Internship Report
Training and Application of AI in the Field of Cybersecurity

Omar Abdesslem
Université de Genéve
Swiss AI Experts Labs

September 2025

Contents

1 Abstract

With the growing sophistication of cyberattacks, organizations require accessible and cost-effective solutions for threat detection and log analysis. This project introduces a customized LogLLM, specifically adapted to preprocess and classify raw Security Operations Center (SoC) logs into a structured and digestible format. Beyond anomaly detection and attack prediction, the system incorporates a final interpretation layer that translates model outputs into actionable insights. This layer is deployed through a Langchain interface, ensuring usability and accessibility for security analysts. The workflow includes log preprocessing, normalization, visualization, and correlation analysis to identify key features, followed by the application of the modified LogLLM for classification and interpretation. Validation is performed through simulated attack scenarios, enabling continuous evaluation and iterative refinement, based on Romain Christen's work [2]. By combining automated log intelligence with an interactive interface, this work delivers an affordable and scalable solution for both small and large organizations, including public institutions, to address fundamental cybersecurity needs.

Acknowledgements

I would first like to express my deep gratitude to Dr. Lionel Beaugé (SeLabs.ai) for his attentive guidance, availability, and the quality of his advice throughout this project. His expertise and support were essential to the progress of my work and to my learning during this summer internship. I would also like to thank Mr. Dominique Vidal, CEO of SecuLabs SA and SeLabs.ai, for giving me the opportunity to join his company and to take part in an enriching project. I extend my appreciation as well to Mr. Romain Christien, whose master's thesis provided a valuable and inspiring foundation for this internship.

Abréviations

AD	Anomaly Detection
CIC	Canadian Institute for Cybersecurity
CNN	Convolutional Neural Network
CSE	Communications Security Establishment
DDoS	Distributed Denial of Service
DL	Deep Learning
DoS	Denial of Service
GAN	Generative Adversarial Network
GNN	Graph Neural Network
GPO	Group Policy Object
GRU	Gated Recurrent Unit
AI	Intelligence Artificielle
ID	Intrusion Detection
IDS	Intrusion Detection System
IOC	Indicator Of Compromise
IP	Internet Protocol
LSTM	Long Short-Term Memory
LLM	Large Language Model
LOF	Local Outlier Factor
ML	Machine Learning
PCAP	Packet CAPture
SMEs	Small and medium-sized enterprises
RNN	Recurrent Neural Network
SSH	Secure SHell
SOC	Security Operation Center
SQL	Structured Query Language
SSL	Secure Sockets Layer

2 Introduction

Small and medium-sized enterprises (SMEs) often face significant challenges in managing cybersecurity threats due to limited resources and expertise. While larger organizations deploy complex and expensive Security Operations Centers (SoCs), SMEs require lighter, smarter alternatives. Advances in artificial intelligence, particularly in natural language processing, offer new opportunities for automating threat detection and log analysis.

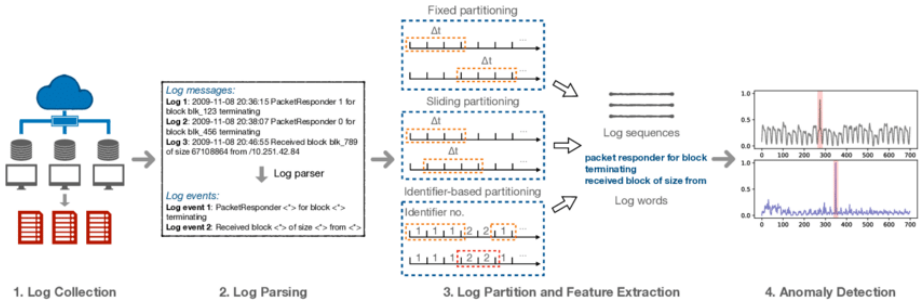

Figure 1: Overall Framework of Log based Anomaly Detection [6]

This project builds on such innovations to explore how a language model can be adapted for cybersecurity tasks. The goal is to bridge the gap between technical complexity and practical usability, providing a scalable solution that can evolve with emerging threats.

3 Literature Review

3.1 Security Operations Centers (SOC)

A Security Operations Center (SOC) is a centralized facility where an organization's information systems are monitored, assessed, and defended against cybersecurity threats. SOCs operate as the frontline of defense, combining human expertise, processes, and technological tools to ensure continuous visibility and protection of digital infrastructures. Their primary functions include real-time monitoring of network traffic, incident detection, forensic analysis, and coordinated response to security events [1].

The role of a SOC has become increasingly critical as organizations face growing volumes of sophisticated cyberattacks.

Figure 2: SOC operations by Wallarm [1]

In practice, the efficiency of SOC operations depends not only on the deployment of advanced monitoring technologies but also on the maturity of incident response procedures and the expertise of security analysts. Research has also highlighted the increasing relevance of automation and artificial intelligence within SOC environments, enabling faster triage of alerts and reducing analyst fatigue in the face of high false-positive rates.

3.2 Log definition

A log is a machine-made file or record containing information about activities in a computer system. According to Karen Kent(2006) [4], Logs are composed of log entries; each entry contains information related to a specific event that has occurred within a system or network. Many logs within an organization contain records related to computer security.

format standard de logs.

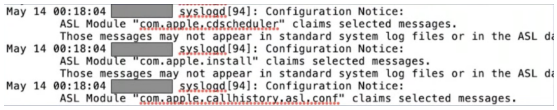

Figure 3: Example of Logs configured in Crowdstrike [5]

These computer security logs are generated by many sources, including security software, such as antivirus software, firewalls, and intrusion detection and prevention systems; operating systems on servers, workstations, and networking equipment; and applications[4].

3.3 LogLLM

This section introduces the `LogLLM` model and how it is used to analyze and process log files in a cybersecurity setting. The core idea is to combine a lightweight encoder that specializes in log tokens with a general-purpose causal language model capable of reasoning over sequences and producing a concise natural-language classification.

3.3.1 Definition

LogLLM is a hybrid architecture for log–sequence classification. Given a sequence of log messages $\mathcal{S} = [m_1, \ldots, m_T]$, each message is first embedded with a pretrained encoder (BERT). The resulting vectors $h_i \in \mathbb{R}^{768}$ are linearly projected to the hidden size of a causal LLM (Llama family). The projected sequence is then concatenated—in the embedding space—with a fixed instruction prefix and a short answer prefix. The causal LLM autoregressively generates a short decision string.

Concretely, the pipeline proceeds as follows:

1. **Message encoding:** BERT produces one vector per message (using the `[CLS]` / pooler output).
2. **Projection:** a linear layer $W \in \mathbb{R}^{d_{\text{LLM}} \times 768}$ maps BERT vectors to the LLM hidden size.
3. **Conditioning:** instruction tokens (prompt) + projected message embeddings + answer prefix are fed to the causal LLM.
4. **Decision:** the LLM outputs a short, normalized decision string (e.g., *"The sequence is normal."*).

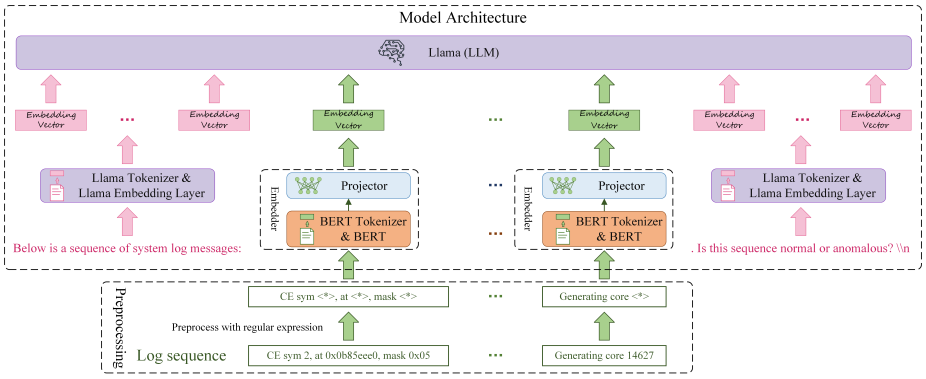

Figure 4: Overview of the LogLLM processing and interpretation workflow [3].

The open-source implementation is intentionally modular: the encoder, the projector, the prompt template, and the causal LLM can be swapped or fine-tuned independently (e.g., via LoRA adapters and 4-bit quantization).

4 Methodology

4.1 Modification of LogLLM

In this project, I introduced two main changes to `LogLLM`:

1) Prompt modification. The instruction prompt was redesigned to better contextualize a log sequence for anomaly detection. In practice, the text prefix surrounding the BERT embeddings was rewritten so that the model first sees a concise task description and then the sequence. A representative template is:

```
''Below is a sequence of system log messages:  <EMBEDDED_SEQUENCE>.
Is this sequence normal or anomalous?''
```

At generation time, an answer prefix `''The sequence is''` is appended in the embedding space to steer the LLM toward a short, well-formed completion.

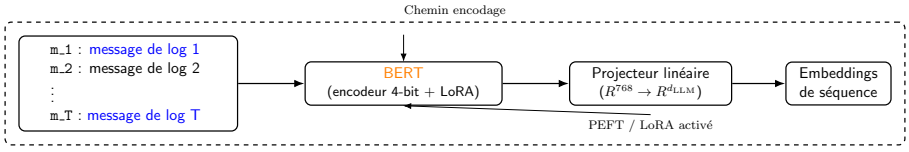

Figure 5: Encoding schema (LogLLM): message encoding with BERT, linear projection, and construction of the embedded prompt.

The revised prompt improves task grounding: the LLM receives an explicit instruction, the embedded evidence (projected message vectors), and a constrained opening for the answer, which together reduce off-task generations.

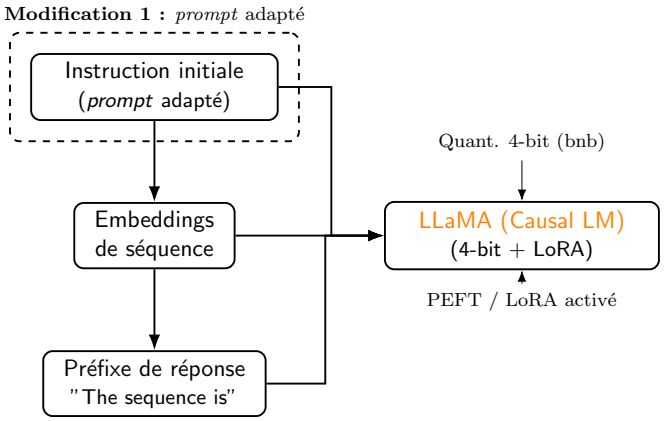

Figure 6: Context construction and LLaMA conditioning with instruction and answer prefixes.

2) Output normalization. The output space was constrained to a *normalized* binary decision to increase robustness and ease of evaluation. Instead of free-form text, the model is required to complete with one of:

‘‘`The sequence is normal.`’’ or ‘‘`The sequence is anomalous.`’’

This restriction simplifies downstream parsing, reduces label ambiguity, and makes metrics (precision/recall/F1/accuracy) less sensitive to phrasing.

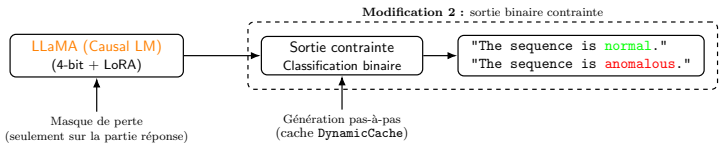

Figure 7: Output of the modified LogLLM: constrained, normalized decision string.

4.2 Addition of an Interpretation Stage

Beyond the binary decision, we added a post-processing stage that *interprets* anomalous sequences and translates them into analyst-friendly summaries with a severity estimate. The goals are (i) to bridge the gap between classification and actionability, and (ii) to support triage.

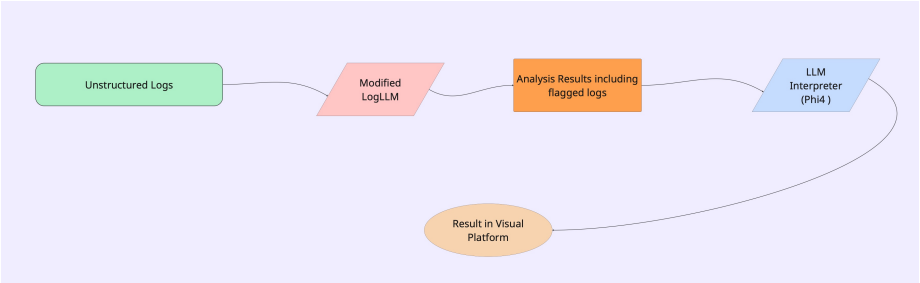

Figure 8: Full Implementation steps

Inputs and mechanism. The interpreter receives (a) the original log subsequence flagged as anomalous, (b) the model's decision, and (c) optional context (host, time window, component). It then produces a structured assessment using a lightweight LLM (e.g., *Phi-4*) guided by a strict template and guarded by rule-based checks. Concretely:

1. **Summarization:** produce a short natural-language summary of what the logs indicate (one or two sentences).
2. **Evidence extraction:** list salient indicators (e.g., repeated authentication failures, sudo escalation, unusual network destinations).

3. **Severity mapping:** assign a level from {*Low, Medium, High*} based on heuristics (frequency, diversity of sources, privilege impact) and LLM reasoning.

4. **Action hints:** suggest immediate next steps (e.g., isolate host, rotate credentials, increase logging, correlate with IDS).

Output format. The interpreter emits a compact txt-like record for downstream systems and a human-readable note for analysts, for example:

```
{ "decision": "anomalous", "severity": "High", "summary": "Multiple
failed SSH logins followed by successful sudo.", "indicators": ["auth
failure x45", "sudo from svc-user", "IP 203.0.113.5"], "suggested_actions":
["lock account", "rotate keys", "review sudoers"] }
```

This stage translates model output into operationally meaningful guidance without retraining the classifier.

4.3 Use of Streamlit

We developed a lightweight *Streamlit* web application to display intermediate artifacts, execute the end-to-end pipeline, and orchestrate the different stages of the workflow (data ingestion, preprocessing, inference, post-hoc interpretation, and evaluation).

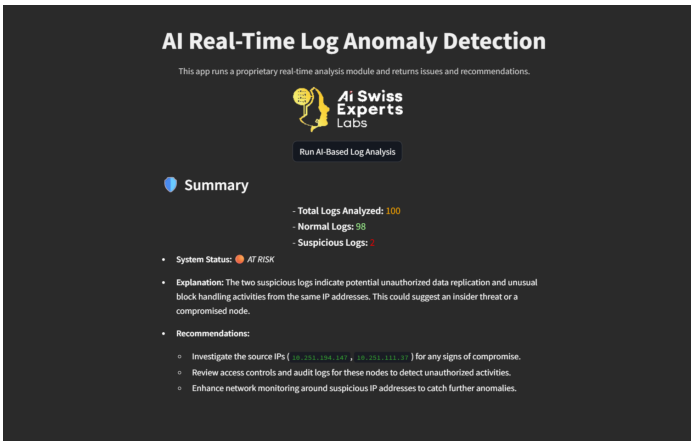

Figure 9: AI SOC interface (Streamlit prototype) used to run, monitor, and compare experiments.

The app exposes a small set of controls to deploy the real-time log analysis dataset (e.g., `HDFS_v1`, `CIC_UNSW_NB15`), display intermediate results, and toggle the interpretation stage.

Benchmarking instrumentation. To support consistent, comparable experiments, the application integrates standard benchmarking utilities: `scikit-learn` for classification metrics (precision, recall, F1, accuracy), `time` for wall-clock measurements, and `psutil` (plus optional `codecarbon`) for resource/energy observations. Results are exported as CSV/JSON and can be reloaded in the interface for side-by-side comparisons.

4.4 Use of HDFS Logs

We applied the model to the `HDFS_v1` log corpus. The dataset was defined as sequences of chronologically ordered log messages, each sequence labeled as *normal* or *anomalous*. Training was conducted in July (train/validation/test split), with BERT used as the encoder for message-level representations and a linear projector to match the hidden size of the causal LLM. The LLM (Llama family) was conditioned via an explicit instruction prompt and an answer prefix, and fine-tuned using parameter-efficient adapters (LoRA) under 4-bit quantization when appropriate. We report per-sequence metrics on the held-out test set and analyze typical failure modes (e.g., rare patterns, ambiguous templates).

4.5 Use of CIC UNSW NB15 Logs

We further experimented with the labeled `CIC_UNSW_NB15` logs. The same pipeline and hyperparameter schedule were used for comparability, with minor adjustments to sequence construction to respect dataset-specific labeling. We then contrasted performance against `HDFS_v1`, highlighting the effect of distributional shift (diverse attack types, class imbalance) on recall and overall F1, and we inspected the interpreter's summaries for high-severity cases.

5 Results

5.1 Accuracy

On `HDFS_v1`, the model achieves high accuracy with balanced precision and recall. In particular, we observe:

- **HDFS_v1** — Precision: **99.23%**, Recall: **100.00%**, F1: **99.61%**, Accuracy: **99.98%**.
- **CIC_UNSW_NB15** — Precision: **99.02%**, Recall: **91.60%**, F1: **95.17%**, Accuracy: **90.85%**.

The slight degradation on `CIC_UNSW_NB15` is consistent with a more heterogeneous anomaly taxonomy and a more challenging class distribution. Nonetheless, the constrained output format and prompt design help maintain stable F1 across datasets.

5.2 Benchmarking of the Solutions

Dataset	**HDFS_v1**	**CIC_UNSW_NB15**
Anomaly type	Failures and errors	Nine attack types
# test rows	115,014	5,728
Test size	211 MB	78 MB
Precision	0.9923	0.9902
Recall	1.0000	0.9160
F1-score	0.9961	0.9517
Accuracy	0.9998	0.9085

Table 1: Comparison of `LogLLM` performance across two datasets (`HDFS_v1` and `CIC_UNSW_NB15`). Metrics are computed on the held-out test split. [2]

Experimental setup: unless stated otherwise, all measurements were obtained on a single NVIDIA H100 GPU with BERT as the encoder, Llama (LogLLM) as the causal model, and a lightweight interpreter (Phi-4) for post-hoc analysis.

5.3 Efficiency

This section analyzes the efficiency of the proposed system from three complementary angles: (i) algorithmic complexity, (ii) empirical runtime, and (iii) resource usage (energy and cost). Unless otherwise stated, experiments were executed on a single NVIDIA H100 GPU, with BERT as the encoder, Llama (LogLLM) as the causal model, and a lightweight interpreter (Phi-4) for post-hoc analysis.

Algorithmic considerations. Let N denote the number of log messages in a sequence and M_i the number of tokens in message i. Message encoding with BERT requires self-attention over each message, yielding a per-message cost that scales approximately as $\mathcal{O}(M_i^2)$ per layer; for bounded message length this is effectively linear in N. After projection to the LLM hidden size, the causal LLM consumes an embedded sequence whose effective length is $\approx N + C$ (an instruction prefix plus an answer prefix of constant size C), and autoregressively generates a short completion (a few tokens). The dominant term is attention over the context, which scales as $\mathcal{O}((N + C)^2 d)$ per layer for hidden size d. In practice, we keep the completion short (binary label) and leverage 4-bit quantization and KV caching, so that wall-clock time scales near-linearly with N for the sequence lengths considered in this study.

Runtime and resource usage. Table 2 reports end-to-end runtimes together with energy consumption and an estimated monetary cost proxy. These figures include data loading, encoding, inference, and (when enabled) interpretation.

Dataset	**HDFS_v1**	**CIC_UNSW_NB15**
Runtime (s)	2 155.09	37.08
Energy (Wh)	51.022	1.453
Cost (cents)	1.480	0.055

Table 2: End-to-end runtime and resource usage of `LogLLM` on two datasets.

Real-time analysis. To stress-test the system under near real-time conditions, we measured the time required to process fixed-size batches of incoming logs within the AI SOC interface. Figure 10 summarizes the latency profile observed during continuous ingestion experiments.

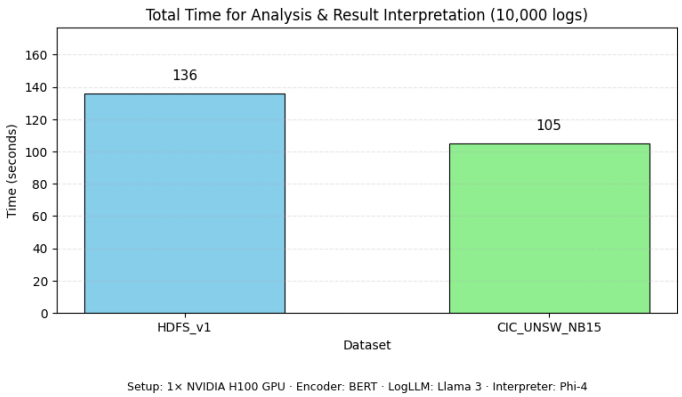

Figure 10: Real-time log analysis latency in the AI SOC interface.

Latency–quality trade-off. We also report an aggregate run-time (minutes) together with the resulting F1-score for the complete pipeline (classification plus interpretation). The numbers in Table 3 reflect a fixed-size evaluation slice for comparability across datasets.

Dataset	**HDFS_v1**	**CIC_UNSW_NB15**
Time (mm:ss)	2:14	1:45
F1-score	0.9961	0.9517

Table 3: End-to-end pipeline latency and F1-score for `HDFS_v1` and `CIC_UNSW_NB15`.

Practical takeaways. (1) Constraining the output space to a normalized label keeps the generated completion short, which materially reduces decoding time. (2) 4-bit quantization with LoRA adapters strikes a favorable accuracy–efficiency balance, lowering energy while preserving F1. (3) Batching sequences with left padding (to exploit KV caching efficiently) reduces per-sequence latency at moderate batch sizes. (4) The interpretation stage adds a small but predictable overhead; its latency can be capped by templated prompts and a fixed max generation length.

6 Conclusion and future work

6.1 Conclusion

This project demonstrates that a hybrid encoder–decoder approach (`LogLLM`) can deliver accurate and operationally useful anomaly detection over heterogeneous log corpora, while remaining efficient enough for near real-time use on a single high-end GPU. Methodologically, we introduced two targeted modifications—prompt redesign and output normalization—that improved robustness and simplified downstream evaluation. We further added a lightweight interpretation stage that translates anomalous sequences into analyst-ready summaries with severity estimates, bridging the gap between raw detection and actionable response.

6.2 Limitations and future work

First, attention cost scales quadratically with sequence length; future work will investigate long-context architectures and sparse attention to further reduce latency for very long windows. Second, while the interpreter produces concise rationales, systematic human-in-the-loop evaluation remains necessary to validate severity assignments.

References

[1] Mukhadin Beschokov. Security operations center (soc), September 2025. Reviewed by Ivan Novikov.

[2] Romain Christen. Security monitoring assisté par ia destiné aux pmes. Master's thesis, School of Criminal Science, University of Lausanne, Faculty of Law, Criminal Justice and Public Administration, University of Lausanne, 2025. In partnership with SeLabs.ai and SecuLabs SA. Under the direction of Prof. Thomas Souvignet. Supervised by Dr. Lionel Beaugé and Dr. Giao Nguyen (SeLabs.ai), Mr. Samuel Bovy and Ms. Emilie Kuhn (SecuLabs SA).

[3] Wei Guan, Jian Cao, Shiyou Qian, Jianqi Gao, and Chun Ouyang. LogLLM: Log-based Anomaly Detection Using Large Language Models, 2024–2025. Version 5 as of April 14, 2025; GitHub: `https://github.com/guanwei49/LogLLM`.

[4] Karen Kent and Murugiah Souppaya. Guide to computer security log management. Special Publication 800-92, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2006. Recommendations of the National Institute of Standards and Technology.

[5] Arfan Sharif. Log files explained, December 2022. Cybersecurity 101: Introduction to Next-Gen SIEM.

[6] Jinyang Zhang, Wei Xu, Adam Oliner, Linghuang Wang, Zhenyu Xu, Min Wang, Yongqiang Li, and Qiang Lin. Experience report: Deep learning-based system log analysis for anomaly detection. *arXiv preprint arXiv:2107.05908*, July 2021. License: CC BY-NC-ND 4.0.