LLMs for Cyberattack Detection

Mobin Tirafkan mobin.ti@ut.ac.ir
Kiyan Khezri kiyankhezri@ut.ac.ir
Mohammad Mehrabian m.mehrabian@ut.ac.ir

1 Problem Statement

In the rapidly evolving landscape of cybersecurity, the sheer volume and velocity of network traffic logs present an overwhelming challenge for security analysts. Traditional rule-based intrusion detection systems often struggle with the sophistication of modern attacks, leading to high false positive rates and missed threats. While large language models (LLMs) have demonstrated impressive capabilities in various domains, their direct application to raw, tabular network log data for threat analysis faces significant hurdles:

1. **Semantic Gap for LLMs:** Raw network logs, characterized by cryptic numerical values and abbreviated categorical codes, lack the natural language context that LLMs are designed to process. This semantic mismatch leads to poor performance, making it difficult for general-purpose LLMs to extract meaningful insights or detect subtle anomalies without extensive feature engineering.

2. **Scalability and Resource Intensiveness:** Utilizing large, off-the-shelf LLMs for real-time network security is computationally expensive and resource-intensive, making them impractical for deployment in many operational environments.

3. **Lack of Explainability:** Most traditional deep learning models, and to some extent even LLMs when forced to classify raw data, operate as ”black boxes.” This inherent lack of explainability hinders security analysts from understanding *why* a particular log entry was flagged as malicious, impeding incident response and forensic analysis.

The core motivation of this project is to bridge these gaps by transforming raw, unintelligible network logs into a human-readable, context-rich ”story” that not only enables effective threat detection but also provides a clear, evidence-based ”reason” for the classification. Our goal is to develop a framework that moves beyond mere anomaly detection to deliver **explainable verdicts**, thereby enhancing the capabilities of security analysts, reducing alert fatigue, and facilitating more efficient incident response.

2 What You Proposed vs. What You Accomplished

Our project largely followed the proposed plan, with one significant modification to the scope. We successfully developed and evaluated a two-agent framework for generating explainable verdicts from network logs.

- Preprocess the UNSW-NB15 dataset and establish a statistical baseline for normal traffic.
- Implement a rule-based “Storyteller” agent to convert raw log entries into contextual, human-readable narratives.
- Curate a diverse and highly efficient training dataset (˜4,000 samples) by clustering and sampling the generated stories.
- Implement a “Reasoner” agent by fine-tuning a small language model (Gemma 3 1B) on the curated story-reason pairs.
- Rigorously evaluate the fine-tuned model against large, general-purpose LLMs using multiple performance metrics.
- Perform an in-depth error analysis to identify the model’s strengths and weaknesses across different attack categories.
- *Implement a third “Action Proposer” agent to suggest mitigation steps:* This agent was descoped due to fundamental challenges in both training and evaluation.

– **Training Challenge:** We could not effectively train a small model for this task. The poor performance of large commercial LLMs in this specific domain meant they could not serve as a reliable "Teacher" to generate the high-quality $reason \rightarrow action$ dataset required to fine-tune a smaller model.

– **Evaluation Challenge:** We lacked a viable method to evaluate the agent's output. Firstly, no standard benchmark dataset exists to quantitatively assess the correctness of proposed actions. Secondly, the poor domain performance of large LLMs made them unreliable for use in an "LLM as a Judge" capacity, leaving no viable path for automated evaluation.

Note on Project Scope: The primary modification from our initial proposal was the shift from a three-agent "From Logs to Action" framework to a two-agent "From Logs to Explainable Verdict" framework, focusing entirely on the novelty of the story-generation and fine-tuned reasoning components.

3 Related Work

Our framework is designed as a two-stage pipeline: (1) log-to-story generation and (2) story-to-verdict reasoning. Each stage builds upon concepts from prior work but is uniquely integrated and extended through our novel approach.

3.1 Log-to-Story Generation

Automatically transforming raw network logs into coherent, human-readable narratives is a largely underexplored area. Most prior studies focus on *log parsing*—extracting structured templates and variables—rather than generating fluent, analytical text. For instance, methods like DivLog use LLMs for template extraction (Xu et al., 2023), while others such as LogPrécis and LibreLog apply LLMs to cluster log sequences without producing natural-text outputs (Boffa et al., 2023; Ma et al., 2024). Some work explores summarization, like LogSummary, which extracts semantic triples (Meng et al., 2020), but this remains a structured summary rather than a fluid narrative.

Unlike these approaches, our "Storyteller" agent uniquely addresses this gap by employing a deterministic, rule-based Python algorithm. Its key innovation is the **contextualization** of log data. By comparing each feature of a log entry against a pre-computed statistical baseline of normal behavior derived from UNSW-NB15 (Moustafa and Slay, 2015) dataset, it generates a rich, analytical story. This narrative is specifically designed to be intelligible to a downstream LLM, highlighting anomalies in a qualitative manner (e.g., "several-fold higher than the Usual") that is far more descriptive than the template-based outputs of prior parsing or summarization techniques.

3.2 Threat Detection and Reasoning in Logs

Anomaly detection in logs is a well-studied field. Early methods like DeepLog used LSTMs to model log sequences and identify deviations (Du et al., 2017). More recent approaches have adapted Transformer-based models, such as LogBERT for embedding log sequences (Guo et al., 2021) and LogGPT for autoregressive anomaly detection (Han et al., 2023).

Recently, the focus has shifted towards applying LLMs directly to tabular cybersecurity data. For example, Moubayed et al. (2025) introduce TAD-GP, a framework that uses guided, chain-of-thought prompting on small LLMs to achieve efficient anomaly detection. Concurrently, work by Li et al. (2024) explores various fine-tuning optimization strategies for LLMs on tabular data, demonstrating the significant impact of data representation (e.g., decimal truncation) on model performance. While powerful, these methods still primarily target binary classification, often functioning as "black boxes" that lack deep explainability (Pellegrino et al., 2021).

Our "Reasoner" agent differs fundamentally from these methods. First, it operates not on raw logs but on the natural language "Story" provided by Agent 1. Second, its core is a novel **"Teacher-Student" fine-tuning strategy**, a form of knowledge distillation (Hinton et al., 2015). We use a powerful "Teacher" LLM to generate a high-quality dataset of $Story \rightarrow Reason$ pairs. This dataset is then used to fine-tune a small, efficient "Student" model (Gemma 3 1B). The key advantage over previous detection models is its output: instead of a simple binary label, our agent produces an **explainable verdict**, consisting of both a classification label and a concise, evidence-based reason, similar in goal to systems that generate natural language explanations for alerts (Bailey et al., 2021). This directly addresses the critical explainability gap, providing human-understandable rationale for detected threats. The inherent limitations of using an "LLM as a Judge" for tasks where models exhibit poor domain performance, as highlighted by Zheng et al. (2023), also underscored the importance of generating direct, evidence-based reasons rather than relying on automated action recommendations.

4 Our Dataset and Task

The core of our project is to transform raw, cryptic network logs into human-readable, explainable verdicts. This task requires a dataset that is both representative of real-world network traffic and contains a diverse set of labeled cyberattacks. For this, we utilize the **UNSW-NB15 dataset** (Moustafa and Slay, 2015), a widely-used benchmark in network intrusion detection research.

The dataset contains over 175,000 records in its training set, each representing a single network flow with 49 features. The primary challenge this data presents is a **semantic gap**: the features are numerical or abbreviated categorical codes (e.g., 'sbytes: 200', 'proto: scps'), which are not inherently intelligible to Large Language Models (LLMs) that thrive on natural language. Furthermore, the distinction between a benign anomaly and a malicious pattern can be incredibly subtle, requiring deep contextual understanding that simple classification models often lack.

Our task is to build a pipeline that first enriches a raw log entry with context (Preprocessing), and then uses that enriched context to generate a final, evidence-based verdict (Annotation for training). An example of this end-to-end transformation is shown in Table 1.

4.1 Data Preprocessing (Agent 1: The Storyteller)

Our preprocessing is a unique **contextualization** step designed to bridge the aforementioned semantic gap. Instead of traditional numerical normalization, our rule-based "Storyteller" agent transforms the data.

First, we performed a feature selection step, choosing **21 key features** from the original 49 based on a statistical analysis of their correlation with the 'attack_cat' label. We then created a statistical baseline by analyzing only the "Normal" records in the training set to compute the mean for numerical features and frequency distributions for categorical ones.

The Storyteller algorithm then processes each log record by comparing its 21 feature values against this baseline. This allows it to generate a rich, qualitative narrative (the "Story") that highlights deviations from normal behavior. This preprocessing step is crucial as it converts the structured, unintelligible log data into a format that is both human-readable and semantically rich for our downstream language model.

4.2 Data Annotation (Agent 2: Training Data Generation)

Our project's "annotation" phase was the creation of a high-quality dataset to fine-tune our Reasoner agent. We followed a "Teacher-Student" methodology.

First, we curated an efficient and diverse set of stories. Using sentence embeddings, we clustered all stories from the training set and sampled approximately 4,000 representative examples. This final set constituted just **3%** of the total available training data.

The core annotation was performed by our "Teacher," a powerful **GPT-5** model. For each of the ˜4,000 curated stories, we provided the Teacher with both the story and its ground-truth label. The model's task was to generate a concise, evidence-based "Reason" explaining why the label was correct. This resulted in a final fine-tuning dataset of high-quality $Story \rightarrow Reason$ pairs, which forms the basis for training our specialized Reasoner agent.

5 Baselines

Given that the primary goal of our project is to produce *explainable verdicts* rather than solely maximizing classification accuracy, we chose not to use traditional machine learning models (e.g., SVM, Random Forest) as our primary baselines. Such models, while often accurate, function as "black boxes" and do not align with our core objective of generating human-readable reasons. Instead, our baselines are designed to measure the effectiveness of our data representation (the "Story") and to establish a performance benchmark using Large Language Models in a pre-fine-tuning context.

We use different configurations of the **Gemma-3 4B IT** model with In-Context Learning (ICL) as our main baselines. This allows us to directly quantify the performance lift provided by our "Storyteller" agent.

5.1 Train and Test Split

Our final, fine-tuned model ('Gemma 3 1B') was trained on our curated dataset of approximately

Input (Sample Raw Log Features from UNSW-NB15)
`proto: scps, service: -, state: INT, dur: 0.000009, sbytes: 200, dbytes: 0, sload: 88888888.0, sttl: 254, attack_cat: Exploits, ...`
Intermediate Output (Generated "Story" from Agent 1)
This record describes a single network flow using scps (protocol; not available in the Usual set); no specific application service was identified (observed in ˜74% of the Usual set), and the connection state recorded as INT (interrupted or incomplete transaction)... source→destination data volume was ˜95% below the Usual... source→destination transfer rate was ˜+124% vs the Usual... source→destination hop limit (TTL) was ˜+104% vs the Usual...
Final Output (Explainable Verdict from Agent 2)
`{"reason": "Unusual protocol with no service identified, abnormally low data volume and packet counts, and TCP handshake timing anomalies suggest a spoofed or incomplete transaction.", "label": "attack"}`

Table 1: An example of a single raw log entry's transformation through our two-agent pipeline, from cryptic features to a final, explainable verdict.

4,000 story-reason pairs. The model's performance was evaluated on a completely separate, held-out **test set of 1,000 examples** from the same curated data pool. All reported metrics are on this test set. No hyperparameter tuning was performed on the test set.

5.2 In-Context Learning (ICL) Baselines

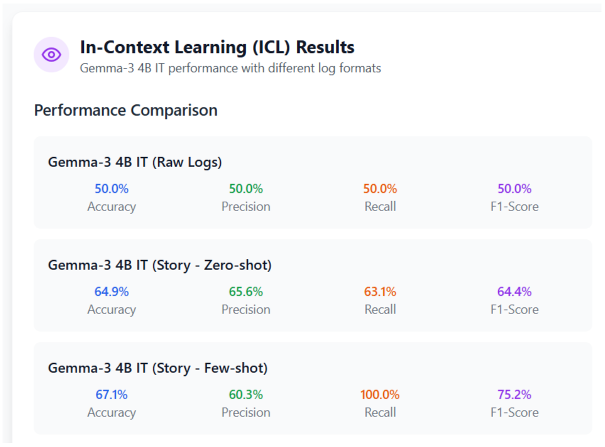
ICL allows us to evaluate an LLM's performance on a task by providing instructions and a few examples directly in the prompt, without updating the model's weights.

- **Baseline 1: Raw Log Prompting.** In this baseline, the LLM was prompted directly with the raw, tabular log features. The model consistently failed to outperform random chance, achieving an **accuracy of 50.0%**. This confirms the semantic gap and the difficulty LLMs face in interpreting non-linguistic, tabular data.

- **Baseline 2: Zero-Shot Story Prompting.** Here, the LLM was prompted with the contextual "Story" generated by our Agent 1, without any examples. The performance improved dramatically to an **F1-Score of 64.4%**, demonstrating that the Story format successfully makes the data intelligible to the LLM.

- **Baseline 3: Few-Shot Story Prompting.** This baseline provided the LLM with the input "Story" along with a few examples of other stories and their correct verdicts in the prompt. This further improved performance, achieving an **F1-Score of 75.2%**. This represents the strongest performance achievable without fine-tuning.

The results of these ICL baselines are summarized in Figure 1. While the "Story" format clearly provides a significant advantage, the performance is still insufficient for a reliable security application, motivating our fine-tuning approach.



In-Context Learning (ICL) Results
Gemma-3 4B IT performance with different log formats

Performance Comparison

	Accuracy	Precision	Recall	F1-Score
Gemma-3 4B IT (Raw Logs)	50.0%	50.0%	50.0%	50.0%
Gemma-3 4B IT (Story - Zero-shot)	64.9%	65.6%	63.1%	64.4%
Gemma-3 4B IT (Story - Few-shot)	67.1%	60.3%	100.0%	75.2%

Figure 1: Performance comparison of In-Context Learning (ICL) with different input formats on the Gemma-3 4B IT model. The "Story" format provides a significant performance boost over raw logs.

5.3 Fine-Tuning Hyperparameters

The following hyperparameters were used for fine-tuning our final 'Gemma 3 1B' model:

- Learning Rate: 5e-4
- Number of Epochs: 7
- Batch Size: 4

6 Our Approach

Our approach is a novel two-agent framework designed to transform raw, cryptic network logs into explainable verdicts. This system, titled "From

Logs to Explainable Verdict," directly addresses the shortcomings of the baselines by first bridging the semantic gap between tabular data and language models, and then training a specialized agent to provide human-readable reasoning. We successfully completed a working implementation of this two-stage pipeline.

6.1 Agent 1: The Rule-Based Storyteller

The first agent, the "Storyteller," functions as an intelligent, deterministic preprocessor. Its purpose is to convert a single row of numerical and categorical log data into a rich, contextual narrative that is intelligible to a language model. This process is not handled by an LLM but by a rule-based algorithm implemented in Python, primarily using the `pandas` library.

The algorithm's logic is twofold. First, an offline analysis is performed on the "Normal" traffic from the UNSW-NB15 training set to build a statistical baseline (e.g., mean values for numerical features, frequency distributions for categorical ones). Second, for each new log record, the agent compares its 21 selected features to this baseline to generate descriptive phrases.

Numerical Feature Processing For features like 'sbytes' or 'sload', the algorithm compares the value to the stored mean, generating a qualitative phrase that describes the deviation. A conceptual snippet of this logic is shown below:

```python
# Conceptual Python Snippet for Numerical Features
def get_numerical_phrase(value, mean):
    if mean <= 0: return "value is present while none is usual"
    ratio = value / mean
    if ratio > 10:
        return f"~{ratio:.0f}x higher than the Usual"
    else:
        diff_pct = (ratio - 1) * 100
        return f"~{diff_pct:+.0f}% vs the Usual"
```

Categorical Feature Processing For features like 'proto' or 'service', the algorithm looks up the value's frequency in the pre-computed distribution map to describe its rarity.

```python
# Conceptual Python Snippet for Categorical Features
def get_categorical_phrase(value, freq_dist):
    percent = freq_dist.get(value, 0) * 100
    if percent < 1:
        return f"a rare value ({value})"
    else:
        return f"a common value ({value})"
```

The final "Story" is a concatenation of these phrases, providing a rich input for the next stage of the pipeline.

6.2 Agent 2: The LLM-Based Reasoner

The second agent, the "Reasoner," is a specialized small language model whose task is to analyze the narrative from the Storyteller and produce a final, explainable verdict.

Model and Fine-Tuning. We implemented this agent using the **Gemma 3 1B** model. The model was fine-tuned on a curated dataset of ∼4 000 `Story→Reason` pairs generated by our "Teacher" (`GPT-5`) model. The fine-tuning process was implemented in Python using `PyTorch`, `Hugging Face Transformers`, and `TRL`. A key library in our implementation was **`Unsloth`**, which provides highly optimized kernels that dramatically reduce GPU memory usage and increase training speed.

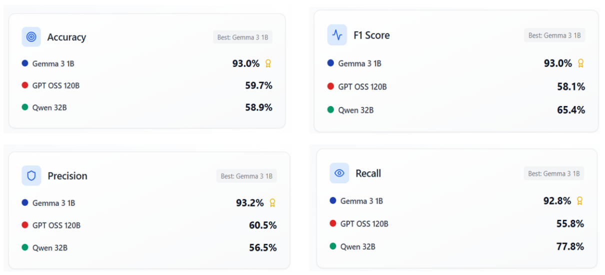
Computational Setup and Challenges Initial experiments on Google Colab faced significant Out-of-Memory (OOM) errors, even with small batch sizes. The use of `Unsloth` was critical in overcoming these limitations, making it feasible to fine-tune the model on a consumer-grade workstation equipped with an **NVIDIA RTX 3090 GPU**. The hyperparameters used for the final successful training run were:

- Learning Rate: 5e-4
- Number of Epochs: 7
- Batch Size: 4

6.3 Results and Comparison to Baselines

Our fine-tuned 'Gemma 3 1B' model achieved outstanding results, demonstrating the success of our two-agent approach. The model reached a final **balanced accuracy of 93.0%** and an **F1-Score of 93.0%** on the held-out test set of 1,000 examples.

As shown in Figure 2, this performance is a dramatic improvement over our ICL baselines, far surpassing the best ICL F1-Score of 75.2% (achieved with few-shot story prompting) and the random-chance performance on raw logs.



Accuracy	Best: Gemma 3 1B
● Gemma 3 1B	93.0%
● GPT OSS 120B	59.7%
● Qwen 32B	58.9%

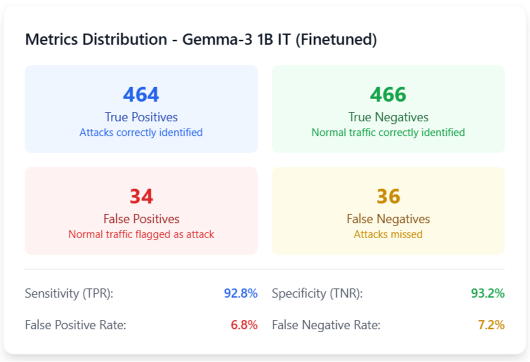
F1 Score	Best: Gemma 3 1B
● Gemma 3 1B	93.0%
● GPT OSS 120B	58.1%
● Qwen 32B	65.4%

Precision	Best: Gemma 3 1B
● Gemma 3 1B	93.2%
● GPT OSS 120B	60.5%
● Qwen 32B	56.5%

Recall	Best: Gemma 3 1B
● Gemma 3 1B	92.8%
● GPT OSS 120B	55.8%
● Qwen 32B	77.8%

Figure 2: Overall Performance Metrics of our fine-tuned Gemma 3 1B model compared to large LLM baselines (on a different task setup). Our model shows consistently high and balanced performance across all key metrics.

The strength of our model lies in its balance. As shown in Figure 3, it produces very few False Positives (34) and False Negatives (36), resulting in high Precision (93.2%) and Recall (92.8%). This reliability is critical for an operational security tool and stands in stark contrast to the unbalanced and poor performance of the ICL baselines.



Metrics Distribution - Gemma-3 1B IT (Finetuned)

464 True Positives — Attacks correctly identified	**466** True Negatives — Normal traffic correctly identified
34 False Positives — Normal traffic flagged as attack	**36** False Negatives — Attacks missed
Sensitivity (TPR): 92.8%	Specificity (TNR): 93.2%
False Positive Rate: 6.8%	False Negative Rate: 7.2%

Figure 3: Metrics distribution for our fine-tuned Gemma 3 1B model, showing a strong balance between True Positives/Negatives and low rates of False Positives/Negatives.

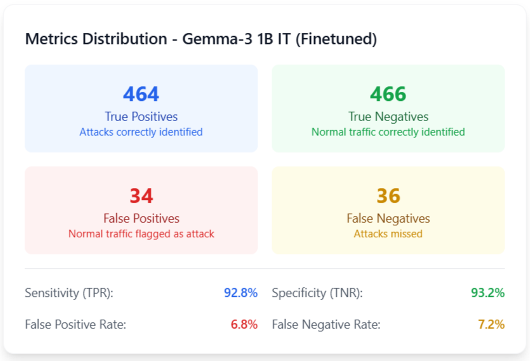
An in-depth error analysis on a per-attack-category basis (Figure 6) reveals that our model excels at detecting critical attack types like `Backdoor`, `DoS`, and `Reconnaissance` with near-perfect accuracy. An issue we could not fully solve is the model's weaker performance on less frequent and more complex categories like `Shellcode`, which remains an area for future work.

6.4 Results and Comparison to Baselines

Our fine-tuned 'Gemma 3 1B' model achieved outstanding results, demonstrating the success of our two-agent approach. The model reached a final **balanced accuracy of 93.0%** and an **F1-Score of 93.0%** on the held-out test set of 1,000 examples.

As shown in Figure **??**, this performance is a dramatic improvement over our ICL baselines, far surpassing the best ICL F1-Score of 75.2% (achieved with few-shot story prompting) and the random-chance performance on raw logs.

The strength of our model lies in its balance. As shown in Figure 5, it produces very few False Positives (34) and False Negatives (36), resulting in high Precision (93.2%) and Recall (92.8%). This reliability is critical for an operational security tool and stands in stark contrast to the unbalanced and poor performance of the ICL baselines.



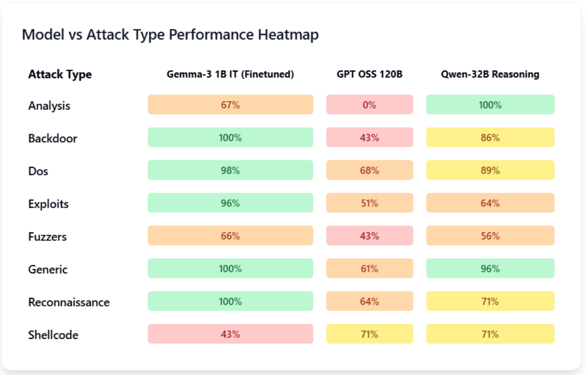
Metrics Distribution - Gemma-3 1B IT (Finetuned)

464 True Positives — Attacks correctly identified	**466** True Negatives — Normal traffic correctly identified
34 False Positives — Normal traffic flagged as attack	**36** False Negatives — Attacks missed
Sensitivity (TPR): 92.8%	Specificity (TNR): 93.2%
False Positive Rate: 6.8%	False Negative Rate: 7.2%

Figure 5: Metrics distribution for our fine-tuned Gemma 3 1B model, showing a strong balance between True Positives/Negatives and low rates of False Positives/Negatives.

6.4.1 Manual Error Analysis: Model vs. Attack Type Performance

To understand the specific strengths and weaknesses of our approach, we conducted a manual error analysis on our test set of 1,000 examples, focusing on performance across different attack categories. Figure 6 provides a detailed heatmap comparing our

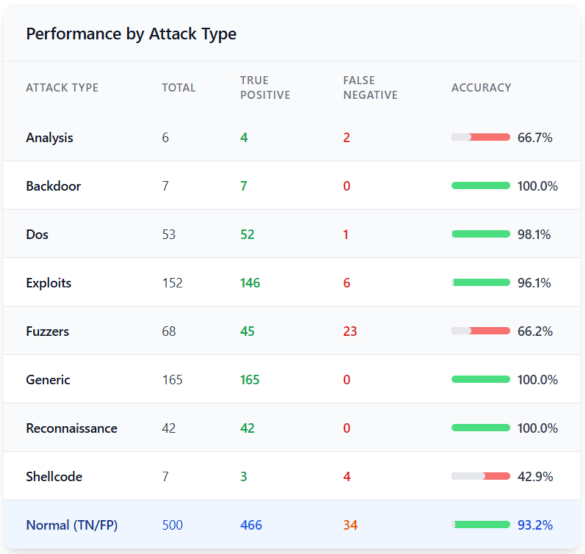


Model vs Attack Type Performance Heatmap

Attack Type	Gemma-3 1B IT (Finetuned)	GPT OSS 120B	Qwen-32B Reasoning
Analysis	67%	0%	100%
Backdoor	100%	43%	86%
Dos	98%	68%	89%
Exploits	96%	51%	64%
Fuzzers	66%	43%	56%
Generic	100%	61%	96%
Reconnaissance	100%	64%	71%
Shellcode	43%	71%	71%

Figure 4: Per-attack-type performance heatmap. Our model (Gemma 3 1B) shows consistently high performance on the most prevalent attack types.

‘Gemma 3 1B (Finetuned)‘ model’s performance against two other large LLM baselines (GPT OSS 120B and Qwen-32B Reasoning) across various attack types.



Performance by Attack Type

ATTACK TYPE	TOTAL	TRUE POSITIVE	FALSE NEGATIVE	ACCURACY
Analysis	6	4	2	66.7%
Backdoor	7	7	0	100.0%
Dos	53	52	1	98.1%
Exploits	152	146	6	96.1%
Fuzzers	68	45	23	66.2%
Generic	165	165	0	100.0%
Reconnaissance	42	42	0	100.0%
Shellcode	7	3	4	42.9%
Normal (TN/FP)	500	466	34	93.2%

Figure 6: Model vs. Attack Type Performance Heatmap, comparing our fine-tuned Gemma 3 1B with two large LLM baselines across different attack categories.

Failures of LLM Baselines (GPT OSS 120B, Qwen-32B Reasoning) The heatmap clearly illustrates that the larger, general-purpose LLMs struggle significantly more across almost all attack types compared to our fine-tuned model.

- **GPT OSS 120B** shows particularly poor performance, even failing completely (0%) on ‘Analysis‘ attacks and exhibiting low accuracy across the board (e.g., 43% on ‘Backdoor‘, 51% on ‘Exploits‘). This suggests that without specific fine-tuning or highly detailed few-shot examples, very large LLMs have difficulty capturing the subtle nuances of network anomaly detection from the ”Story” format. Their broad knowledge isn’t specialized enough for this domain.

- **Qwen-32B Reasoning** performs better than GPT OSS 120B but still lags behind our ‘Gemma 3 1B‘ in most categories. Its highest performance (100% on ‘Analysis‘) might be due to specific patterns it generalizes well, but its lower scores on other types (e.g., 64% on ‘Exploits‘, 71% on ‘Reconnaissance‘ and ‘Shellcode‘) indicate a lack of robust, consistent understanding of all attack characteristics.

Common to the failures of these baselines is a likely inability to discern subtle statistical deviations and their security implications, as expressed in the ‘Story‘ format, without the specialized knowledge imparted through fine-tuning. Their general-purpose nature means they are less sensitive to domain-specific jargon and contextual cues that our fine-tuned model has learned.

Failures of Our Approach (Gemma 3 1B Fine-tuned) While our fine-tuned ‘Gemma 3 1B‘ achieves significantly higher overall performance, the heatmap highlights specific areas of challenge:

- **Shellcode (43%):** This is the most challenging category for our model. Shellcode attacks are often characterized by small, highly evasive packets designed to exploit vulnerabilities. The difficulty likely stems from: * **Subtlety:** Shellcode might not always trigger large, obvious deviations in traffic volume or rate, making its ”Story” less distinct compared to other attacks. * **Rarity/Complexity:** These attacks might be less frequent in the training data, leading to fewer examples for the model to learn from. The semantic patterns in their stories might also be inherently more ambiguous or diverse.

- **Analysis (67%) and Fuzzers (66%):** These categories also show relatively lower performance compared to the others. * ‘Analysis‘ attacks often involve reconnaissance or information gathering, which can sometimes blend with benign exploratory traffic. * ‘Fuzzers‘ generate a high volume of malformed inputs, which might create ”Stories” that share characteristics with other high-volume attacks (e.g., DoS) or even some forms of normal, but noisy, network behavior. This semantic overlap can confuse the model.

The commonality between these difficult examples (‘Shellcode‘, ‘Analysis‘, ‘Fuzzers‘) is their potential for semantic ambiguity or overlap with other categories in their generated ”Stories.” Unlike clear-cut attacks like ‘DoS‘ (which typically involves massive volume deviations) or ‘Backdoor‘ (which often has persistent, unusual communication patterns), these attack types might present more varied and less uniquely identifiable ”Story” patterns, leading to occasional misclassifications. This suggests that while our ”Story” format is excellent, refining the contextualization for these particular subtle attack vectors, or augmenting training data for them, could further improve performance.

7 Contributions of Group Members

All members of the group contributed equally and collaboratively across all aspects of this project, from data collection and preprocessing to model building, training, error analysis, and report writing.

8 Conclusion

Our journey through developing the “From Logs to Explainable Verdict” framework provided deep insights into the practical application of NLP techniques for complex, structured data. It underscored both the power of language models when properly applied and the inherent challenges of real-world data transformation.

8.1 Key Takeaways and Learnings

- **Bridging the Semantic Gap is Paramount:** The most significant takeaway is the critical role of the “Storyteller” agent. Directly feeding raw, tabular network logs to LLMs yielded random results. The transformation into a human-readable ”Story” was not merely a preprocessing step; it was the fundamental enabler, making the task semantically tractable for language models. This highlights that for NLP on non-linguistic data, intelligent contextualization is far more valuable than raw data volume.

- **Small, Specialized LLMs Outperform Large Generalists (with Fine-Tuning):** Our results emphatically demonstrated that a fine-tuned ‘Gemma 3 1B‘ model significantly outperformed much larger, general-purpose LLMs (like GPT OSS 120B and Qwen-32B Reasoning) when performing our specific task. This reinforces the idea that for domain-specific tasks, focused fine-tuning of smaller, efficient models is often superior to relying on the broad, but unspecialized, knowledge of massive models, especially when considering computational costs and deployment.

- **Annotation is a Bottleneck, but Teacher Models Can Accelerate It:** The need for high-quality, explainable ”Reasons” meant traditional manual annotation would be prohibitively expensive. The “Teacher-Student” approach, leveraging a powerful LLM (GPT-5) for initial annotation, proved highly effective in creating a robust and consistent dataset for fine-tuning our student model.

8.2 Surprising Difficulties and Results

- **Unexpected Computational Hurdles:** A surprising difficulty was the severe GPU memory constraints encountered, even with relatively small models and batch sizes, particularly in cloud environments like Google Colab. The reliance on highly optimized libraries like ‘Unsloth‘ to overcome these Out-of-Memory (OOM) issues was a significant, unplanned, but ultimately successful, part of our implementation.

- **ICL’s Limitations vs. Fine-Tuning’s Power:** While we anticipated that fine-tuning would yield better results than In-Context Learning (ICL), the sheer magnitude of the performance gap was noteworthy. Even with few-shot examples and well-crafted stories, ICL could only reach 75% F1-score, whereas fine-tuning propelled the model to 93%. This highlights that for high-stakes, nuanced classification tasks, weight updates are indispensable.

- **Variability in Attack Type Detection:** The significant drop in performance for specific attack categories, notably “Shellcode,” was a point of interest. Despite the general high accuracy, these ”long-tail” or semantically complex attack types present unique challenges that are not easily captured by general patterns, even after fine-tuning.

8.3 Future Directions

If we were to continue working on this project, the following directions would be highly promising:

- **Targeted Data Augmentation for Challenging Attack Types:** Focusing on augmenting the training data specifically for categories like “Shellcode,” “Analysis,” and “Fuzzers” could significantly boost their detection rates. This could involve generating synthetic ”Stories” that highlight the unique, subtle characteristics of these attacks.

- **Real-time Deployment and Stream Processing:** Adapting the framework to ingest and process network logs in a streaming fashion would be the next logical step towards a production-ready system. This involves optimizing the Storyteller for low latency and integrating the Reasoner into a real-time inference pipeline.

- **Third Agent for Action Recommendation:** As LLM capabilities continue to advance, exploring the feasibility of a third agent that suggests concrete mitigation or response actions based on the "Reason" generated by Agent 2 would add another layer of value, moving from mere detection to actionable intelligence.
- **Multi-modal Contextualization:** Investigating whether incorporating other forms of context (e.g., threat intelligence feeds, user behavior analytics) into the "Story" could further enhance the Reasoner's accuracy and depth of explanation.

References

Bailey, J., Al-Shaer, E., and G'omez, L. (2021). Generating natural language explanations for intrusion alerts. In *Proceedings of the 14th International Conference on Natural Language Generation*.

Boffa, A., Ponzanelli, L., and Lanza, M. (2023). Logprécis: automatable log summarization for software systems. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of a Software Engineering*.

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*.

Guo, H., Yuan, S., and Wu, X. (2021). Logbert: A transformer-based model for log anomaly detection. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Han, W., Zhang, Y., Chen, Z., and Liu, D. (2023). Loggpt: Log anomaly detection via gpt. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Li, Z., Liu, J., Wang, H., Wang, Y., Zhang, Y., and Liu, J. (2024). Research on fine-tuning optimization strategies for large language models in tabular data processing. *Future Internet*, 9(11):708.

Ma, W., Zhang, H., Wang, J., and Li, Y. (2024). Librelog: A library for log analysis with log-centric foundation models. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE.

Meng, W., Liu, Y., Li, Y., Zhang, S., and Yin, H. (2020). Logsummary: a log summarization tool for cyber security applications. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*.

Moubayed, A., Injadat, M., and Mago, V. (2025). Efficient anomaly detection in tabular cybersecurity data using large language models. *Scientific Reports*, 15(1):88050.

Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*. IEEE.

Pellegrino, G., Di Sorbo, A., Di Notte, S., Visaggio, C. A., and Canfora, G. (2021). On the explainability of machine learning in cybersecurity. *Journal of Information Security and Applications*, 62:102968.

Xu, Z., Liu, Z., Chen, Z., Zhang, H., Chen, Z., Wang, Y., and Pu, G. (2023). Divlog: Log parsing with in-context learning. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Brooks, D., Xing, E., et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.