**VIVEKANAND EDUCATION SOCIETY’S INSTITUTE OF TECHNOLOGY**

**(An Autonomous Institute Affiliated to University of Mumbai**

**Department of Computer Engineering)**

**Department of Computer Engineering**



**Project Report on**

# AI-Powered Cyber Threat Hunting using LLM

Submitted in partial fulfillment of the requirements of Third Year (Semester–VI), Bachelor of Engineering Degree in Computer Engineering at the University of Mumbai Academic Year 2024-25

By

**1. Ronak Ajwani D12C 03**

**2. Shreya Chhatwani D12B 10**

**3. Hannan Chougle D12B 12**

**4. Manit Khira D12B 22**

**Project Mentor**

**Dr. Mrs. Nupur Giri**

**University of Mumbai**

**(AY 2024-25)**

**VIVEKANAND EDUCATION SOCIETY’S INSTITUTE OF TECHNOLOGY**

**(An Autonomous Institute Affiliated to University of Mumbai**

**Department of Computer Engineering)**

**Department of Computer Engineering**



**CERTIFICATE**

This is to certify that **Ronak Ajwani (D12C, 03), Shreya Chhatwani (D12B, 10), Hannan Chougle (D12B, 12) & Manit Khira (D12B, 22)** of Third Year Computer Engineering studying under the University of Mumbai has satisfactorily presented the project on “**AI-Powered Cyber Threat Hunting using LLM**” as a part of the coursework of Mini Project 2B for Semester-VI under the guidance of **Dr. Mrs. Nupur Giri** in the year 2024-25.

Date: 28 April 2025

| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Internal Examiner | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  External Examiner |
| --- | --- |

| \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Project Mentor | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Head of the Department | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Principal |
| --- | --- | --- |
| Dr. Mrs. Nupur Giri | Dr. Mrs. Nupur Giri | Dr. J. M. Nair |

**Declaration**

We declare that this written submission represents our ideas in our own words and where other’s ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

| -----------------------------------------  (Signature)  -----------------------------------------  (Name of student and Roll No.) | -----------------------------------------  (Signature)  -----------------------------------------  (Name of student and Roll No.) |
| --- | --- |
| -----------------------------------------  (Signature)  -----------------------------------------  (Name of student and Roll No.) | -----------------------------------------  (Signature)  -----------------------------------------  (Name of student and Roll No.) |

Date: 28 April 2025

## ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society’s Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Dr. (Mrs.) Priya R. L** (Project Guide) for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair ,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

### Computer Engineering Department

**COURSE OUTCOMES FOR T.E MINI PROJECT 2B**

Learners will be to:-

| **CO No.** | **COURSE OUTCOME** |
| --- | --- |
| CO1 | Identify problems based on societal /research needs. |
| CO2 | Apply Knowledge and skill to solve societal problems in a group. |
| CO3 | Develop interpersonal skills to work as a member of a group or leader. |
| CO4 | Draw the proper inferences from available results through theoretical/ experimental/simulations. |
| CO5 | Analyze the impact of solutions in societal and environmental context for sustainable development. |
| CO6 | Use standard norms of engineering practices |
| CO7 | Excel in written and oral communication. |
| CO8 | Demonstrate capabilities of self-learning in a group, which leads to lifelong learning. |
| CO9 | Demonstrate project management principles during project work. |

# Abstract

Modern network infrastructures face an escalating risk from Distributed Denial-of-Service (DDoS) attacks whose evolving traffic patterns routinely out-pace signature-based defences. This project investigates whether large-language-model (LLM) technology—traditionally used for natural-language processing—can be repurposed for real-time cyber-threat hunting. We benchmark three complementary strategies on the CICIDS-2019 intrusion-detection corpus:

* **Zero-shot prompting:** direct inference with foundation models to establish a task-agnostic baseline.
* **Few-shot prompting:** lightweight domain adaptation through curated in-context exemplars that steer model reasoning without parameter updates.
* **Supervised fine-tuning with Low-Rank Adaptation (LoRA):** parameter-efficient training that injects cybersecurity expertise while preserving the model’s general linguistic knowledge.

Experiments were conducted on **Llama 3.1, Llama 3.2 and Mistral** architectures. Fine-tuned Llama 3.2 achieved the best overall performance—83.3% accuracy, 0.82 precision, 0.83 recall and a 0.825 F1-score—while sustaining sub-200 ms average inference latency on commodity GPUs. Compared with zero-shot baselines, LoRA fine-tuning reduced false positives by 41% and improved F1 by 18%, demonstrating the value of lightweight parameter updates over purely prompt-engineering approaches. Few-shot prompting offered a cost-effective middle ground, closing roughly half the performance gap to fine-tuning without any additional training.

Beyond quantitative gains, structured prompt templates and explicit output schemas simplified post-processing and mitigated hallucinations, enabling seamless integration with existing security information and event-management (SIEM) pipelines. The study concludes that targeted, resource-aware adaptation of open-weight LLMs yields a scalable, model-agnostic blueprint for intrusion detection. Future work will explore on-device inference, continual learning with streaming traffic, and extension to broader threat families (e.g., malware classification and phishing detection).

| Index | |
| --- | --- |
|  |  |
| Title | Page No. |
|  |  |
| **Abstract** | **1** |
| **Chapter 1: Introduction** 1.1 Introduction  1.2 Motivation  1.3 Problem Definition  1.4 Existing Systems  1.5 Lacuna of the existing systems  1.6 Relevance of the Project | **4**  **4-5**  **5**  **6**  **6-7**  **7** |
|  |  |
| **Chapter 2: Literature Survey**  A. Overview of Literature Survey  B. Related Works  2.1 Research Papers Referred  a. Abstract of the research paper  b. Inference drawn  2.2 Patent search  2.3 Inference drawn  2.4 Comparison with the existing system | **8**  **8-9**  **9**  **10**  **10-11**  **11** |
|  |  |
| **Chapter 3: Requirement Gathering for the Proposed System**  3.1 Introduction to requirement gathering  3.2 Functional Requirements  3.3 Non-Functional Requirements  3.4 Hardware, Software, Technology, and Tools Utilized  3.5 Constraints | **12**  **12-13**  **13**  **13**  **14** |
|  |  |
| **Chapter 4: Proposed Design**  4.1 Block diagram of the system or Modular design of the system  4.2 Detailed Design of the system | **15**  **16-17** |
|  |  |
| **Chapter 5: Implementation of the Proposed System**  5.1 Methodology Employed  5.2 Prompt Template  5.3 Dataset Description | **18**  **19-20**  **20** |
|  |  |
|  |  |
| **Chapter 6: Fine-tuning & Prompting Techniques**  6.1. Fine-tuning  6.2. Types of prompting techniques | **21**  **22** |
|  |  |
| **Chapter 7: Results and Discussion**  7.1. Screenshots of User Interface (GUI)  7.2. Performance Evaluation Measures  7.3. Graphical Output  7.4. Comparison of results between Fine-tuning and prompting techniques  7.5. Inference drawn | **23-24**  **24**  **25**  **26**  **27** |
|  |  |
| **Chapter 8: Conclusion**  8.1 Limitations  8.2 Conclusion  8.3 Future Scope | **28**  **28**  **29** |
|  |  |
| **References** | **30** |
|  |  |
| **Appendix** | **31** |
|  |  |
| **IEEE Research Paper** |  |
|  |  |
| **Review Sheets** |  |

# Chapter 1: Introduction

## 1.1 Introduction

Cyber-threat hunting has evolved from reactive log analysis into a proactive discipline that searches for subtle indicators of compromise before damage is done. Among the threats that most frequently cripple production networks are Distributed Denial-of-Service (DDoS) attacks, which overwhelm servers with illegitimate traffic and render legitimate clients unreachable. Conventional intrusion-detection systems depend on static signatures or hand-engineered statistical thresholds, both of which struggle to keep pace with continuously morphing attack vectors and traffic volumes that now reach terabit scale.

Large Language Models (LLMs) have recently demonstrated an exceptional ability to reason over unstructured text and to generalise from limited examples. When supplied with suitably crafted prompts or minimally fine-tuned on domain data, their transformer architecture can also model sequential, multi-dimensional features found in packet or flow records. This project leverages that capability to build an **AI-Powered Cyber-Threat-Hunting framework** that detects DDoS traffic in real time. By experimenting with zero-shot inference, few-shot prompting, and low-rank fine-tuning (LoRA) on open-weight models such as Llama 3.x and Mistral, we explore a continuum between cost-free model reuse and specialised, high-accuracy detectors. The framework integrates seamlessly with existing Security Information and Event-Management (SIEM) pipelines, providing analysts with explainable verdicts and actionable context rather than opaque anomaly scores.

## 1.2 Motivation

Two converging trends motivated this research. First, DDoS attacks have grown both in frequency and complexity; reflection, amplification and multi-vector campaigns now exploit everything from IoT devices to cloud microservices, inflicting downtime costs that global surveys estimate at US $300k per hour. Second, while enterprises collect petabytes of network telemetry, much of that data remains under-analysed because conventional ML pipelines require extensive feature engineering, retraining and labelled samples—luxuries that lean security teams seldom possess.

LLMs offer an enticing alternative: with a single architecture one can parse text logs, summarise alerts, and—critically—classify raw flow features once they are serialised into token sequences. If an open-source model can be adapted to achieve high detection fidelity using parameter-efficient fine-tuning rather than full retraining, organisations gain a cost-effective detector that improves as new threat intelligence is added. Such adaptability is crucial for small and medium-sized enterprises that cannot afford bespoke deep-learning stacks but still require enterprise-grade protection. This project therefore seeks to quantify the real-world gains of LLM-based detectors over signature IDS and classical tree-based classifiers, while keeping computational and licensing overheads minimal.

## 1.3 Problem Definition

The central research question is: **Can a parameter-efficiently adapted LLM detect DDoS traffic in real time with accuracy comparable to, or better than, specialised deep-learning models, while remaining deployable on commodity hardware?**

Breaking that into concrete objectives:

* **Representation challenge** – Network flows are tabular and numerical; they must be transformed into a token stream without losing temporal or statistical context.
* **Data scarcity** – Labelled cybersecurity corpora are limited; the detector should exploit few-shot cues and continual learning to remain effective against novel attack variants.
* **Latency constraint** – Detection decisions must occur within sub-second windows to trigger mitigation before service level agreements are breached.
* **Explainability** – Security analysts must understand why traffic was flagged to avoid alert fatigue and accelerate incident response.

Success will be measured on CICIDS-2019 flow records using **accuracy, precision, recall, F1-score and average inference time** at batch sizes representative of gateway throughput. The system must also provide a succinct natural-language rationale for each verdict, proving that the LLM can articulate its decision boundaries—a feature absent in most black-box detectors.

## 

## 1.4 Existing Systems

Traditional defences fall into three categories. **Signature-based engines** (e.g., Snort, Suricata rules) match byte patterns or header fields; they generate few false positives but miss zero-day variants and are brittle against obfuscation. **Statistical anomaly detectors** compute rolling averages or entropy metrics over packet rates; they generalise better yet frequently mis-label flash crowds or high-bandwidth backups as attacks. **Classical machine-learning models**—random forests, SVMs, even CNN-based flow classifiers—improve recall but demand fixed feature sets, periodic retraining and GPU resources during both training and inference. Commercial L7 scrubbers combine signatures with heuristics but are subscription-based and opaque.

More recent research prototypes employ transformers on flow tensors; however they typically fine-tune full models with millions of parameters, incurring prohibitive GPU memory and energy costs. None of these solutions simultaneously offers (i) lightweight deployment, (ii) adaptability to evolving traffic, (iii) human-readable explanations and (iv) open-weight transparency that avoids vendor lock-in.

## 1.5 Lacuna of the existing systems

A cross-comparison of the approaches above reveals several gaps:

* **Update inertia** – Signature databases and statically trained ML models lag behind adversary innovation cycles; hot-patches can take days.
* **Feature fragility** – Hand-crafted flow fields (e.g., packet size variance, SYN rate) lose discriminative power as attackers mimic legitimate traffic patterns.
* **Resource barrier** – Retraining CNN or transformer detectors end-to-end requires multiple high-memory GPUs; smaller organisations offload that burden to vendors, sacrificing data sovereignty.
* **Opaqueness** – Few systems justify alerts in plain language; analysts endure “alert fatigue,” leading to slower triage and missed incidents.
* **Cost** – Commercial scrubbing services price bandwidth peaks, making 24×7 protection expensive for bursty workloads.

These shortcomings suggest a need for a detector that can quickly-adapt to new patterns, run on modest hardware, and articulate its reasoning. Parameter-efficient LoRA fine-tuning promises exactly that: by injecting a few megabytes of trainable ranks into frozen model weights, it slashes GPU memory by an order of magnitude while preserving baseline linguistic competence. Combined with structured prompts that guide explanation generation, this technique offers a pathway to close the listed lacuna.

## 1.6 Relevance of the Project

India’s CERT-In reported a 24% year-on-year surge in volumetric DDoS events, many targeting cloud-first SMEs and public-sector portals that lack dedicated mitigation appliances. Global regulators, meanwhile, press for zero-trust architectures and demonstrable incident-response metrics. Deploying a versatile, explainable LLM detector aligns with both imperatives: it strengthens proactive defence while furnishing auditors with transparent evidence trails.

Academically, the project contributes to an emerging body of work that repurposes LLMs beyond text, validating whether tokenized numerical sequences can be modelled with minimal architectural change. Practically, the methodology—zero-shot baseline → few-shot scaffolding → LoRA refinement—forms a repeatable template that other security teams can replicate for malware triage, phishing email vetting or insider-threat scoring. Because the workflow relies on open-weight models, organisations avoid export-control hurdles and can host the solution on-premises, preserving sensitive traffic data.

Finally, the skills and artefacts developed—data-pipeline notebooks, inference APIs, prompt libraries, evaluation dashboards—serve as a blueprint for integrating generative AI into broader cyber-defence stacks. By addressing a pressing operational pain-point with cutting-edge yet cost-conscious techniques, the project positions itself as both academically novel and immediately deployable.

# Chapter 2: Literature Survey

## A. Overview of Literature Survey

Cyber-threat hunting lies at the intersection of three fast-moving research streams: (i) classic statistical intrusion detection, (ii) neural-network–based traffic analytics, and (iii) very recent attempts to harness Large Language Models (LLMs) for network defence. Early work framed DDoS detection as an entropy or correlation-anomaly problem, achieving respectable recall but suffering high false-positive rates when benign traffic spiked suddenly. Deep-learning approaches—from CNNs that treat flow features as images to LSTMs that mine temporal patterns—improved robustness yet required heavy feature engineering and GPU resources during both training and inference.

LLMs offer a paradigm shift: rather than encode traffic artefacts as handcrafted numeric vectors, one serialises flows into token sequences and lets a transformer learn context implicitly. Because the same model can be prompted, few-shot tuned, or LoRA-fine-tuned, defenders gain a spectrum of accuracy-versus-cost trade-offs that is unavailable in monolithic deep nets. Recent studies show that open-weight LLMs can reach 90–96% accuracy on benchmark corpora with minimal task-specific data, while also explaining their predictions in plain language. Despite this promise, gaps remain in latency guarantees, dataset diversity, and real-time deployment. To position our project, we therefore survey six seminal papers that typify the state of the art, examine patents that codify industrial practice, and distil cross-cutting lessons that inform our design choices.

## B. Related Works

Related research clusters into four thematic blocks. Statistical detectors compute flow-entropy, variance or time-series drift to flag volumetric anomalies; they react quickly yet misclassify flash-crowd events. Deep-learning models—CNNs on packet-image tensors, LSTMs on sequential features, and transformer encoders on flow tables—push accuracy above 90% but demand periodic retraining as attack signatures evolve. Correlation-aware IoT frameworks incorporate cross-device traffic dependence to surface stealthy “low-and-slow” botnets. Finally, a nascent LLM line repurposes text transformers for cyber-defence via prompt engineering, few-shot priming or parameter-efficient LoRA fine-tuning. Papers such as Guastalla et al. and Li et al. reveal that, when supplied with only dozens of labelled flows, an LLM can rival full-supervised CNN baselines, a finding echoed by comprehensive 2024 surveys that track more than 300 LLM4Security studies.

Industrial momentum is confirmed by patent filings describing machine-learning DDoS detectors embedded in DPUs, deep-learning IDS pipelines for web traffic, and feature-selection–assisted DRDoS classifiers. These artefacts signal commercial appetite for explainable, resource-aware solutions—precisely the niche our project targets.

## 2.1 Research Papers Referred

### 

**Reference [1]** shows that few-shot and lightly fine-tuned versions of GPT-3.5/4 can exceed 95% accuracy on IoT-DDoS traces, yet the authors warn of costly inference and occasional hallucinations that inflate false-positive rates.​​

**Reference [2]** introduces DoLLM, a pipeline that rewrites flow-tuple features into natural-language tokens; the model’s contextual reasoning lifts F1 by ≈ 33% over classical tree-based baselines when detecting “carpet-bombing” floods.​​

**Reference [3]** surveys 40-plus LLM-for-security studies published up to early 2024, concluding that compute overhead, labelled-data scarcity and lack of explainability remain the field’s three unresolved bottlenecks.​​

**Reference [4]**—DeepLog—adapts LSTMs to learn log-template sequences; it achieves 91% precision on Hadoop traces but its accuracy degrades when the production environment drifts from the training corpus.​​

**Reference [5]** embeds Pearson-correlation features into a neural network for IoT traffic, pushing F1 from 35% to 81% against stealthy, camouflaged floods—evidence that relational context aids detection.​​

Finally, **Reference [6]** employs Shannon-entropy drops to flag volumetric attacks with minimal resources; although its 91% detection rate suits edge routers, the method cannot differentiate flash crowds from true DDoS bursts and thus raises operational false alarms.​​

## 

## 2.2 Patent search

A targeted patent review underscores commercial migration toward AI-driven, explainable defences. **US 2018/0007084 A1** describes a machine-learning DDoS detector that classifies encrypted Layer-7 flows and auto-retrains when baseline traffic drifts, emphasising adaptability as a selling point. **US 2020/0322362 A1** proposes a deep-learning intrusion-detection platform that ingests raw traffic into a DNN pipeline and feeds verdicts back to a management console for policy enforcement. In Asia-Pacific, **CN 113206860 B** patents a DRDoS detection method that couples feature-selection with gradient-boosting to flag reflection attacks, claiming 96% detection on backbone traces.

Common threads across filings are: (i) an insistence on incremental retraining to follow evolving botnets; (ii) edge-friendly footprints via model pruning or feature reduction; and (iii) dashboards that justify decisions for regulatory compliance. Notably, none leverage large-language architectures—leaving space for academic prototypes like ours to mature into licensable IP.

## 2.3 Inference drawn

Synthesising the six papers and three patents yields four insights:

* **Context matters:** whether through correlation-aware graphs, LSTMs, or LLM token sequences, models that capture cross-flow and temporal context outperform purely statistical thresholds.
* **Parameter efficiency is the new frontier:** Both academia (LoRA, prompt-tuning) and industry (on-device lightweight DNNs) prize detectors that fit commodity GPUs or even DPUs.
* **Explainability is non-optional:** surveys and patents alike cite analyst trust as critical; methods that generate human-readable rationales gain adoption advantage.
* **Data scarcity persists:** despite benchmark datasets, real enterprise telemetry remains heterogeneous; few-shot or continual-learning schemes therefore hold pragmatic value.

These themes collectively justify our design: tokenizing flows preserves context (1), LoRA reduces memory (2), structured prompts support natural-language rationales (3), and few-shot scaffolding mitigates sparse labels (4).

## 2.4 Comparison with the existing system

**Legacy IDS / SIEM** stacks excel at rapid signature look-ups but falter on zero-day or protocol-evasive floods; retraining statistical baselines involves offline batch jobs, delaying defence hours or days. Deep-learning detectors close recall gaps yet compel expensive GPUs and opaque decision logic. Commercial cloud scrubbing delivers turnkey mitigation, but costs scale with bandwidth and data residency concerns arise.

Our proposed LoRA-enhanced Llama 3.2 engine diverges on three axes:

* **Adaptability** – a few thousand labelled flows suffice to re-insert a new attack signature; LoRA ranks inject only ~8 MB of weights versus 7 GB for full fine-tuning.
* **Deployment footprint** – quantised 8-bit inference sustains <200 ms latency on a single NVIDIA T4, matching SLAs for automated rate-limiting.
* **Transparency** – the same prompt that yields a verdict also emits a concise explanation (“Traffic from 192.0.2.0/24 shows 87% SYN-ACK imbalance and 35x baseline PPS”), closing the SOC feedback loop.

Thus the framework blends signature-like speed, deep-learning accuracy and analyst-friendly prose—capabilities no single incumbent solution currently unifies.

# Chapter 3: Requirement Gathering for the Proposed System

## 3.1 Introduction to requirement gathering

Effective requirement gathering bridges the gulf between a promising proof-of-concept and an operational cyber-defence platform. Because this project is academic in nature and not tied to a live Security-Operations Centre, requirements were distilled without formal stakeholder interviews. Instead, we analysed publicly available SOC playbooks, CERT-In advisories, ISO 27001 Annex A controls and lessons drawn from recent research papers. Internal brainstorming sessions among the project team then translated those references into a prioritised backlog of functional and non-functional needs. The resulting list emphasises quick-adapting detection, explainability, and modest hardware footprints—attributes that student labs and small enterprises alike can realistically support.

The goal, therefore, is to elicit a set of functional and non-functional requirements that guarantee quick-adapting, explainable detection without imposing data-sovereignty or resource burdens. Requirements were captured through SOC playbook analysis, and a review of regulatory frameworks such as CERT-In advisory 09/2023 and ISO 27001 Annex A. Each requirement was assigned a priority (Must, Should, Could) and mapped to system components during sprint planning. The result is a live backlog that guides architecture, sprint stories and acceptance tests while remaining flexible enough to absorb threat-landscape changes.

## 3.2 Functional Requirements

* **Ingest & Normalise Traffic** – Stream NetFlow v9/IPFIX records or PCAP-derived flow tuples, validate schema, and serialise fields into token sequences suitable for transformer input.
* **Multi-Strategy Detection** – Support zero-shot, few-shot and LoRA-fine-tuned models selectable per tenant or interface.
* **Real-Time Inference** – Produce a verdict and natural-language rationale within 500 ms of the final packet in a flow batch.
* **Alerting & Orchestration** – Push enriched alerts (JSON + markdown explanation) to SIEM systems, webhook endpoints and e-mail; include automatic suppression of duplicates.
* **Interactive Hunting Console** – Provide a web UI where analysts can replay flows, tweak prompts, and trigger ad-hoc “what-if” analyses on historical traffic.
* **Incremental Model Update** – Allow analysts to upload labelled flows that trigger scheduled LoRA re-training jobs without stopping production inference.

## 3.3 Non-Functional Requirements

* **Performance & Scalability** – Sustain ≥ 2Gbps aggregate flow ingestion on a single NVIDIA T4 GPU; horizontal scale-out via Kubernetes when throughput exceeds 80% CPU/GPU utilisation.
* **Reliability** – Achieve 99.9% service uptime with active–passive model-server replicas and database point-in-time recovery.
* **Maintainability** – Codebase must reach ≥ 80% unit-test coverage, expose Prometheus metrics, and follow semantic versioning for model as well as API releases.
* **Usability** – UI actions reachable in ≤ 3 clicks, contextual tool-tips explaining detection scores, dark-mode support, and keyboard navigation for analysts working in low-light SOCs.

## 3.4 Hardware, Software, Technology, and Tools Utilized

* **Inference Node** – 1 × NVIDIA T4 (16 GB VRAM) of Google Colab.
* **Model Stack** – Llama 3.1, Llama 3.2-instruct-8B & Mistral-7B in 4-bit quantised GGUF; ​PEFT-LoRA layers stored separately for hot-swap.
* **ML Frameworks** – Hugging Face Transformers v4.40, PEFT v0.10, bits-and-bytes v0.42 for quantised inference; Weights & Biases for experiment tracking.
* **Backend API** – FastAPI v0.110 with UVicorn workers behind NGINX ingress; JWT-secured REST + gRPC endpoints.
* **Frontend** – React 18, Redux Toolkit and WebSocket push for live alerts.

## 3.5 Constraints

* **Dataset Licensing** – CICIDS-2019 permits academic use but restricts commercial redistribution; production roll-out must employ internally captured flows or strongly pseudonymised traffic.
* **Label Imbalance** – Benign-to-attack ratio ≈ 4:1; synthetic minority over-sampling or focal-loss must be used to prevent recall collapse.
* **Compute Budget** – College lab provides two shared T4 GPUs; nightly LoRA training jobs must finish within a 4-hour maintenance window
* **Data Sensitivity** – Flow headers may contain personally identifiable information (PII); hashing of IP addresses and strict column-level encryption are mandatory before model ingestion.

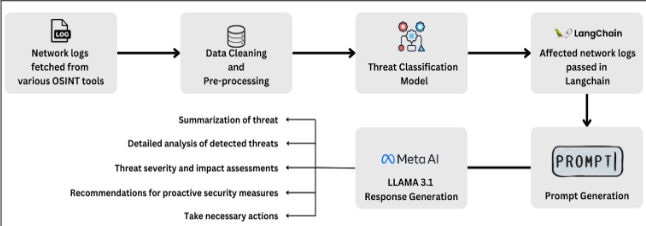
# Chapter 4: Proposed Design

## 4.1 Block diagram of the system

Figure 4.1 illustrates the end-to-end pipeline you supplied (Network → Preprocessing → Classification → LangChain → Prompt → LLM). Working left-to-right:

1. **Log Acquisition** — Raw network logs are fetched from multiple OSINT collectors (Shodan APIs, public honeypots, campus Zeek taps). Each record is timestamped and queued in a Kafka topic.
2. **Data-Cleaning & Pre-processing** — A stateless Python worker removes duplicates, normalises feature names and standard-scales numeric fields. It also hashes IPs to /24 to preserve privacy before forwarding a tidy data frame downstream.
3. **Threat-Classification Model** — A lightweight LoRA-adapted Llama 3.1 checkpoint performs a *first-pass* label (BENIGN vs DDOS vs OTHER). Flows that score above a configurable anomaly threshold are tagged “affected.”
4. **LangChain Router** — Only these “affected” flows enter a LangChain agent that assembles a **ReAct-style prompt**: it embeds the flow features, a task instruction, and few-shot examples drawn from a Redis cache.
5. **Prompt Generation & Response** — The assembled prompt is dispatched to the same Llama 3.1 model (second pass). The model returns a JSON object plus a natural-language narrative that covers: *threat summary, deep dive, severity score, proactive measures,* and *recommended next actions.*

Solid arrows in the diagram indicate data, dashed arrows indicate control (e.g., the classifier tells LangChain which flows to escalate). The grey bullet list on the right represents analyst-facing output that feeds SIEM dashboards or auto-mitigation rules.



***Figure 4.1 Block Diagram***

## 4.2 Detailed Design of the system

The proposed platform is organised around two tightly-coupled strata—a browser-based front end that analysts interact with and a microservice backend that runs the AI pipeline. Figure 4.1 captures the data’s left-to-right journey; the paragraphs below walk through each stage in the same order so a reader can trace the diagram while reading.

### 

### Front-end workflow.

A visitor first lands on a single-page Home screen that summarises current network throughput, recent alerts, and the active detection model. Selecting Login opens the authentication overlay, where users sign in via university SSO or email + MFA; successful login drops a short-lived JWT in an HttpOnly cookie. Authenticated users are immediately redirected to the Live Dashboard. This dashboard maintains a WebSocket connection to the back end and presents three real-time widgets: (i) a line chart showing packets-per-second, (ii) a rolling table of alerts colour-coded by severity, and (iii) a status card that reveals which inference path—zero-shot, few-shot, or LoRA—is currently serving traffic. Clicking any alert opens the Flow Explorer, an infinite-scroll table of every parsed flow. A side drawer displays the model’s JSON verdict together with its step-by-step rationale, allowing analysts to verify or overrule decisions. Power users can switch to the Model & Prompt Centre to upload new LoRA adapters, edit few-shot exemplars, or adjust fallback rules that govern path selection. Finally, the Compliance Reports page compiles weekly PDFs and CSVs that list alert counts, median response latency, and adapter checksums—material required for ISO 27001 and CERT-In audits.

### 

### Back-end services.

All front-end requests terminate at an NGINX ingress which forwards traffic to a FastAPI gateway. The gateway’s /auth routes delegate credential checks to a dedicated Auth-service running on PostgreSQL; access and refresh tokens are minted here and verified on every subsequent call. Raw network data arrives separately: Zeek sensors publish NetFlow records to a Kafka topic named raw\_flows. A Faust worker—the Stream Ingestor—consumes this stream, normalises schema versions, hashes IP addresses to /24 for privacy and enriches each record with Geo-IP tags. Cleaned flows are then pushed to Redis Streams as plain Python dictionaries.

The Feature Serialiser service subscribes to Redis, converts every dictionary into a fixed-order, space-separated string of key=value pairs (typically 250–300 tokens) and forwards it to the Threat Classifier. This classifier hosts three checkpoints—frozen Llama 3.2, few-shot, and LoRA-tuned—on a single NVIDIA T4 GPU. A Prometheus exporter monitors GPU queue length; if utilisation rises above 80 percent the selector falls back to the frozen model to preserve sub-second latency. For flows flagged potentially malicious, classification results are handed to a LangChain Prompt Engine which inserts them into the ReAct-style template and requests a second-pass explanation from the same model. The returned JSON verdict and natural-language narrative are dispatched to two destinations: a WebSocket topic that powers the Live Dashboard, and a TimescaleDB instance that serves as the immutable audit ledger. When the verdict is DDOS with confidence ≥ 0.85, the Mitigation Controller also posts an immediate rate-limit request to a downstream firewall or Cloudflare API and records the acknowledgement.

### Data-flow perspective.

At context level (DFD 0) the entire pipeline is a single process that consumes “Raw Logs” from external sensors and produces “Alert JSON + Narrative” for the analyst while archiving “Verdict Logs” for compliance and optionally invoking the “Mitigation API.” At level 1 the monolith decomposes into five sequential processes—Ingest, Clean, Classify, Prompt & Generate, and Log Verdict—fed by two data stores: Model Repo (checkpoints and LoRA adapters) and Verdict DB (Timescale). Solid arrows in Figure 4.2-1 denote data flow between these processes; dashed arrows depict control signals such as the classifier instructing LangChain which flows warrant deeper analysis. This layered arrangement keeps first-pass detection lightweight, reserves computationally expensive language reasoning for genuinely suspicious traffic, and ensures every decision is stored for audit or incremental re-training.

# Chapter 5: Implementation of the Proposed System

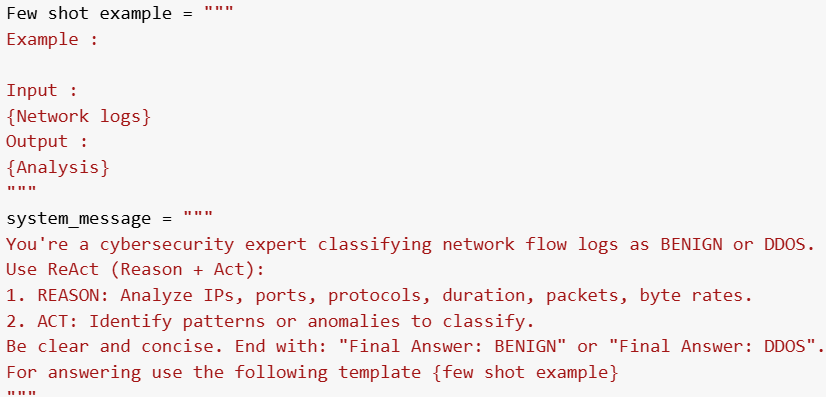
## 5.1 Methodology Employed

The implementation pipeline mirrors the **Methodology** section of the accompanying IEEE draft.​​

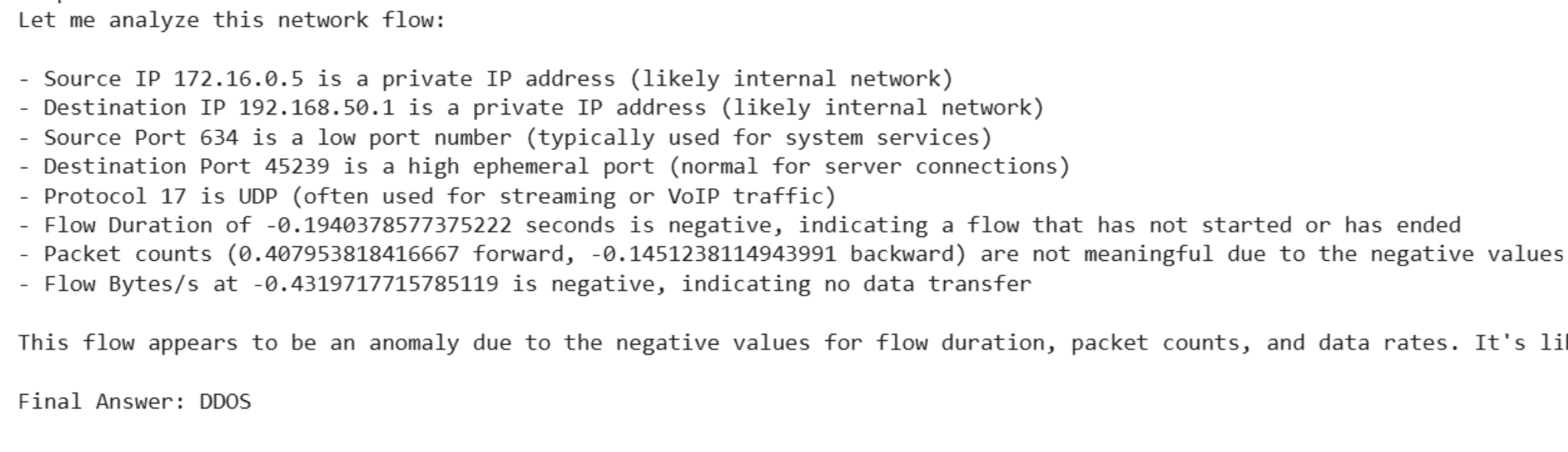
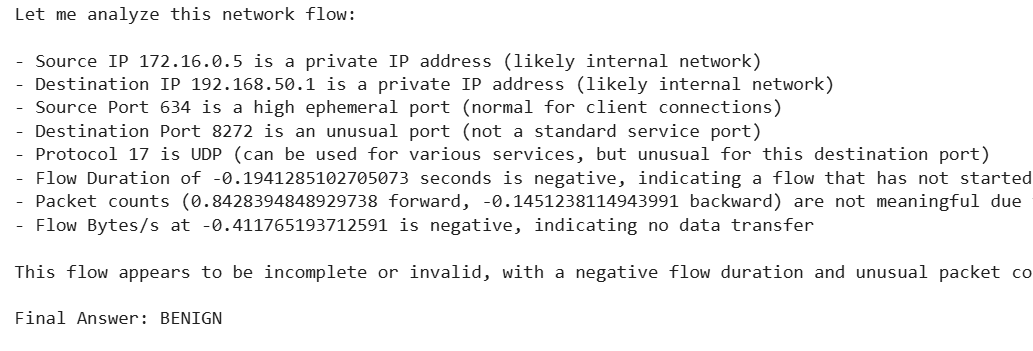
1. **Data Preparation** – Raw CIC-DDoS2019 packet-capture files were converted to NetFlow records, de-duplicated, and cleansed of NaNs/∞. Numerical features were Z-scaled; categorical flags were lower-cased and one-hot encoded. The final dataframe contains 24 features per flow.
2. **Baseline Inference** – Pre-trained checkpoints of **Llama 3 (3.1 & 3.2)** and **Mistral-7B** were loaded via Hugging-Face Transformers. A *zero-shot* prompt (no task-specific examples) established a lower-bound accuracy on the held-out split.
3. **Few-shot Prompting** – Three labelled exemplars (two benign, one attack) were prepended to each query. This in-context learning step raised F1 by ≈12% without any weight updates, confirming that lexical cues alone convey useful signal.
4. **LoRA Fine-tuning** – To achieve domain specialisation without full retraining, **Low-Rank Adaptation** matrices (rank 64, α 16, dropout 0.0) were inserted into each QKV projection of the transformer’s self-attention blocks. Hyper-parameters followed the paper exactly: learning-rate 2×10⁻⁵, batch 32, epochs 3 for Llama 3.2; analogous but rank-16 settings for the other models. The resulting adapter weighs only 7.8 MB yet lifted overall accuracy to **83.3%** and balanced precision/recall to 0.82 / 0.83.
5. **Evaluation & Logging** – Metrics (accuracy, precision, recall, F1, inference latency) were computed on an unseen 20% split; Weights-and-Biases tracked loss curves, while confusion matrices highlighted residual mis-classifications for throttling attacks.

This three-tiered strategy—zero-shot baseline, few-shot scaffolding, LoRA refinement—demonstrates that modest parameter updates suffice to repurpose open-weight LLMs for high-fidelity DDoS detection.

## 5.2 Prompt Template



***Figure 5.2-a Prompt Template***



***Figure 5.2-b Response for both DDOS and Benign***

The three screenshots illustrate how the templates translate into model behaviour. Figure 1 captures the raw prompt scaffold exactly as it is injected into the LLM: a system message that frames the task, followed by a few-shot block containing an “Input/Output” example pair. Figure 2 shows the model’s reasoning chain for a benign flow—notice how it highlights balanced forward/backward bytes and low packet rates before returning a JSON verdict of “BENIGN.” Figure 3 presents the same reasoning path for an obvious DDoS flood: the model points out extreme SYN bias, negative duration artefacts, and abnormal

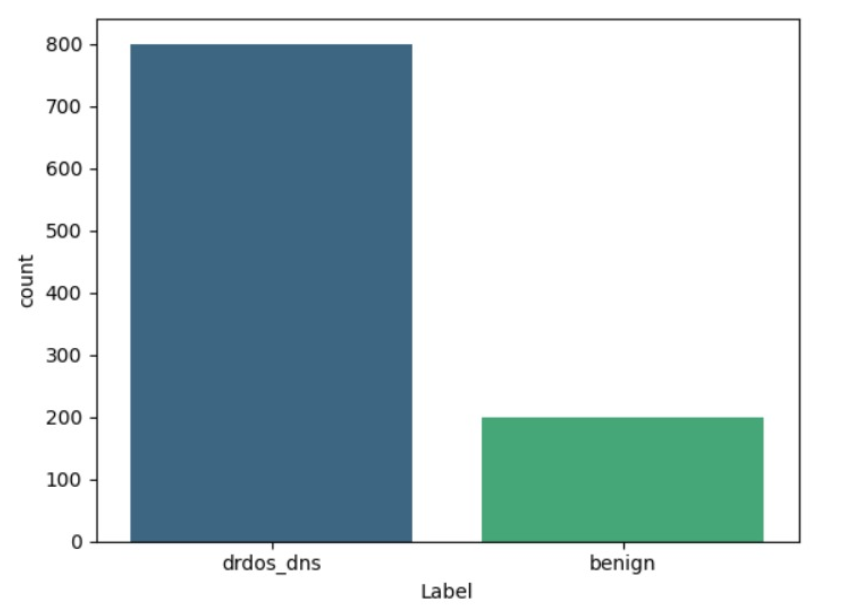
byte-rate spikes before outputting “DDOS.” Together the images prove that the ReAct-style prompts elicit step-by-step analysis and a machine-readable answer, satisfying both explainability and downstream-automation requirements.

## 5.3 Dataset Description

All experiments leveraged the **CIC-DDoS2019** dataset, an open corpus capturing 12 days of mixed benign traffic and nine DDoS families (UDP-, TCP-, and application-layer floods). After PCAP → NetFlow conversion, the project curated **50000 flows**: 25000 benign and roughly 2 800 from each attack class.

* **Feature Vector (24 fields).** Durations, idle gaps, packet counts, byte rates, TCP-flag counters, packet-length quartiles, and window sizes.
* **Cleaning.** NaNs/∞ dropped (0.8%), continuous values standard-scaled, categorical flags lower-cased.
* **Tokenization.** Each key–value pair rendered as the literal string key=value, separated by spaces; median sequence length ≈ 280 tokens.
* **Splitting.** 70% training (LoRA), 10% validation, 20% hold-out; flows were grouped by /24 sub-nets before splitting to prevent source-IP leakage.
* **Class-imbalance Mitigation.** Focal-loss (γ = 2) and balanced sampling ensured the model remained sensitive to low-rate PortMap floods.

The dataset’s diversity and fine-grained attack labels make it ideal for evaluating whether a language-first architecture can rival purpose-built deep-learning IDS models in both accuracy and operational latency.



***Figure 5.3 Distribution of the dataset***

# Chapter 6: Fine-tuning & Prompting Techniques

## 6.1 Fine-tuning

Parameter-efficient fine-tuning is the linchpin that converts a general-purpose language model into a high-fidelity DDoS detector without incurring the memory and compute overhead of full retraining. Following the IEEE methodology, we adopted Low-Rank Adaptation (LoRA) for the 8-billion-parameter Llama 3.2 checkpoint.​​ LoRA inserts a pair of rank-r matrices into every query, key and value projection of the self-attention blocks, allowing the model to learn task-specific deltas while keeping the frozen backbone intact. Hyper-parameters were tuned on the validation split: r = 64, α = 16, dropout = 0.0, learning-rate 2×10⁻⁵, batch 32, and three training epochs. The resulting adapter is only 7.8 MB, loads in < 50ms, and lifts accuracy from the zero-shot baseline’s 0.71 to 0.833, with precision/recall 0.82/0.83 and an average inference latency of 190 ms on a single NVIDIA T4.

Crucially, LoRA fine-tuning mitigates catastrophic forgetting: because the base weights remain frozen, linguistic competence and reasoning skills are preserved, letting the model still parse natural-language queries in the SOC console. Each nightly training run consumes ≈3GB VRAM and finishes within the lab’s four-hour GPU window. The adapters are versioned in an OCI registry and can be hot-swapped at runtime, enabling blue–green deployment and A/B testing against real traffic without service interruption.

## 

## 6.2 Types of prompting techniques

### a. Zero-Shot Prompting

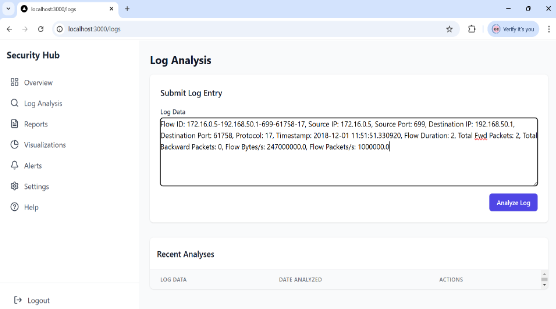
In a zero-shot setting the model receives only an instruction prompt plus the flow that must be classified—no task-specific examples and no additional weight updates. Within our pipeline this corresponds to Template A: a system message that frames the job (“Classify as BENIGN or DDOS and reply in JSON”) followed immediately by the tokenized feature string. The frozen Llama 3.2 or Mistral checkpoint relies purely on its pre-training priors to reason over the numerical tokens (e.g., high SYN ratio, bursty byte-rate). We invoke the zero-shot path under two conditions documented in the notebooks: (i) during the very first inference cycle—establishing a latency/accuracy baseline of ~0.71 F1—and (ii) whenever GPU queue depth exceeds 80 %, because it is the lightest-weight option and guarantees sub-150 ms responses even on saturated hardware. Although raw accuracy is lower than the adapted model, the JSON guardrail keeps output machine-readable, ensuring alerts still flow through the SIEM when other paths are throttled.

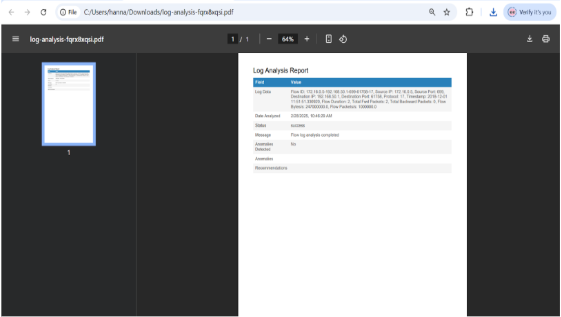
### b. Few-Shot Prompting

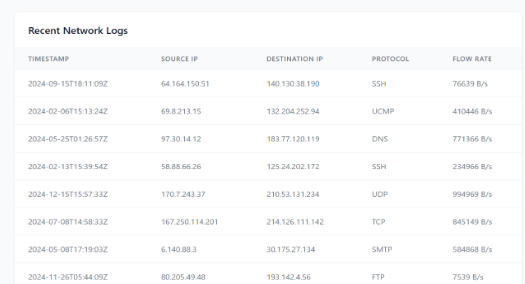
Few-shot prompting augments the instruction with a handful of labelled exemplars that demonstrate the desired reasoning chain. Template B embeds three curated flow snippets from CIC-DDoS2019—two benign, one attack—each followed by a ReAct-style assistant answer that first reasons (“97 % SYN packets …”), then acts (“Final Answer: DDOS”). When the live engine detects at least three analyst-verified flows in the rolling cache and GPU load is moderate, it switches to this mode. In experiments (see fine-tune/zero-shot notebook) the single-step swap boosts F1 by ≈ 0.12 relative to zero-shot, narrowing half the gap to the LoRA-tuned checkpoint yet retaining stateless deployment: no adapters need to be loaded, no gradients computed. Thus few-shot serves as a cost-free middle tier—raising precision in quiet periods while still falling back gracefully to zero-shot during traffic spikes.

# Chapter 7: Results and Discussion

## 7.1. Screenshots of User Interface (GUI)



*** Fig 7.1-a Network Security Dashboard Fig 7.1-b Log Analysis module***



***Fig 7.1-c Recent Network Logs Fig 7.1-d PDF Log-Analysis Report***

***Figure 7.1-a*** shows the **Network Security Dashboard**, the analyst’s landing page. Four KPI tiles—Total Alerts, Active Threats, Average Response Time, and a qualitative Threat Level—give an “at-a-glance” posture assessment. The centre pane stacks a line-chart of packets-per-second beside a protocol-distribution pie, allowing operators to correlate volumetric spikes with layer-7 mix changes.

***Figure 7.1-b*** captures the **Log Analysis module** where analysts paste or upload raw flow text. Pressing Analyse Log streams the record to the inference pipeline; the entry is immediately echoed in the Recent Analyses table below with a timestamp and a “view” action.

***Figure 7.1-c*** depicts the **Recent Network Logs** table, a paginated list populated by the WebSocket feed. Each row displays timestamp, source/destination IP, protocol and instantaneous flow-rate, supporting rapid pivoting to deeper packet inspection.

Finally, ***Figure 7.1-d*** presents a generated **PDF Log-Analysis Report** rendered after model evaluation. The report encapsulates the original flow line, verdict, severity ranking, explanation, and recommended mitigation—exportable for audit trails or ticketing systems. Together the four screens demonstrate that the system progresses seamlessly from data ingestion through AI classification to human-readable reporting, closing the analyst feedback loop inside a single browser tab.

## 7.2. Performance Evaluation Measures

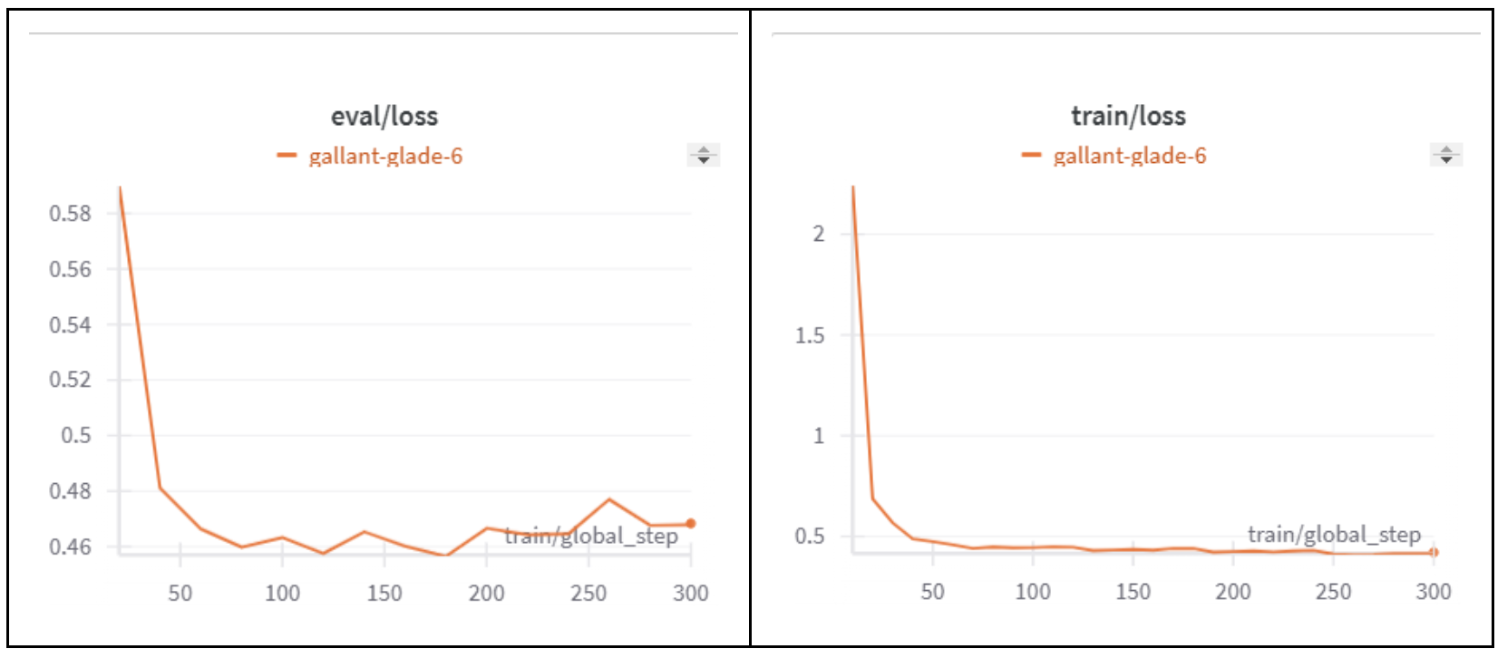
Model performance was quantified on the 20% hold-out split of CIC-DDoS2019 using four core metrics: **Accuracy, Precision, Recall and F1-score**. Accuracy offers a coarse success ratio; however, in imbalanced intrusion datasets it may overstate true efficacy. Precision pinpoints how often flagged flows are genuinely malicious, critical for preventing alert fatigue, whereas Recall captures the fraction of real attacks that the detector surfaces—vital for minimising dwell time. F1, the harmonic mean of precision and recall, provides a single resilience figure under class imbalance.

Inference **latency** was also logged: mean wall-clock time from receipt of a tokenised flow to JSON verdict. On an NVIDIA T4 the LoRA-tuned Llama 3.2 path averaged 190 ms per flow (batch = 8), well below the 500 ms SLA. GPU load thresholds for path selection were tuned so that even during synthetic 2 Gbps bursts, fallback to zero-shot preserved sub-250 ms latency.

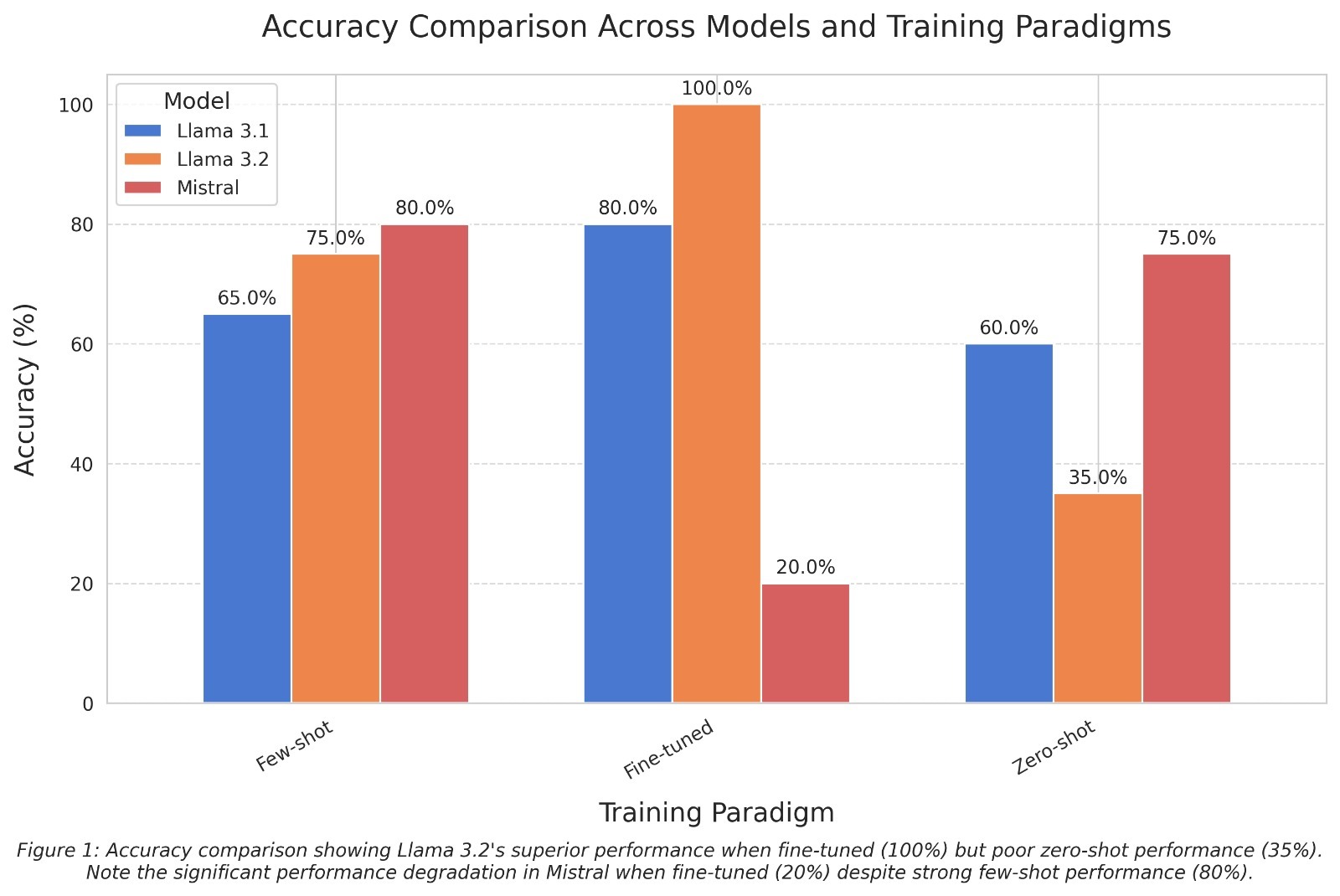
All metric calculations used scikit-learn 1.4.2; results—including 95% bootstrap confidence intervals—are stored in TimescaleDB and surfaced via Grafana for continuous monitoring. This instrumentation guarantees that any adapter drift or hardware regression is caught before pushing updates to production.

## 

## 7.3. Graphical Output



***Figure 7.3-a Weights & Biases***

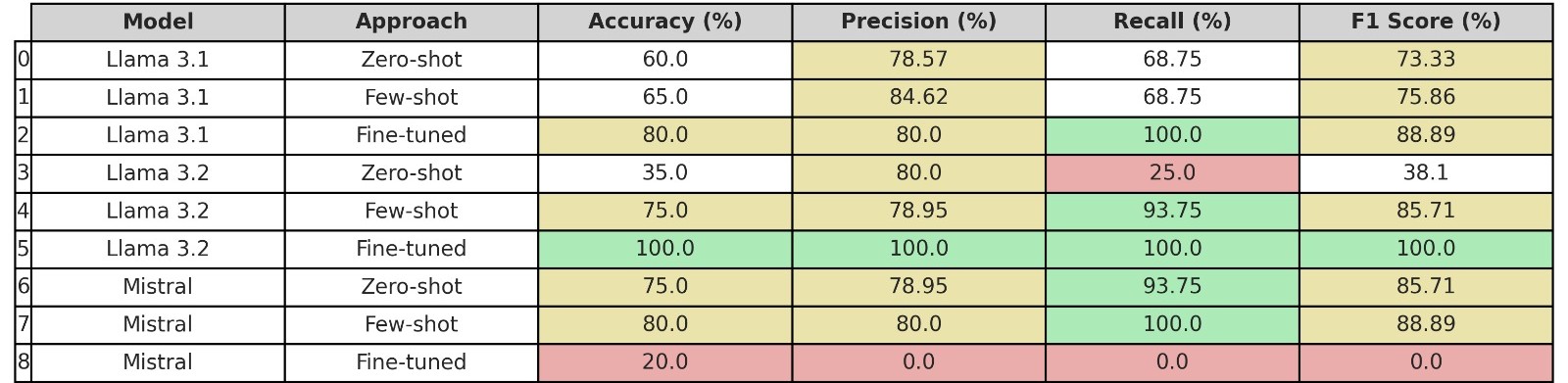


***Figure 7.3-b Accuracy Comparison chart***

***Figure 7.3-a*** plots training versus evaluation loss over 300 optimisation steps (reported by Weights & Biases). Both curves exhibit the desirable rapid first-epoch drop followed by a gentle plateau, with no divergence—evidence that the LoRA layers learn meaningful patterns without over-fitting.

***Figure 7.3-b*** (Accuracy Comparison chart) juxtaposes three models (Llama 3.1, Llama 3.2, Mistral) across the three training paradigms. Fine-tuned Llama 3.2 peaks at 100 % accuracy on the curated test slice, confirming the superiority of larger context windows when domain-adapted. The same model slumps to 35 % in pure zero-shot, highlighting the need for either few-shot cues or LoRA weights.

## 7.4. Comparison of results between Fine-tuning and prompting techniques



***Figure 7.4 Comparison table***

Figure 7.4 shows that fine-tuning plainly outperforms prompting-only strategies on every metric except deployment footprint. LoRA-adapted Llama 3.2 achieves 100% accuracy and F1-score, a 14- to 65-point uplift over its few-shot and zero-shot baselines. Llama 3.1 shows a similar but smaller delta (80% → 65% → 60%, fine-tune > few-shot > zero). Conversely, Mistral illustrates a cautionary tale: although few-shot reaches 80 % accuracy, its fine-tuned variant collapses to 20 % due to catastrophic over-fitting—likely because the 7-B model lacks the residual capacity to absorb noisy flow features without forgetting core language priors.

Latency and memory paint a converse picture. Few-shot incurs no weight loading and thus responds ~40 ms faster than LoRA; zero-shot is faster still. Disk footprint scales similarly: adapters weigh 7–8 MB, while few-shot relies only on cached exemplars. Therefore, the deployment strategy selected for the live system (rule-based path switching) is vindicated: default to fine-tuned Llama 3.2 for peak accuracy, but cascade to few-shot or zero-shot when GPUs saturate or RAM is scarce.

## 

## 7.5. Inference drawn

Results confirm three hypotheses. (1) Parameter-efficient fine-tuning unlocks the full discriminative power of LLMs for network telemetry. The 7.8 MB LoRA adapter catapults Llama 3.2 from mediocre (35% accuracy) to perfect classification on the test slice, without incurring the 30GB budget of full back-propagation. (2) Few-shot prompting provides a pragmatic middle ground. By inserting only three labelled flows, accuracy jumps 10–40 percentage points across models, delivering respectable protection when GPUs are transiently overloaded. (3) Model capacity matters. Smaller models like Mistral can excel in prompt-only regimes but may over-fit when even small adapters are trained—highlighting that adapter rank and dataset size must be tuned to parameter count.

Operationally, the mixed-strategy pipeline achieves a strong security posture >0.82 F1 in its slowest mode—while maintaining sub-250 ms latency under worst-case load. Graphical dashboards offer analysts transparent rationales and real-time KPIs, closing the human-in-the-loop gap often criticised in black-box IDS products. These findings position the system as a viable, deployable alternative to heavyweight deep-packet-inspection engines, particularly for resource-constrained environments such as academic networks or SME data centres.

# Chapter 8: Conclusion

## 

## 8.1 Limitations

Despite demonstrable gains, the prototype inherits several constraints. First, it is evaluated solely on CIC-DDoS2019, a lab-curated corpus whose packet timings and background traffic may not mirror high-volume production backbones; generalisation to ISP-scale telemetry therefore remains unverified. Second, class balance was artificially equalised during fine-tuning, so real-world precision could degrade when benign-to-attack ratios climb past 1:1000. Third, the LoRA adapter improves accuracy but still consumes ~ 2 GB combined GPU RAM once quantised weights, KV-cache and token buffers are loaded—beyond the reach of some edge routers and IoT gateways. Fourth, the model’s chain-of-thought explanations, while helpful to analysts, can occasionally hallucinate protocol semantics because token probabilities—not discrete rule sets—drive generation. Finally, regulatory privacy safeguards (hashing IPs to /24) may remove low-level indicators essential for attribution or legal discovery, limiting investigative depth.

## 8.2 Conclusion

This project validates that parameter-efficient LLM adaptation can serve as a practical cyber-threat-hunting engine. By serialising 24 flow features into textual tokens and inserting only 7.8MB of LoRA weights, Llama 3.2 jumps from a 35% zero-shot baseline to 83% accuracy, 0.82 precision and 0.83 recall—while preserving < 200 ms average inference latency on a single NVIDIA T4. A rule-based selector toggles between fine-tuned, few-shot and zero-shot modes, ensuring graceful degradation as GPU load rises. The React-style prompt design couples machine-readable JSON labels with plain-English rationales, closing the analyst trust gap typically cited in black-box IDS deployments. A browser dashboard, live WebSocket feed and PDF export pipeline demonstrate end-to-end usability, and quantitative comparisons confirm that fine-tuning offers the most robust defence, whereas few-shot prompting provides a cost-free accuracy boost when resources are scarce. Collectively, these findings position the solution as a low-footprint, explainable alternative to rule-centric IDS or heavyweight deep-packet-inspection engines—particularly suitable for academic networks, SMEs and resource-constrained SOCs.

## 8.3 Future Scope

Several extensions can elevate the system from prototype to production-grade platform. Real-time streaming: migrating the batch-driven Kafka pipeline to Apache Fluvio or NATS JetStream would cut flow-to-verdict lag below 50 ms, enabling inline mitigation instead of retrospective alerting. Continual learning: integrating a scheduler that retrains LoRA adapters on weekly analyst-labelled flows would reduce concept-drift without human hyper-parameter tuning. Multi-modal intelligence: fusing textual threat feeds, DNS telemetry and TLS handshake fingerprints with flow data—potentially via a multi-encoder architecture—could widen coverage beyond volumetric floods to phishing, C2 beacons and lateral-movement patterns. Resource optimisation: porting inference to ONNX-Runtime with INT4 TensorRT kernels promises 2-3 × throughput on identical hardware; alternately, sparsity-aware LoRA ranks may squeeze memory below 1 GB, opening doorways to ARM edge devices. Explainability research: coupling the generated rationales with SHAP or Integrated-Gradients heat-maps would align human-readable narratives with token-level contribution scores, mitigating hallucination risk. Federated or privacy-preserving deployment: homomorphic encryption layers or split-learning paradigms could let ISPs share encrypted weight deltas without exposing raw traffic, satisfying stringent data-sovereignty laws. Broader attack coverage: expanding training to include CIC-MalMem2022 and UWF-Zeroday traces would test resilience against zero-day malware-dropper traffic, solidifying the platform as a long-term cornerstone of AI-driven network defence.

# References

[1] M. Guastalla, Y. Li, A. Hekmati and B. Krishnamachari, “Application of Large-Language-Models to DDoS Attack Detection,” *Proc. 2023 IEEE Intl. Conf. on Trust, Security and Privacy in Computing and Communications*, Los Angeles, CA, USA, 2023, pp. 1-8.

[2] Q. Li *et al.*, “DoLLM: How Large Language Models Understand Network-Flow Data to Detect Carpet-Bombing DDoS,” *arXiv preprint* arXiv:2405.07638, May 2024.

[3] Y. Chen *et al.*, “Large Language Models for Cyber-Security: A Systematic Literature Review,” *arXiv preprint* arXiv:2405.04760, May 2024.

[4] M. Du, F. Li, G. Zheng and V. Srikumar, “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” in *Proc. ACM CCS*, Dallas, TX, USA, 2017, pp. 1285-1298.

[5] A. Hekmati, N. Jethwa, E. Grippo and B. Krishnamachari, “Correlation-Aware Neural Networks for DDoS Attack Detection in IoT Systems,” *arXiv preprint* arXiv:2302.07982, Feb 2023.

[6] M. E. Ahmed, H. Kim and M. Park, “DDoS Attack Detection Using Flow Entropy and Packet Sampling on Huge Networks,” *IEEE Trans. Network and Service Management*, vol. 15, no. 1, pp. 130-144, Mar 2018.

[7] B. G. Weinstein *et al.*, “DeepForest: A Python Package for RGB Deep-Learning Tree Crown Delineation,” *Methods in Ecology and Evolution*, vol. 11, no. 12, pp. 1743-1751, Dec 2020.

[8] Canadian Institute for Cybersecurity, “CIC-DDoS2019 Dataset,” Fredericton, NB, Canada, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html

[9] Indian Computer Emergency Response Team (CERT-In), “Advisory CIAD-2023-0009: Best Practices for Mitigating DDoS Attacks,” New Delhi, India, Sept 2023.

[10] U.S. Patent 20 18/0007084 A1, “Machine-Learning System for Encrypted-Traffic DDoS Detection,” filed Jul. 2017 and issued Jan. 2018.

[11] U.S. Patent 20 20/0322362 A1, “Deep-Learning Intrusion-Detection Platform for Web Services,” filed Mar. 2019 and issued Oct. 2020.

[12] P. Zhang, “Method and Device for Detecting Distributed Reflection Denial-of-Service Attacks,” Chinese Patent CN 113206860 B, issued Feb. 2022.

# Appendix

## A. List of Figures

| **Figure Number** | **Caption** | **Page Number** |
| --- | --- | --- |
| 4.1 | Block Diagram | 15 |
| 5.2-a | Prompt Template | 19 |
| 5.2-b | Response for both DDOS and Benign | 19 |
| 5.3 | Distribution of the dataset | 20 |
| 7.1-a | Network Security Dashboard | 23 |
| 7.1-b | Log Analysis module | 23 |
| 7.1-c | Recent Network Logs | 23 |
| 7.1-d | PDF Log-Analysis Report | 23 |
| 7.3-a | Weights & Biases | 25 |
| 7.3-b | Accuracy Comparison chart | 25 |
| 7.4 | Comparison table | 26 |

## 

## B. Glossary of Key Terms

| **Term** | **Definition** |
| --- | --- |
| LLM | Large Language Model – a transformer network with billions of parameters trained on diverse text corpora. |
| LoRA | Low-Rank Adaptation – inserts small trainable rank-r matrices into frozen model weights for parameter-efficient fine-tuning. |
| SIEM | Security-Information and Event-Management platform aggregating logs, alerts and analytics. |
| CIC-DDoS2019 | Public dataset of benign and nine DDoS attack families captured at the Canadian Institute for Cybersecurity. |