-controlled design also improved the interpretability of the final comparison. Because the Suricata baseline remained isolated in DB-2 and the LLM results were stored independently in DB-3, agreement and disagreement could be evaluated only after both processing paths had been completed. This database separation reduced the risk of accidental label contamination during model inference and preserved clear provenance for the two result sources.

However, the term bias-controlled should not be interpreted as meaning that the LLM analysis was completely unbiased. The model itself was pretrained on external data and may contain prior knowledge of network-security concepts, protocol behavior, attack terminology, or datasets similar to CIC-IDS2017. In addition, the selection and formatting of input fields, prompt wording, label definitions, and two-stage decision rules were designed by the researcher and may influence model behavior.

The bias-control procedure therefore addresses a specific experimental risk: leakage of Suricata-derived information into the LLM prediction path. It does not eliminate all possible sources of model or experimental bias. Within that defined scope, the approach provided a clearer basis for evaluating how an LLM interprets network-flow evidence independently of the rule-based baseline.

Overall, the controlled input construction and database separation were essential to the validity of the comparative design. They allowed the observed agreement and disagreement patterns to be interpreted as differences between two independently executed analysis approaches rather than as a consequence of directly supplying one system's classifications to the other.

6.5 Implications for LLM-Assisted Intrusion Detection

The results of this study suggest that LLMs may be useful as an additional analytical component within intrusion-detection workflows rather than as a direct replacement for conventional rule-based systems. Suricata and the LLM frequently agreed on benign traffic, but they showed substantial differences in the interpretation of noisy, suspicious, and attack-related flows. These differences indicate that the two approaches provide distinct forms of analysis that may be complementary in a broader security-monitoring process.

One potential role for an LLM is secondary review of traffic that has already been identified as requiring further attention. The two-stage architecture used in this study demonstrates such a workflow. Stage-1 performed broad triage across all flows, while Stage-2 applied more detailed analysis only to a small candidate subset. Because only approximately 1.14% of all processed flows proceeded to Stage-2, detailed model-based analysis could be concentrated on a limited portion of the traffic rather than applied with the same depth to every flow.

The results also indicate that LLM-based review may help reduce the volume of flows retained as higher-risk candidates. Of the 22,455 Stage-2 candidates, only 1,064 remained classified as suspicious or malicious after deeper review. In an operational environment, a similar design could potentially support prioritization by reducing the number of flows requiring further analyst attention. However, this interpretation should be treated cautiously because the present study did not measure analyst workload, operational response time, or the correctness of the Stage-2 reclassifications against verified ground truth.

Another potential advantage of LLM-assisted analysis is the ability to produce richer explanatory output. Stage-2 predictions included confidence information, supporting evidence, reasoning, and more specific attack-class interpretations. Such structured explanations could support security analysts by providing additional context for why a flow was considered suspicious or associated with a particular attack type. This capability differs from a purely rule-triggered alert and may be valuable for investigation and prioritization tasks.

At the same time, the observed disagreement with Suricata demonstrates why LLM output should not be accepted without validation or independent review. Some flows classified as benign by Suricata received non-benign LLM labels, while many Suricata-noisy or suspicious flows were classified as benign by the LLM. Specific attack-class disagreements were also observed. Without verified ground truth, these differences cannot be interpreted as evidence that the LLM consistently improved or degraded detection quality.

The implementation therefore supports a human- or system-in-the-loop role for LLMs in intrusion detection. A conventional IDS can continue to provide deterministic rule-

based monitoring, while an LLM may provide an additional interpretation layer for selected traffic. Structured validation, traceable model runs, controlled inputs, and separation of baseline and LLM results are important requirements if such a system is to remain auditable and reproducible.

The findings also emphasize the importance of output validation. During this study, model responses were checked for malformed structure, missing flow identifiers, duplicate predictions, invalid categorical values, and incomplete outputs before being accepted. This suggests that practical LLM-assisted IDS systems require validation and retry mechanisms as part of the processing architecture rather than treating the raw model response as a reliable final result.

Overall, the experiment indicates that LLMs can contribute additional triage, contextual interpretation, and candidate-review capabilities to network intrusion analysis. Their most defensible role based on the present results is as a complementary analytical layer operating alongside established IDS mechanisms, with explicit bias controls, structured validation, provenance tracking, and independent evaluation of the resulting predictions.

6.6 Discussion in Relation to the Research Questions

The findings of this study can be related directly to the research questions introduced in Section 1.5. The experimental workflow, implementation results, and Suricata-versus-LLM comparison provide evidence for answering each question within the scope of the study.

RQ1: How can LLM input be constructed as bias-controlled to exclude decision information from Suricata alerts and labels?

The workflow addressed this question through explicit input whitelisting and exclusion rules. Only selected fields from reconstructed flow and protocol information were made available to the LLM. Suricata alert signatures, alert categories, severity values, baseline labels, and previous model predictions were excluded from the LLM evidence. A dedicated bias-control check was performed before model inference, and the Suricata baseline remained stored separately from the LLM results until DB-4 construction. This design controlled the specific risk of Suricata-derived label leakage into the LLM prediction path. However, it does not imply that the LLM itself was free from all forms of model or experimental bias.

RQ2: What is the agreement rate of the triage label and attack class between Suricata baseline results and bias-controlled LLM prediction results?

The results showed substantial but non-uniform agreement across the five pilots. Triage-label agreement ranged from 53.76% in Pilot4 to 85.35% in Pilot7. Pilot5 and Pilot6 achieved agreement rates of approximately 81%, while Pilot8 achieved approximately 73%. Attack-class agreement followed a closely similar pattern. The

variation between pilots indicates that agreement depended strongly on the traffic composition of the individual dataset days rather than remaining constant across the experiment. These values represent agreement between two independently generated classifications and should not be interpreted as LLM accuracy because the Suricata baseline was not treated as verified ground truth.

RQ3: How does the LLM classify flows that Suricata marks as noisy, suspicious, or malicious?

The cross-classification results showed that the final LLM frequently assigned benign to flows labelled noisy or suspicious by Suricata. For example, 206,092 Suricata-noisy flows in Pilot4 and 131,058 in Pilot8 were classified as benign by the LLM. Similar behavior occurred for Suricata-suspicious flows across all five pilots. Some specific Suricata attack classifications were also reassessed by the LLM; for example, most Suricata PortScan flows and all Suricata Bot flows in Pilot7 were not retained as corresponding specific attack classes by the final LLM.

The disagreement also occurred in the opposite direction. A smaller number of Suricata-benign flows received noisy, suspicious, malicious, or specific attack-class interpretations from the LLM. This demonstrates that the LLM did not merely reproduce or uniformly reduce Suricata classifications. Instead, it produced a distinct interpretation of the available flow-level and protocol-level evidence.

RQ4: What are the limitations of using LLMs for flow-level intrusion detection and in reasoning from protocol-level evidence?

Several limitations were identified. The LLM received selected flow-level and protocol-level information rather than complete packet payloads, meaning that potentially relevant attack evidence was unavailable for some flows. Predictions were dependent on the selected model, prompt design, output schema, input representation, and two-stage candidate-selection procedure. Stage-2 reviewed only flows classified as suspicious or malicious during Stage-1; therefore, a potentially important flow classified as benign or noisy in Stage-1 could not be reconsidered during Stage-2. In addition, the experiment used one final LLM model and did not evaluate predictions against independently verified flow-level ground truth. These limitations restrict conclusions about detection accuracy and generalizability and are examined further in Chapter 7.

Overall, the research questions were addressed through the implemented four-database architecture, controlled LLM input construction, two-stage prediction process, structured output validation, and cross-pilot comparison. The results demonstrate that a bias-controlled LLM analysis path can be constructed independently of a rule-based IDS baseline and can produce measurable agreement and disagreement patterns at flow level. At the same time, the findings emphasize that such comparisons should be interpreted as differences in analytical behavior rather than as evidence that either approach is inherently more accurate.

7 Limitations

7.1 Suricata Baseline Is Not Ground Truth

Although CIC-IDS2017 provides official flow labels, these labels were not mapped to the Suricata-reconstructed `flow_id` records used in the final comparison. Therefore, the Suricata baseline is treated as an independent comparison baseline rather than verified ground truth for the reconstructed flows. Agreement between Suricata and the LLM therefore cannot be interpreted as LLM accuracy. The study measures agreement, disagreement, and cross-classification behavior rather than conventional accuracy, precision, recall, or F1-score.

7.2 Limited Input Evidence

The LLM input contains selected flow-level and protocol-level fields rather than complete packet payloads or full network context. As a result, some evidence required to identify an attack may not be available to the model. Attacks that depend on payload content, temporal relationships, or broader host behavior may therefore be difficult to interpret from the compact flow representation alone.

7.3 Prompt and Model Dependency

LLM outputs depend on the selected model, prompt instructions, label definitions, and response schema. The final experiment used meta-llama-3.1-8b-instruct, and different models or prompt formulations could produce different classifications. Temperature was set to 0 to reduce unnecessary variation, but this does not guarantee identical behavior under every inference environment.

7.4 Stage-2 Candidate Selection Limitation

Stage-2 reviews only flows classified as suspicious or malicious during Stage-1. Flows classified as benign or noisy do not receive deeper Stage-2 analysis. Therefore, if Stage-1 assigns a potentially important flow to a lower-risk class, the two-stage design does not provide a later opportunity to reconsider that flow.

7.5 Context-Window and Chunk-Size Constraints

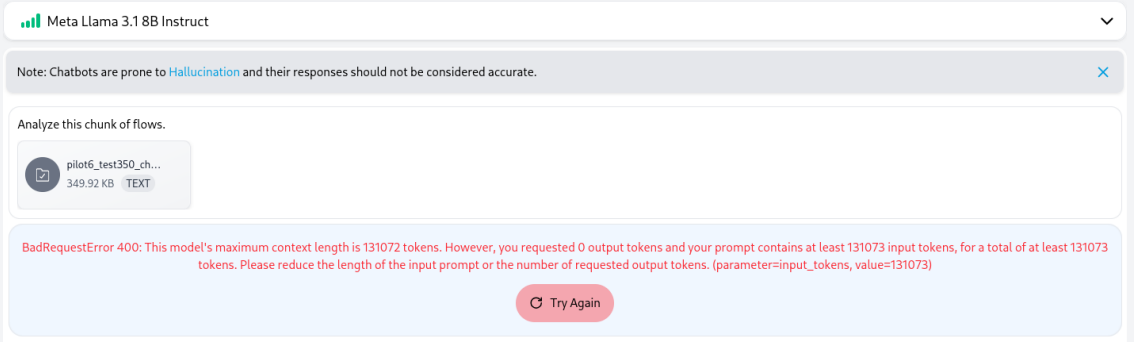

Figure 7.1: Token limit issue

Large LLM input requests were limited by the model context window. Preliminary tests produced BadRequestError 400 responses when the input exceeded the model's 131,072-token context length. Reducing the requested output tokens did not solve the problem when the input itself remained too large. These tests showed that large file size or large numbers of flows could not be processed reliably in a single request and motivated the use of smaller flow-based chunks.

7.6 API Rate Limits and Provider Dependency

The LLM processing pipeline was also affected by provider-side API restrictions. During batch processing, some requests returned 429 API rate limit exceeded errors. For example, one Pilot7 processing run completed 552 of 668 chunk requests successfully while 116 requests failed, including rate-limit failures. Consequently, large-scale processing depended not only on the model but also on the availability and request limits of the external KISSKI API.

7.7 LLM Output Instability and Missing Predictions

A successful API request did not always produce a complete prediction for every submitted flow. For example, during Pilot6 testing, validation identified missing predictions and incomplete chunk outputs even when the request itself had completed. In one 10,500-flow validation batch, 10,394 canonical predictions were initially recovered and 106 flows were identified as missing. Retry processing and canonicalization were therefore required to recover the complete set of 10,500 unique flow predictions.

7.8 Validation and Retry Overhead

The need for parsing, canonicalization, missing-flow detection, and selective retry increased the complexity of the implementation. Model responses could contain missing records, malformed structure, unexpected output, or incomplete chunks. The final results therefore depended on an additional validation layer rather than directly using raw LLM responses. Although this improved result completeness and traceability, it added processing overhead and additional implementation logic. The validation

process checked structural correctness and output completeness, but structurally valid output can still contain a semantically inappropriate attack class (e.g., LDAP, SMB). So, future validation should use a strict attack-class whitelist and either reject unexpected values, retry them, or map them to the Unknown class.

7.9 Dataset and Evaluation Scope

The final evaluation was conducted on five pilots derived from CIC-IDS2017. Although the pilots represent different traffic days and attack conditions, the dataset is a controlled benchmark and does not fully represent contemporary production networks. The results therefore demonstrate the behavior of the proposed workflow under the selected experimental conditions and should not be generalized directly to all operational network environments.

8 Conclusion and Future Work

8.1 Conclusion

This study evaluated the use of a Large Language Model for flow-based network traffic analysis in comparison with a conventional rule-based intrusion detection baseline. The experimental workflow was developed using Suricata EVE JSON output from five selected CIC-IDS2017 pilots and was designed to keep the Suricata baseline and LLM analysis paths separate until the final comparison stage.

A four-database architecture was implemented to support this separation. DB-1 preserved raw EVE-derived events, DB-2 stored reconstructed flow-level information and the Suricata baseline, DB-3 stored LLM input metadata, model-run information, and validated Stage-1 and Stage-2 predictions, and DB-4 contained the final Suricata-versus-LLM comparison. This architecture provided traceability across the complete processing pipeline while reducing the risk that Suricata-derived classifications could influence the LLM during inference.

The LLM input was constructed using selected flow-level and protocol-level evidence while excluding Suricata alert signatures, alert categories, severity values, and baseline labels. A dedicated bias-control check was applied before LLM processing. This design enabled the LLM to generate its classifications independently of the Suricata output and allowed the final comparison to focus on agreement and disagreement between two separately executed analysis approaches.

Across the five pilots, a total of 1,967,638 flows were processed. Stage-1 generated one validated LLM prediction for every flow and classified 22,455 flows as suspicious or malicious, which were subsequently forwarded to Stage-2. Stage-2 provided deeper candidate review and reduced this set to 654 suspicious and 410 malicious predictions. The final LLM result therefore contained 1,930,912 benign, 35,662 noisy, 654 suspicious, and 410 malicious classifications.

The final Suricata-versus-LLM comparison showed that agreement varied considerably across the pilots. Triage-label agreement ranged from 53.76% in Pilot4 to 85.35% in Pilot7, while Pilot5 and Pilot6 achieved approximately 81% agreement and Pilot8 approximately 73%. A major source of disagreement was the treatment of traffic classified as noisy or suspicious by Suricata, much of which was classified as benign by the LLM. Disagreement also occurred in the opposite direction, with some Suricata-benign flows receiving non-benign LLM classifications.

The results demonstrate that the LLM did not simply reproduce the Suricata baseline. Instead, the two approaches produced overlapping but distinct interpretations of the same reconstructed flow evidence. The two-stage architecture further demonstrated that broad LLM triage could be followed by a more selective review of a relatively small

candidate set, while structured validation and retry procedures were necessary to obtain complete and consistent LLM outputs.

The study therefore achieved its primary objective of constructing and evaluating a reproducible, bias-controlled workflow for comparing rule-based IDS output with LLM-based flow interpretation. However, because Suricata was used as an independent baseline rather than verified ground truth, the observed agreement and disagreement cannot be interpreted as direct evidence that either approach is more accurate. The principal contribution of the work is therefore the experimental architecture and comparative methodology, together with empirical evidence showing how a bias-controlled LLM analysis can differ from a conventional IDS baseline at flow level.

8.2 Future Work

Ground-truth-based evaluation: Future work should map the reconstructed Suricata flows to the official CIC-IDS2017 flow labels. This would allow both Suricata and LLM predictions to be evaluated using conventional measures such as precision, recall, F1-score, false-positive rate, and false-negative rate rather than relying only on agreement between the two systems.

Multi-model LLM evaluation: The final experiment used meta-llama-3.1-8b-instruct. Future studies should compare multiple LLMs under the same bias-controlled input conditions to determine whether the observed behavior is model-specific or consistent across different architectures and model sizes.

Improved Stage-2 candidate selection: The current design sends only Stage-1 suspicious and malicious flows to Stage-2. Future work could also review selected benign and noisy flows based on confidence, uncertainty, or sampling so that potentially important flows are not excluded from deeper analysis too early.

Richer bias-controlled network evidence: Future input construction could include additional temporal, bidirectional, host-level, and protocol-context information while continuing to exclude Suricata-derived labels and alert information. This may improve interpretation of attacks that cannot be identified reliably from individual compact flow records.

Automated reliability and retry handling: The pipeline could be extended with automatic token estimation, dynamic chunk sizing, API rate-limit handling, missing-flow detection, selective retry, and final completeness verification. This would reduce the manual effort required to recover from context-window errors, rate-limit failures, and incomplete LLM responses.

Real-world and operational evaluation: The workflow should be evaluated on more recent and realistic network traffic in addition to CIC-IDS2017. Future experiments should also measure latency, throughput, computational requirements, API cost, scalability, and suitability for practical IDS or security-operations environments.

References

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013, doi: 10.1016/j.jnca.2012.09.004.

[2] “1. What is Suricata — Suricata 9.0.0-dev documentation.” Accessed: Aug. 13, 2026. [Online]. Available: https://docs.suricata.io/en/latest/what-is-suricata.html

[3] “8.1. Rules Format — Suricata 9.0.0-dev documentation.” Accessed: Aug. 22, 2026. [Online]. Available: https://docs.suricata.io/en/latest/rules/intro.html

[4] “18. Protocols — Suricata 9.0.0-dev documentation.” Accessed: Aug. 22, 2026. [Online]. Available: https://docs.suricata.io/en/latest/protocols/protocols.html

[5] “15.1.1. Eve JSON Output — Suricata 9.0.0-dev documentation.” Accessed: Aug. 13, 2026. [Online]. Available: https://docs.suricata.io/en/latest/output/eve/eve-json-output.html

[6] “15.3. Syslog Alerting Compatibility — Suricata 9.0.0-dev documentation.” Accessed: Aug. 22, 2026. [Online]. Available: https://docs.suricata.io/en/latest/output/syslog-alerting-comp.html

[7] Y. Chen *et al.*, “A survey of large language models for cyber threat detection,” *Computers & Security*, vol. 145, p. 104016, Oct. 2024, doi: 10.1016/j.cose.2024.104016.

[8] S. Yang *et al.*, “Large Language Models for Network Intrusion Detection Systems: Foundations, Implementations, and Future Directions,” Jul. 07, 2025, arXiv:2507.04752. doi: 10.48550/arXiv.2507.04752.

[9] L. Huang *et al.*, “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions,” *ACM Trans. Inf. Syst.*, vol. 43, no. 2, pp. 1–55, Mar. 2025, doi: 10.1145/3703155.

[10] H. W. Njogu, L. Jiawei, J. N. Kiere, and D. Hanyurwimfura, “A comprehensive vulnerability based alert management approach for large networks,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 27–45, Jan. 2013, doi: 10.1016/j.future.2012.04.001.

[11] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[12] “IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB.” Accessed: Aug. 13, 2026. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[13] K. A. Scarfone and P. M. Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS),” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[14] N. A. Alrajeh, S. Khan, and B. Shams, “Intrusion Detection Systems in Wireless Sensor Networks: A Review,” *International Journal of Distributed Sensor Networks*, vol. 9, no. 5, p. 167575, May 2013, doi: 10.1155/2013/167575.

[15] “About SQLite.” Accessed: Aug. 13, 2026. [Online]. Available: https://www.sqlite.org/about.html

[16] “What is ChatGPT: FAQ” Accessed: Aug. 26, 2026. [Online]. Available: https://help.openai.com/en/articles/12677804-what-is-chatgpt-faq

Declaration

I hereby declare that I have written this research project report independently and without unauthorized assistance. All sources and materials used in this work have been properly acknowledged. Any parts of the work that have been taken directly or indirectly from published or unpublished sources have been clearly identified as such.

I further declare that this report has not been submitted, either in the same or in a substantially similar form, to any other examination authority.

Essen, 26.08.2026

Place, Date

Signature