



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Vilnius
University

Shubham Negi

IP Reputation Assisted Automatic Threat Blocking Based on Local AI

Master Dissertation

Master's on International Cybersecurity and Cyberintelligence

Work developed under the guidance of:

Professor Henrique Manuel Dinis Santos

Rodrigo Araujo

June 2026

Acknowledgments

I would like to thank Professor Henrique Manuel Dinis Santos, my academic supervisor, for the academic direction and support he provided. Our discussions were useful, and I appreciated having him available at the points where guidance was needed—he gave me the space to pursue the research independently, which is ultimately how the work came to be what it is.

At Eurotux Informatica, I was fortunate to have Rodrigo Araujo as my company mentor. Rodrigo was the person I had the most discussions with throughout this work—he went out of his way to provide the technical details I needed about Eurotux’s internal tooling, connected me with the right people inside the organisation, offered practical perspective at every stage where the research met real operational constraints, and was genuinely available whenever I needed to think something through. Alongside him, André Rocha was genuinely invaluable—someone who was always willing to listen when something wasn’t working, offered inputs whenever I needed a second pair of eyes, and helped set up access to company tools and resources wherever that was needed. I would also like to thank Cátia Machado, who handled the administrative and onboarding side of the internship smoothly and made the whole process far less daunting than it might otherwise have been.

To my parents and sister—thank you for everything, as always. The support from home across four countries and two years meant more than I can put into words here.

And then there are the people who made this particular semester in Braga what it was. Shashank—my best friend this semester, flatmate, and coursemate, who happened to be doing his internship at Eurotux at exactly the same time—was with me through all of it, the tough stretches and the good ones alike, and helped enormously both with motivation and just by being there. Christopher and Gabriel, my other flatmates and coursemates, kept things lively with the side conversations and the random discussions about thesis topics that made the long days a lot more manageable. And Prabin and Naren, the owners of Mid Town Ramen near the university, deserve a mention too—many study sessions were powered by their food, and the time spent there was genuinely some of the best this semester had to offer. Guilherme, Gustavo, and the rest of the coursemates who were part of this programme across Braga, Granada, Vilnius, and Padova also deserve a mention here—two years of a joint master’s spanning four countries is not a small thing, and the people who shared that journey are a big part of what made it worthwhile. Eesha, a friend from my time in Padova, also deserves a mention. And Pravesh, a friend from back home who has known me for long, deserves a mention too—that kind of friendship, the sort that stays steady regardless of which country you happen to be living in at the moment, is something I am genuinely grateful for.

ॐ असतो मा सद्गमय ।

तमसो मा ज्योतिर्गमय ।

मृत्योर्माऽमृतं गमय ॥

Om asato mā sad gamaya ।

tamaso mā jyotir gamaya ।

mṛtyor mā'mṛtaṁ gamaya ॥

"From the unreal, lead me to the real.

From darkness, lead me to light.

From death, lead me to immortality."

Brihadaranyaka Upanishad – Pavamana Mantra

Abstract

Security Operations Center (SOC) teams face numerous challenges in managing high alert volumes and operational burden of manual log analysis in the existing threat blocking system pipelines. Static IP blocklists—one of the most common first-layer controls—disagree with each other by as much as 80% for the same address (Lewis, Tambaliuc, Narman, & Yoo, 2020), meaning a substantial share of analyst triage effort is spent resolving contradictions between reputation sources rather than investigating genuine threats. Current solutions rely on cloud based Threat Intelligence platforms which introduce data sovereignty risks, raising compliance incompatibility with GDPR regulatory guidelines. IP reputation has been a good discriminative feature when detecting malicious traffic and identifying threat by current industry standard frameworks or Machine Learning models and is primarily used as a first line of defence considering its low resource consumption and easier implementation. A lot of first line SOC defence relies on IP reputation score based on static blocklists which introduces challenges like high false positive rates and in turn alert fatigue to the analyst, which is why current state of the art methods rely more on behavioural signals from the network traffic. Behaviour-based ML models address this limitation but most are trained on a single synthetic dataset and do not generalise well when tested against a different network environment or attack distribution (Amin, Shen, Ishak, Manickam, & Karuppayah, 2026), a cross-domain generalisation gap that this thesis directly targets. This thesis discusses automatic threat blocking tied with IP reputation enforcement, compares different ML techniques for the behaviour driven automation, and proposes an IP reputation enforced threat blocking framework driven by locally deployable AI system. The rapid advancements in LLM model performance in terms of analysing and deriving meaningful inferences from the system logs and the availability of open-source pre-trained LLMs on cybersecurity corpus have shifted the current paradigm in the field, but with it comes a significantly high infrastructure need and questionable feasibility of deployment in in-line network operations. As a result the implementation and performance results of the models have also been provided with the feasibility study in terms of in-line deployment of the complete framework.

Index

1	Introduction	1
1.1	Contextualisation	1
1.2	Problem Statement	2
1.3	Research Questions and Objectives	3
1.4	Dissertation Structure	5
2	Literature Review	6
2.1	IP Reputation Systems and Architecture	6
2.2	Behavioral Flow Analysis and Intrusion Detection	9
2.3	IDS Datasets and the Cross-Domain Generalisation Problem	10
2.4	Transformer-Based Models for Network Intrusion Detection	11
2.5	SecureBERT 2.0 and Log-Semantic Anomaly Detection	12
2.6	Knowledge Gap and Research Positioning	14
3	Methodological Approach	16
3.1	Research Paradigm: Design Science Research	16
3.2	Literature Review Methodology	17
3.3	Experimental Design and Implementation Methodology	18
3.3.1	Loss Function Experimentation	18
3.3.2	Training Infrastructure and Kaggle Constraints	19
3.3.3	Evaluation Metric and Threshold Methodology	20
3.3.4	Cross-Dataset Training and Label Handling	20
3.3.5	Iterative Refinement Cycle	23
3.4	Feature Selection: Mutual Information and Conditional Entropy Pruning	23
3.5	Dataset Selection Methodology	25
4	Work Done	27
4.1	Framework Architecture	27
4.2	Design Justification	28
4.2.1	Limitations of Unsupervised Anomaly Detection as a First Stage	29
4.2.2	Justification for Supervised Flow-Based Classification	29
4.2.3	Complementarity of Flow-Level and Log-Level Analysis	29
4.2.4	Cross-Validation and Explainability	30
4.2.5	Operational Resilience and Business Continuity	30

4.2.6	Alignment with SOC Detection and Analysis Phases	30
4.2.7	Summary	30
4.3	Implemented Components	31
4.3.1	Distributed Honeypot Infrastructure	31
4.3.2	Combined CIC-IDS Dataset Construction	32
4.3.3	AIT Log Dataset Construction	33
4.3.4	Flow Classification Experiments	38
4.3.5	Final Model Training	41
4.4	Proposed Components	45
5	Results and Discussion	46
5.1	Feature Selection Results	46
5.1.1	Pipeline Stage Summary	46
5.1.2	Mutual Information Scores: Top 20 Features	47
5.1.3	Conditional Entropy Redundancy Drops	48
5.1.4	Discussion	48
5.2	Tier-1 Flow Classifier: Training Setup	49
5.2.1	Hardware and Training Configuration	49
5.2.2	Training Duration and Epoch Progress	49
5.3	Tier-1 Flow Classifier: Test Evaluation	50
5.4	Per-Class Analysis and Discussion	51
5.4.1	Dominant and Mid-Frequency Attack Classes	53
5.4.2	Heartbleed and Infiltration: An Artefact of Label Semantics	53
5.4.3	Macro F1 in Context	54
5.5	Comparison with Baseline Models from Literature	54
5.5.1	Throughput and Operational Latency	56
5.6	Tier-2 Log Classifier: Results	57
5.6.1	Training Setup	57
5.6.2	Test Evaluation and Overall Metrics	58
5.6.3	Per-Label Analysis	59
5.6.4	Benign Row False Positive Analysis	61
5.6.5	Inference Latency and Throughput	63
5.6.6	Cross-Scenario Generalisation and Limitations	64
5.7	Chapter Summary	65

6 Conclusion	67
Bibliography	71

Figures Index

The proposed IP reputation-enforced threat blocking framework. The Local IP Reputation Engine acts as the central orchestrator: it gates every incoming flow against the IP blocklist, dispatches unknown IPs to the two-tier AI pipeline, and consolidates the returned scores into a single enforcement decision.	27
Normalised confusion matrix for SecureBERT 2.0 (checkpoint-45000) on the combined CIC-IDS test set. Near-perfect recall across 14 of 16 classes; Heartbleed and Infiltration are the primary off-diagonal sources.	52
Per-class F1-score for SecureBERT 2.0 (checkpoint-45000); dashed line marks macro F1 = 0.922. Heartbleed is the sole red-tier class (< 0.50) due to its 2-sample test support.	52
One-vs-rest ROC curves for all 16 classes; macro-averaged AUC = 0.9994. Heartbleed achieves ROC-AUC 0.9994 despite F1 = 0.000, confirming sound probability calibration.	52
Single-sample latency (left) and batched throughput (right) from 50 timed runs on the Kaggle T4 GPU; mean 63.04 ms and 16 samples/s.	56
Per-label F1-score for the Tier-2 log classifier (checkpoint-4800, Shaw test); dashed line marks macro F1 = 0.765. <code>recon_scan</code> is the sole red-tier label despite a support of 415, owing to threshold transfer failure rather than representational failure.	60
Precision-recall curves per label on the deduplicated Shaw test set. All labels except <code>recon_scan</code> achieve high area under the PR curve, confirming that the probability calibration is sound even for the labels with low F1 at the operating threshold.	60
Per-label TP/FP/FN breakdown for the Tier-2 log classifier (checkpoint-4800, Shaw test), sorted by support. The bar widths confirm that false positives are confined to a small number of labels and are negligible relative to the true positive counts for the dominant attack activities.	62

Tables Index

Per-scenario occurrence counts for all 13 attack labels in AIT-LDS-v2.1.	36
Per-label occurrence counts in the AIT-LDS-v2.1 training corpus and Shaw test partition.	37
Feature selection pipeline: stage-by-stage reduction from 85 raw columns to 50 final features.	46
Top 20 features by mutual information score (descending), after zero-MI removal and CE pruning.	47
Features dropped by conditional entropy pruning (non-zero MI only); $\Delta H(Y) < \varepsilon = 0.005$ bits triggered each drop.	48
Overall test metrics for SecureBERT 2.0 (checkpoint-45000), combined CIC-IDS test set ($n = 1,127,684$).	50
Per-class metrics for SecureBERT 2.0 (checkpoint-45000), sorted by F1-score descending.	51
Tier-1 SecureBERT 2.0 vs. published baselines. [†] multi-dataset benchmark; [‡] non-CIC-IDS benchmark. Latency is per-sample; ‘—’ = not reported; Acc.* not comparable to macro F1.	55
Inference latency and throughput for the Tier-1 classifier (Kaggle T4, FP16 AMP).	56
Overall test metrics for the Tier-2 log classifier (SecureBERT 2.0, checkpoint-4800) on the deduplicated Shaw cross-scenario test set ($n = 39,109$).	58
Per-label metrics for the Tier-2 log classifier (SecureBERT 2.0, checkpoint-4800) on the deduplicated Shaw test set, sorted by F1 descending. Thresh = per-label sigmoid threshold tuned on the validation set; Pred-Pos = number of positive predictions made.	59
False positive counts and rates on the $\approx 33,311$ benign rows (all-zero ground-truth label vectors) of the deduplicated Shaw test set.	62
Inference latency and throughput for the Tier-2 log classifier (SecureBERT 2.0 checkpoint-4800, Kaggle T4, FP16 AMP).	63

Acronyms

AIT-LDS	AIT Log Dataset.
AMP	Automatic Mixed Precision.
API	Application Programming Interface.
APT	Advanced Persistent Threat.
ASL	Asymmetric Loss.
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge.
AUC	Area Under the Curve.
BCE	Binary Cross-Entropy.
BERT	Bidirectional Encoder Representations from Transformers.
CE	Conditional Entropy.
CIC	Canadian Institute for Cybersecurity.
CICIDS	CIC Intrusion Detection System dataset.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
CSV	Comma-Separated Values.
CVE	Common Vulnerabilities and Exposures.
CVSS	Common Vulnerability Scoring System.
DDoS	Distributed Denial of Service.
DDP	Distributed Data Parallel.
DGA	Domain Generation Algorithm.
DL	Deep Learning.
DNS	Domain Name System.
DoS	Denial of Service.
DPI	Deep Packet Inspection.
DSR	Design Science Research.
DTs	Dynamic Threat Score.
FN	False Negative.

FP False Positive.

FP16 16-bit floating point precision.

FTP File Transfer Protocol.

GCN Graph Convolutional Network.

GDPR General Data Protection Regulation.

GNN Graph Neural Network.

GPT Generative Pre-Trained Transformer.

GPU Graphics Processing Unit.

GRU Gated Recurrent Unit.

HIDS Host-based Intrusion Detection System.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IAT Inter-Arrival Time.

IDS Intrusion Detection System.

IMA Integrity Measurement Architecture.

IOC Indicator of Compromise.

IoT Internet of Things.

IP Internet Protocol.

IPS Intrusion Prevention System.

IS Information Systems.

ISO International Organization for Standardization.

IT Information Technology.

JSON JavaScript Object Notation.

LLM Large Language Model.

LoRA Low-Rank Adaptation.

LSTM Long Short-Term Memory.

MI Mutual Information.

MITM Man-in-the-Middle.

MITRE MITRE Corporation.

ML Machine Learning.

MTTA Mean Time to Acknowledge.

MTTR Mean Time to Respond.

NIDS Network Intrusion Detection System.

NIST National Institute of Standards and Technology.

NLP Natural Language Processing.

OSINT Open-Source Intelligence.

OWASP Open Web Application Security Project.

PCAP Packet Capture.

PDF Portable Document Format.

PVLAN Private Virtual Local Area Network.

RAM Random Access Memory.

RF Random Forest.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

RQ Research Question.

SIEM Security Information and Event Management.

SMTP Simple Mail Transfer Protocol.

SOAR Security Orchestration, Automation and Response.

SOC Security Operations Center.

SSH Secure Shell.

SSL Secure Sockets Layer.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

TI Threat Intelligence.

TLS Transport Layer Security.

TN True Negative.

TP True Positive.

TTPs Tactics, Techniques, and Procedures.

UDP User Datagram Protocol.

VLAN Virtual Local Area Network.

VPN Virtual Private Network.

VRAM Video Random Access Memory.

WLAN Wireless Local Area Network.

XDR Extended Detection and Response.

XGBoost Extreme Gradient Boosting.

1 Introduction

IP reputation scoring has been of prime importance of use in SOC threat blocking pipelines as one of the easiest and quickest ways to filter malicious network traffic. This is because a simple lookup against a reputation or blocklist is computationally cheap and easy to develop/deploy in form of an API or script without the need of deep packet inspection methodologies (Lewis et al., 2020; Xu et al., 2025). Still, first line SOC defence in many organizations comprises of such IP reputation based methodologies based on static blocklists like AbuseIPDB, Shodan etc., even though the results from these sources disagree among themselves by up to 80% in some cases for the same IP (Lewis et al., 2020). Also the facts that IP addresses usually get reassigned, compromised hosts may later be cleaned and attackers quickly move to new infrastructure makes static blocklists, an unreliable representation of the current state of the IP (Miranda-García, Crespo-Martinez, Guerrero-Higueras, & Matellán-Olivera, 2023). All of this results in high false positive rates and in turn affects the SOC analyst by alert fatigue which keeps getting worse as the traffic generated by ever growing internet userbase increases while there is a shortage of analysts capable of performing manual log analysis and triage in Intrusion Detection Systems (IDS) (Miguel-Diez et al., 2026; Habibzadeh, Feyzi, & Atani, 2025; Lasantha, Abeysekara, & Maduranga, 2024b).

1.1 Contextualisation

To address the limitations described above, behavior based signals were preferred over the blocklists which gave way for Machine Learning (ML) models to be introduced in IDS systems and in some threat blocking pipelines, these were also used to model reputation as a Dynamic risk score which gets updated based on external threat intelligence and local telemetry data. However, most of these models are trained on a single IDS dataset (most of them being synthetic) and do not generalize well when tested on a different network environment or attack distribution and this problem comes up frequently in recent IDS literature (Amin et al., 2026; Hu et al., 2025; Mondragon, Branco, Jourdan, Gutierrez-Rodriguez, & Biswal, 2025). Part of the reason is also that tree-based models and most traditional classifier or anomaly detection models essentially learn the exact feature columns produced by the flow conversion tool like CICFlowMeter, Zeek, Netflow etc., used when the dataset was being created or the flows were being recorded from the network traffic. So if production traffic uses a different flow converter, the feature names and value distributions shift in ways the model has never seen during training (Amin et al., 2026; Engelen, Rimmer, & Joosen, 2021). Another problem is that most behavioral classifiers treat the prediction as the final output and never feed it back into any enforcement mechanism, so the same IP ends up triggering expensive model inference on every new connection instead of only on new or unknown sources (Siam et al., 2025). The cloud dependency of current threat intelligence platforms adds to the problems discussed above as it introduces data sovereignty risks for organizations under GDPR or similar compliance requirements, which is the reason why a local AI based system is better positioned for SOC environments (Aghaei, Jain, Arun, & Sambamoorthy, 2025; Habibzadeh et al., 2025).

For the problems described above, this research analyses existing methods, datasets and models for IP

reputations, IDS systems, log based anomaly detection systems, LLM usage in SOC pipelines and proposes a **locally deployable, IP reputation-enforced threat blocking framework**. The framework can be described as a system which first checks IP of the incoming traffic against a local reputation database so known bad-IPs are blocked at near-zero cost and the two tiered ML pipeline only runs for the new or unknown IPs and this is what makes it as an IP reputation enforced threat blocking architecture rather than a standalone ML based IDS. BERT based models which tokenize each flow record as `FeatureName=value` pairs learn the feature names and their values as semantic units rather than fixed column positions, which makes them more robust to the converter induced drift and this is one of the reasons why SecureBERT 2.0 with proper feature level tokenization is used and proposed in this research (Hu et al., 2025). The first tier is the flow classification component for which this dissertation presents a comparison of CysecBERT and SecureBERT 2.0 fine-tuned on a combined multi-source CICIDS dataset combined specifically for this research from the Distrinet reproductions of CIC IDS 2017 and CSECIC IDS 2018 (Sharafaldin, Lashkari, & Ghorbani, 2018; Engelen et al., 2021), looking at the effect on macro F1 score of Mutual Information based feature selection and class weighting techniques for handling imbalances. A **Dynamic Threat Score (DTS)** scoring algorithm modelled on the IP SafeGuard framework (Siam et al., 2025), along with a log-refinement tier using SecureBERT 2.0 fine-tuned on the AIT log dataset v2.1 (Landauer, Skopik, et al., 2023), are proposed as the remaining components of the full architecture described in Chapter 4.

1.2 Problem Statement

The core problem this dissertation addresses is the absence of a unified, locally deployable framework that connects IP reputation enforcement, behavioral ML classification, and analyst triage into a single coherent pipeline. Each of these components exists in the literature in some form, but they are typically studied in isolation: reputation systems do not feed back from ML outputs, ML classifiers are evaluated on single datasets without cross-domain testing, and LLM-based triage is explored without a realistic model of where in a high-throughput pipeline it can actually be placed.

More specifically, four interconnected gaps motivate this work. Static IP blocklists are unreliable as a first-line defence because they are sourced from external platforms that frequently disagree with each other and do not reflect the current state of an IP address (Lewis et al., 2020; Miranda-García et al., 2023); yet in many SOC deployments they remain the primary or only automated blocking mechanism. Behavioral flow classifiers trained on a single IDS dataset suffer from well-documented cross-domain generalization failure when tested on traffic from a different network environment or processed through a different flow conversion tool (Amin et al., 2026; Engelen et al., 2021; Mondragon et al., 2025); this limits their practical deployability in production SOC environments. The ML prediction is typically not fed back into any reputation or enforcement mechanism, meaning that repeated inference is triggered for already-scored IPs, which is computationally expensive and introduces unnecessary latency (Siam et al., 2025). And the semantic gap between a model's classification label and the analyst action it should trigger

remains largely unaddressed in architectures that do not include an explanation or reasoning layer, even though the evidence suggests that decoder-LLM triage can meaningfully close this gap if deployed selectively rather than inline on every flow (Habibzadeh et al., 2025; Guan, Cao, Zhao, Gu, & Qian, 2025).

This dissertation proposes a framework that addresses all four gaps together: a Tier-0 local blocklist that checks every connection first, a local honeypot that independently seeds it, a two-tier ML pipeline (Tier 1 flow classification, Tier 2 log refinement) that runs only for new or unknown IPs, a Dynamic Threat Score that combines external threat intelligence with local ML outputs to risk-score IPs and update the blocklist, and a decoder LLM that produces evidence-grounded triage reports for the subset of flows that reach Tier 2. The framework is designed to operate entirely on local infrastructure, with no cloud dependency for inference or telemetry.

1.3 Research Questions and Objectives

Following the knowledge gap identified and discussed above, this dissertation focuses on the following **Research Questions (RQs)**:

- **RQ1:** How do BERT-based flow classifiers (CysecBERT and SecureBERT 2.0) compare to published tree-based IDS baselines in terms of macro F1, and what is the effect of Mutual Information-based feature selection and class-weighting techniques on classification performance across the combined multi-source CICIDS dataset?
- **RQ2:** To what extent does training on a combined multi-source CICIDS dataset, with BERT-based semantic tokenization of flow features, improve cross-domain generalization compared to models trained on a single IDS dataset?
- **RQ3:** To what extent does the IP reputation gate—the Tier-0 blocklist check—reduce the volume of flows that require ML pipeline inference, and what does this imply for inline deployment feasibility in a production SOC in terms of latency and throughput?
- **RQ4:** Can a two-tier architecture (Tier 1 flow filter followed by Tier 2 log refinement and decoder-LLM triage) provide the behavioral depth required for evidence-based analyst triage without requiring full LLM inference on every flow, and does the decoder-LLM output meaningfully bridge the semantic gap between a classification label and an actionable SOC response?

These research questions are addressed through the following objectives:

- Design and implement a locally deployable, IP reputation-enforced threat blocking framework comprising a Tier-0 blocklist, a local honeypot seeding component, a two-tier ML pipeline (SecureBERT 2.0 fine-tuned for flow and log classification), a Dynamic Threat Score generator, and a decoder-LLM triage layer.

- Build and curate a combined multi-source CICIDS dataset from the Distrinet reproductions of CIC-IDS 2017 and CSECIC-IDS 2018 (Sharafaldin et al., 2018; Engelen et al., 2021), applying Mutual Information-based feature selection and class-weighting to address class imbalance, and use this dataset to fine-tune and evaluate BERT-based flow classifiers.
- Fine-tune SecureBERT 2.0 on the AIT log dataset v2.1 (Landauer, Skopik, et al., 2023) as the Tier 2 log refinement component, and evaluate its classification performance and its role in reducing the load on the decoder-LLM layer.
- Evaluate the IP reputation gate's effect on inference volume and inline deployment feasibility, and assess whether the Dynamic Threat Score, seeded by both external threat intelligence and local ML outputs, produces reputation updates that are more current and accurate than static blocklist sources alone.
- Assess the quality of the decoder-LLM triage reports in terms of evidence grounding, semantic coherence, and analyst utility, and evaluate the overall latency and throughput characteristics of the full pipeline against a realistic SOC deployment baseline.

As this research evolved from an initial proposal stage to the current implementation, the original research questions from the thesis proposal are also recorded here for traceability, as they reflect the broader exploratory framing that motivated the subsequent design decisions:

- **RQ-P1:** Is it feasible to include a flow-based ML analysis and automated threat blocking in the IP reputation framework to improve accuracy and reduce false positives?
- **RQ-P2:** Does a local LLM or fine-tuned flow-based model operating on enriched per-IP flow sequences improve accuracy and be more feasible overall in terms of inline real-time threat blocking compared to existing ML-based analysis and threat blocking approaches?
- **RQ-P3:** What are the trade-offs to consider when introducing an LLM-based model in line with static IP reputation filters or ML-based flow analysis models?
- **RQ-P4:** How much better or worse would it be if a hybrid architecture—having IP reputation-based filtering as the first line of defence and LLM-based flow analysis for ambiguous logs—is introduced in the existing SOC analysis pipeline?

The initial objectives from the thesis proposal were similarly as follows:

- Include a minimal flow-analysis-based ML model in the framework for automated threat blocking and tune parameters to improve accuracy and reduce false positives.

- Fine-tune an open-source pre-trained LLM model with open datasets like CIC-IDS, UNSW-NB15, or UGR-16 flow data and build an LLM-based classifier to classify malicious or benign traffic based on flow data (Sharafaldin et al., 2018; Engelen et al., 2021).
- Compare results of LLM-based and ML-based models based on accuracy, false-positive rates, throughput, and latency.
- Propose and build a skeleton of a hybrid, multi-stage threat blocking pipeline having IP reputation as the first line of defence and LLM/ML-based flow analysis as the next-stage analyser for complex or ambiguous logs.

1.4 Dissertation Structure

The remainder of this dissertation is organized as follows. Chapter 2 presents the Literature Review, covering the theoretical foundations and existing work across the five areas the framework draws on: IP reputation systems and their limitations, behavioral flow-based IDS and the cross-domain generalization problem, BERT-based models for network traffic classification, log-based anomaly detection, and the use of LLMs in SOC pipelines. Chapter 3 describes the Methodological Approach, situating this research within a Design Science Research paradigm and explaining the design and evaluation strategy for each component of the framework. Chapter 4 presents the Work Done, covering the design, implementation, and experimental evaluation of the framework, including dataset construction, model training and fine-tuning, and DTS algorithm design. Chapter 5 presents the Results and Discussion, covering the full test evaluation of both the Tier-1 flow classifier and the Tier-2 log classifier, per-class analysis, baseline comparisons, and inference latency benchmarks. Chapter 6 concludes the dissertation with a summary of findings, answers to each research question, a candid discussion of the framework's limitations, and directions for future work.

2 Literature Review

This chapter presents the theoretical concepts necessary for the development of this project, along with the corresponding literature review.

The search strategy, database selection, inclusion and exclusion criteria, and iterative refinement process that governed this review are described in Chapter 3; the sections below organise the resulting literature into six thematic areas that map directly onto the design decisions and research questions of this thesis.

2.1 IP Reputation Systems and Architecture

IP reputation scoring and related methodologies have been used extensively as one of the quickest ways to filter malicious network traffic in the past decades. A simple IP reputation based threat blocking framework essentially is a method or an API which checks the IP address against one or multiple reputation lists or blocklist databases like AbuseIPDB, Shodan etc., that record threat associations such as spam, botnet, phishing, scanning, malware delivery etc., based on user reports, honeypot telemetry or other threat intelligence sources like Cisco Talos, Virustotal API etc. and transforms the result to a reputation score (Lewis et al., 2020; Xu et al., 2025; Miranda-García et al., 2023). This approach is operationally supportive of low-latency blocking decisions without the need of expensive deep packet inspection methodologies. However, IP reputation sources differ significantly by up to 80% in some cases among themselves, which leads to confusion and the likelihood of increase in false positives (Lewis et al., 2020). Furthermore, this static view of reputation scoring is limited and not really representative of the current state of the IP in the sense that IP addresses usually get reassigned, compromised hosts may later be cleaned, and attackers quickly move to new infrastructure.

Miranda-García et al. pursue this inconsistency problem directly, evaluating the *reliability index* of IP addresses appearing across multiple reputation lists and demonstrating that disagreement across sources is systematic rather than random (Miranda-García et al., 2023). Their analysis underscores that IP reputation sources reflect different data-collection methodologies, reporting frequencies, and threat-actor coverage windows, which means that an IP can simultaneously appear clean on one list and heavily flagged on another without either list being wrong in an absolute sense.

The landscape of existing IPR architectures reveals several distinct approaches to improving on the single-feed model, each addressing a different dimension of its fragility. One class of systems anchors reputation scoring in the MITRE ATT&CK framework, matching observed IP behaviour against documented tactics, techniques, and procedures (TTPs) to flag addresses exhibiting patterns characteristic of known threat actor playbooks rather than relying solely on prior report volume (Lasantha, Abeysekara, & Maduranga, 2024b). A particularly difficult evasion problem for any reputation-based system is the emergence of Residential IP Proxies (RESIPs), services that route malicious traffic through legitimate home-user IP addresses rather than data-centre ranges, deliberately exploiting the fact that ASN/range-based reputation rules almost universally whitelist residential address space because

blocking it en masse would produce catastrophic false-positive rates against normal users (Lasantha, Abeysekara, & Maduranga, 2024b). The BlackEye IPR framework addresses this class of evasion through a multistage ML pipeline that combines data cleansing, logistic regression, and random forest classification, reporting approximately 90% accuracy and a reduction in mean time to detect malicious IP activity of 27 days on average compared to conventional blocklist update cycles (Lasantha, Abeysekara, & Maduranga, 2024b). At larger scale, cross-protocol telemetry fusion approaches apply ML interpretability techniques so that analysts understand not just *that* an IP received a particular score but *which* features drove that classification, a requirement that is frequently cited in the literature as a prerequisite for SOC adoption. In terms of the AIPRA system specifically, Lewis et al. find that Apility.io outperforms AbuseIPDB as a standalone reputation source in terms of both precision and coverage, and that logistic regression outperforms random forest in the ML aggregation pipeline that combines multiple feeds into a single unified score (Lewis et al., 2020; Lasantha, Abeysekara, & Maduranga, 2024b).

Xu et al. provide the most thorough taxonomy of network node reputation evaluation methods, organising the field into three families with meaningfully different capability and limitation profiles (Xu et al., 2025). The *statistical family* encompasses feedback-based assessment (aggregating user and sensor interaction reports into a reputation score), statistics-based inference (detecting anomalies from historical node statistics), collaboration-based methods (building a global reputation picture through cross-domain data exchange), and pattern-based approaches (detecting specific threat patterns like DDoS group activity from coordinated traffic signatures); the critical limitation shared across this family is vulnerability to *slow-ingress attacks*, where an adversary mimics normal user behaviour at a low rate, gradually accumulating positive reputation before weaponising it, and statistical approaches have no reliable mechanism to distinguish this from genuinely benign activity. The *similarity-based family* exploits the empirical observation that attackers reuse infrastructure, including similar hosting providers, overlapping autonomous systems, and reused web page templates, and includes multi-source threat intelligence similarity pooling (comparing feature-by-feature across different TI feeds), network and geographical feature combinations such as the GeoNet dataset which merges traffic statistics with IP geolocation, and rendered content similarity for detecting reused phishing page templates; the key limitations here are dependence on the quality of known-malicious anchor samples for comparison, computational cost that scales poorly to large networks, and the requirement that similarity metrics be robust to noise in the feature representations. The *ML-based family* is the most heterogeneous and includes feature-engineering approaches that operate on static IP attributes, including URL length, WHOIS age, and passive DNS resolution frequency, as well as dynamic network-level relationships such as the domain-to-IP resolution graph (examples from the surveyed literature include SNARE, PreSTA, MalPortrait, and Notos, as cited in Xu et al.); deep learning and Graph Neural Network (GNN) approaches that learn end-to-end from raw network behaviour including handshake metadata, packet length sequences, and byte distributions without hand-crafted features; and hybrid models that combine traditional scoring with ML, such as CNN-based analysis of user link-following behaviour combined with random forest on correlated features, or

clustering-based discovery of unknown malicious domains similar to known-bad ones (Xu et al., 2025). The shared limitations of the ML family are the well-documented interpretability problem (black-box models are difficult for analysts to interrogate and audit), high data labelling cost, substantial computational overhead, and adversarial vulnerability, where attackers who understand the feature space can craft evasion inputs that cross decision boundaries without triggering detection (Xu et al., 2025).

To address this limitation, behaviour driven machine learning models have been introduced in later IP reputation systems which model reputation as a dynamic risk score that can be updated using both external threat intelligence and local telemetry data. Lasantha et al. proposed an AI-based framework for validating IP reputation in cloud and firewall environments, showing the reliability of ML-driven signature generation and Security Operations Centre (SOC) integration in terms of reputation based threat blocking (Lasantha, Abeysekara, & Maduranga, 2024b, 2024a; Lasantha, Abeysekara, Tilwari, et al., 2024).

On a similar idea of Dynamic Threat Source (DTS) score, IP SafeGuard presents significant improvement in the accuracy and precision by the proposed ensemble model which is a combination of Graph Convolutional Network (GCN) IP embeddings with the XGBoost ML training on extracted NetFlow/Whois/PassiveDNS based features (Siam et al., 2025). The DTS computation applies an LSTM-based reliability tracker to each source to compute adaptive weights, an exponential temporal decay factor with roughly 10% daily decay in relevance, and a 15% consensus bonus when two or more sources agree on a threat classification, normalising the resulting score to a $[0, 100]$ range for backward compatibility with existing SOC tooling (Siam et al., 2025).

The feature extraction pipeline underlying the IP SafeGuard model is worth examining in detail because it illustrates both the appeal and the risks of combining heterogeneous data sources into a single scoring system (Siam et al., 2025). On the input side, the system queries WHOIS for `domain_age` and `last_update`, passive DNS records for `dns_frequency`, and NetFlow for `packet_count` and `avg_packet_size`; missing values are handled by mean imputation, robust scaling is applied for normalisation, and SelectKBest feature selection is used to reduce dimensionality based on feature importance scores before the XGBoost classifier and GCN embedding model are trained (Siam et al., 2025). The hybrid XGBoost-plus-GCN design is genuinely interesting in terms of the way it combines tabular feature learning with graph-structured IP relationship modelling, and the reported accuracy figures are strong. However, the ML claim is only fully convincing if the training labels are independent of the DTS features used as inputs, otherwise the model is largely learning to reproduce a threat score rather than to predict threat reality, and the paper would require considerably stronger detail on timestamp alignment between external public threat-intelligence IP lists and internal NetFlow observations, as well as on dataset construction and label validity, before these two data streams could be treated as interchangeable inputs to the same model. This is a limitation I identified in reviewing the paper and one that the IP SafeGuard design motivated me to avoid in the construction of the DTS component of this thesis: the local honeypot and ML tier outputs used to update the DTS in this framework are generated internally, reducing the label-validity and

timestamp-alignment concerns that arise when mixing public threat-intelligence feeds with live network telemetry.

Xu et al. also identify several development trends that cut across all three families and that together describe the direction the field is moving in terms of architecture and methodology (Xu et al., 2025). The first is a shift from single-source to multi-source heterogeneous data fusion, driven by the consistent finding that no single feed or telemetry type is sufficient to cover the full threat landscape. The second is a movement from black-box decisions toward Explainable AI, as SOC adoption of ML-based reputation systems has been consistently hampered by the inability of analysts to audit or interrogate model outputs. The third is a transition from centralised processing to Federated Learning for massive-scale deployments, enabling reputation computation across distributed nodes without centralising raw telemetry. The fourth is a move from static defence to dynamic adversarial game framing, acknowledging that threat actors continuously adapt to reputation signals. The fifth is a push from isolated, proprietary evaluations toward standardised benchmarks that would allow fair cross-system comparison (Xu et al., 2025). These trends align directly with the design of the framework proposed in this thesis: the DTS aggregates local multi-source data from the honeypot, the ML tier, and cached external threat intelligence, addressing the first trend; the decoder LLM triage layer provides a natural language explanation of every score-update decision, addressing the second; the fully local deployment model means no raw telemetry leaves the SOC environment, addressing the data-sovereignty concern that centralised federated approaches still partially raise; and the dynamic DTS update cycle ensures that attacker adaptation is reflected in the blocklist without requiring manual analyst intervention.

Overall, the literature on IP reputation systems converges on three findings that directly motivate the design choices in this thesis. First, no single threat intelligence source is sufficient, and cross-feed aggregation with source reliability weighting is necessary. Second, static blocklists suffer from *reputation sink*, where scores become irrelevant over time without continuous real-time recalibration (Lasantha, Abeysekara, & Maduranga, 2024a). Third, the most effective architectures integrate reputation signals into the broader SOC workflow rather than treating them as a standalone perimeter filter.

2.2 Behavioral Flow Analysis and Intrusion Detection

The limitations of purely reputation-based filtering have driven a significant research pivot toward behavioural analysis of network traffic flows. Static reputation feeds suffer from problems like reputation decay over time, incomplete information, inconsistent across threat intelligence sources etc., and so can only be used as a first filtering layer in the threat blocking architecture of modern SOC systems (Miranda-García et al., 2023; Xu et al., 2025). Flow-based network traffic analysis addresses this gap as it provides a way to capture the behavior of the IP by examining how it interacts with the network rather than whether it has been reported as malicious or not in the past.

Network flows capture statistical properties like packet counts, byte counts, flow duration etc. which are helpful in detection of malicious behavior such as scanning, DDoS, botnet and multiple other attack types. These features can be extracted easily with tools like CICFlowMeter, Zeek, NetFlow etc. which makes flow analysis scalable and less

privacy invasive than payload based deep packet inspection methods (Miguel-Diez et al., 2026; Mondragon et al., 2025).

Miguel-Diez et al. provide a systematic literature review of unsupervised learning algorithms for anomalous traffic detection based on network flows, covering 63 scientific articles following the PRISMA guideline (Miguel-Diez et al., 2026). Their review establishes that the flow-based paradigm is now well-accepted as the primary scalable approach to IDS in large networks, motivated in part by the anticipated traffic volumes of 5G infrastructure where packet-level analysis becomes computationally unfeasible. The most widely used flow protocols in the reviewed literature are NetFlow (Cisco’s proprietary protocol), sFlow (a sampled-flow standard), and IPFIX (the IETF standardisation of NetFlow concepts) (Miguel-Diez et al., 2026). Among unsupervised methods, autoencoders emerge as the dominant algorithm, followed by SVM-based anomaly detection and Self-Organising Maps, reflecting the broader trend toward representation learning approaches that can detect novel attack patterns without requiring labelled training data (Miguel-Diez et al., 2026).

Mondragon et al. conduct a comparative study of datasets and machine learning algorithms for network flow-based intrusion detection, evaluating a range of algorithms including Random Forest, XGBoost, LightGBM, Decision Tree, and neural architectures across several benchmark datasets (Mondragon et al., 2025). Their findings confirm the general consensus in the field: tree-based ensemble methods achieve very high accuracy on individual datasets but generalise poorly across different network environments, while deep learning methods show more robust cross-environment performance at the cost of higher computational overhead and training complexity (Mondragon et al., 2025). Siam et al. explicitly integrate NetFlow-derived features (packet count, average packet size, connection duration) into their DTS computation alongside external threat intelligence, demonstrating that the combination substantially outperforms either signal alone (Siam et al., 2025). Therefore in this thesis, flow analysis is considered as a necessary behavioral layer for IP reputation scoring system.

2.3 IDS Datasets and the Cross-Domain Generalisation Problem

The quality and diversity of training data are arguably the most consequential and least acknowledged factors in determining the real-world deployability of a machine-learning-based IDS. The research community has produced a number of open-source benchmark datasets that have driven significant algorithmic progress, but these same datasets carry structural limitations that create a gap between reported laboratory performance and operational reality.

A lot of work has already been done in the field of ML models for flow-based Intrusion Detection Systems (IDS). Miguel-Diez et al. identify 16 usable datasets for network anomaly detection, observing that most of these contain flow data from synthetic network traffic because real-world data is difficult to empirically label as benign or malicious (Miguel-Diez et al., 2026). Similarly, Mondragon et al. benchmark 14 recent datasets for malicious traffic classification, which includes popular ones like CICIDS2017, and some combined datasets like NF-UQ-NIDS-

v2 (Mondragon et al., 2025).

But these datasets suffer from problems like high class imbalance, where benign traffic is usually around 80–90% of the data and some minority attack types are heavily underrepresented, which in some cases are as low as just 0.00001% of the data. This makes it challenging to train ML models because a model can easily miss rare attack types while learning to perform well on majority classes. Liu et al. demonstrate that standard class-imbalanced learning approaches, including oversampling, cost-sensitive loss weighting, and synthetic minority oversampling (SMOTE), are insufficient on their own for network intrusion detection because the minority attack classes are not just rare but also structurally heterogeneous across different network environments (Liu, Wang, Ruan, & Lin, 2022).

Another problem in terms of ML model performance trained on these datasets is cross-dataset generalisation. A model trained on one dataset may not perform well on another dataset because of differences in terms of network environment while capturing the flow, attack type labels, feature distribution, and differences in flow converters used etc. (Amin et al., 2026; Hu et al., 2025). The *cross-domain generalisation problem* refers to the well-documented phenomenon where a model trained on one network environment experiences substantial and sometimes catastrophic performance degradation when deployed on a different dataset with different attack distributions, background traffic characteristics, or feature extraction tooling (Amin et al., 2026). Amin et al. describe this as a consequence of distribution shift between training and deployment environments, and their Heterogeneous Deep Stacked Ensemble for IDS (HDSE-IDS) framework achieved an average F1-score of 88.77% across four NetFlow-standardised benchmark datasets, representing up to a 30% improvement over recent state-of-the-art methods on the same cross-domain evaluation protocol (Amin et al., 2026).

For this reason, combining datasets for better cross-dataset performance and using techniques like class weighting and undersampling for handling class imbalance have been proposed in the implementation of the ML model in this thesis. The practical implication is direct: training a Tier-1 flow classifier on a single dataset is insufficient for a SOC deployment intended to handle real-world traffic from a heterogeneous enterprise network, and the design choice to combine multiple CICIDS-family datasets and apply mixed-dataset training is grounded in this body of evidence, which consistently shows that dataset diversity during training is one of the most effective mitigations for the cross-domain generalisation problem (Amin et al., 2026; Liu et al., 2022).

2.4 Transformer-Based Models for Network Intrusion Detection

The dominant paradigm in flow-based IDS has historically been tree-based ensemble methods: Random Forest, XGBoost, LightGBM, and gradient boosting variants. Traditional tree-based models such as XGBoost, Random Forest etc. are still considered strong baselines for flow-based IDS because of their efficiency and effectiveness on structured tabular feature data. Based on these there are a lot of other models proposed in recent research, out of which researchers found that the Contrast Network Flow Classifier (CFC) model performed best overall in a comparative study of 7 state-of-the-art algorithms on 14 recent network flow datasets (Mondragon et al., 2025). They attributed

this to its use of Supervised Contrastive Learning (SupCon) and Dense ResNet block based architecture of the model which provides stability and improved class separation under imbalanced network intrusion settings (Liu et al., 2022).

However, most of these models or their derived variants depend heavily on manually engineered features and do not model sequential or semantic relationships across heterogeneous telemetry sources (Aghaei et al., 2025; Habibzadeh et al., 2025). And for this reason, combined with the popularity of transformer based models in recent years, the recent research focus has shifted towards transformer and language-model based approaches to address this limitation. Transformer-based models leverage self-attention mechanisms that learn to weight the relative importance of feature interactions dynamically, enabling them to capture complex, non-linear relationships and long-range dependencies in traffic sequences that tree models cannot represent efficiently (Hu et al., 2025). Dang showed the application of transformer techniques to intrusion detection while Davey et al. and Zhang et al. discussed the potential of large language models (LLMs) for network attack detection/classification (Dang, 2023; Davey, Karim, Hassen, & Batatia, 2025; Zhang et al., 2025).

BERTector demonstrated that joint-dataset learning combined with NSS tokenization and LoRA based fine-tuning can produce better results than the models discussed above with less than usual computational demands of transformer based models (Hu et al., 2025). Hu et al. propose BERTector's NSS-Tokenizer (*Network Security Semantics Tokenizer*), a dedicated tokenizer designed specifically for network traffic that reduces maximum token lengths by roughly 70% compared to the standard FullTokenizer, directly translating to faster inference and lower memory consumption (Hu et al., 2025). The BERTector-MIX configuration achieves F1-scores of 0.9887, 0.9610, and 0.9987 on KDD99, UNSW-NB15, and X-ILoTID respectively, with consistent cross-dataset performance that substantially exceeds models trained on single datasets, and robustness testing under adversarial distribution perturbations further confirms that BERTector degrades far less than classical ML and DL baselines under noise (Hu et al., 2025).

Habibzadeh et al. provide a comprehensive survey of LLMs applied to SOC workflows more broadly, identifying a consistent finding across the literature: LLMs significantly outperform classical approaches on tasks requiring semantic understanding of heterogeneous, semi-structured data, but they also document the persistent practical barriers to LLM deployment in real SOC environments, including high alert volumes, limited compute budgets, and the requirement for sub-second response times in inline detection scenarios (Habibzadeh et al., 2025). This observation directly motivates the tiered architecture proposed in this thesis, where a lightweight encoder-based classifier handles the high-volume inline flow classification task while the heavier decoder LLM is reserved for post-hoc triage of the subset of flows that reach Tier 2.

2.5 SecureBERT 2.0 and Log-Semantic Anomaly Detection

Another evolution which motivated this research is the BERT based models pre-trained on cybersecurity corpus like security logs, journals etc. which have shown to perform better than general purpose LLMs on cybersecurity tasks

(Aghaei et al., 2025; Habibzadeh et al., 2025). Small models with token-size limit of 512 and based on BERT like **CysecBERT** to bigger variants like **SecureBERT2.0** by Cisco with token size limit of 1024 and based on ModernBERT have been proposed in recent developments. The selection of a pre-trained language model as the backbone for both the Tier-1 flow classifier and the Tier-2 log classifier is a consequential design decision: the core argument for domain-specific pre-training is well established in the NLP literature, as models trained on general-purpose corpora develop representations that are poorly calibrated for specialised technical domains where the vocabulary distribution, entity types, and semantic relationships differ substantially from those in general text.

SecureBERT2.0 is particularly interesting for this research because it is a domain-adapted encoder-based language model pre-trained on a massive, multi-modal corpus of cybersecurity data including over 13 billion text tokens and 53 million code tokens and it performs great on semantic search, named entity recognition and threat analysis (Aghaei et al., 2025). SecureBERT 2.0, developed by Aghaei et al. at Cisco AI, builds on the ModernBERT architecture, which introduces extended attention mechanisms for long-context modelling, hierarchical encoding for understanding both fine-grained and high-level structures common in technical text and source code, and hybrid tokenization that handles both natural language and code (Aghaei et al., 2025). The pretraining corpus represents a thirteen-fold increase in data volume compared to the original SecureBERT, with the training process using a 1024-token sequence length and a microannealing curriculum that gradually adjusts sampling probabilities based on intermediate model perplexity (Aghaei et al., 2025).

Recent work on LogLLM and SaRLog support the value and performance of language models for log-based anomaly detection, especially when semantic context is required to distinguish normal operational events from suspicious security ones (Guan et al., 2025; Jilcha, Kim, & Kwak, 2024). System and application logs are semi-structured records of runtime events that contain substantial semantic information that flow-level analysis entirely misses: a network flow tells you that a connection lasted 3.2 seconds and transferred 47,000 bytes, while the corresponding system log might tell you that the process making that connection was `sshd`, that it triggered a PAM authentication failure, and that it subsequently accessed `/etc/shadow`. Jilcha et al. propose SaRLog (Semantic-aware Robust Log anomaly detection), which combines domain-specific BERT-based contextual embeddings with a Siamese network architecture and contrastive loss for few-shot generalisation, demonstrating F1-score improvements of up to 31.2% on the BGL dataset and 46.7% on the Thunderbird dataset compared to static-embedding baselines (Jilcha et al., 2024).

Guan et al. propose LogLLM, which combines a BERT encoder for extracting semantic vectors from individual log messages with a Llama decoder model for classifying log sequences, bridged by a projector that aligns the vector representation spaces of the two models (Guan et al., 2025). Unlike traditional methods that require log parsers to extract templates before analysis, LogLLM preprocesses log messages using regular expressions to replace variable parameters with a constant token `<*>`, thereby standardising the input without requiring parser training and making the approach more robust to parser failure on out-of-vocabulary log templates (Guan et al., 2025). LogLLM was

evaluated across four public log datasets (HDFS, BGL, Spirit, and Thunderbird) and outperformed all state-of-the-art baselines, including deep learning methods and prior LLM-based approaches (Guan et al., 2025). The architecture is directly relevant to the decoder LLM component of the Tier-2 design in this thesis, where a similar BERT-encoder-plus-decoder-LLM structure is used to produce evidence-grounded analyst triage reports from log evidence.

2.6 Knowledge Gap and Research Positioning

Having reviewed the existing literature across these five thematic areas, I can now articulate the specific combination of problems that the proposed framework addresses and that no existing work covers in an integrated way.

The IP reputation literature demonstrates that dynamic, multi-source threat scoring significantly outperforms static blacklists, and that integrating reputation signals into the SOC workflow rather than treating them as a standalone perimeter filter improves overall detection quality (Siam et al., 2025; Lasantha, Abeysekara, & Maduranga, 2024a). But the frameworks proposed in this literature, including IP SafeGuard, AIPRA, and the Lasantha et al. triple-validation system, are all cloud-connected architectures that depend on querying external threat intelligence platforms (AbuseIPDB, VirusTotal, Cisco Talos) in real time. This creates an immediate problem for organisations with data sovereignty requirements or GDPR obligations: sending connection metadata to an external cloud service for reputation scoring may itself constitute a personal data processing operation, and the network telemetry that accompanies each query can reveal sensitive information about internal users and systems. No existing IP reputation framework has been designed specifically for fully local deployment where all intelligence feeds are ingested and processed on-premises.

The flow-based IDS literature demonstrates that transformer-based models with domain-aware tokenization and joint multi-dataset training substantially outperform traditional tree-based methods on cross-domain generalisation (Hu et al., 2025; Liu et al., 2022; Amin et al., 2026). But these models are evaluated in isolation (as standalone classifiers on benchmark datasets) and are not integrated with an upstream reputation gating mechanism that would allow the system to avoid running full inference on every single flow from a known-clean or known-malicious IP. Running a fine-tuned BERT model on every flow in a high-volume enterprise network is computationally expensive and introduces latency that makes inline deployment impractical without GPU acceleration at the network edge. The combination of a fast reputation lookup as a first-pass filter with a transformer classifier reserved for uncertain cases is architecturally obvious in retrospect, but it has not been implemented and evaluated as a system in the existing literature.

The log anomaly detection literature demonstrates that log-level semantic analysis captures attack evidence that flow-level analysis misses, and that domain-specific BERT models substantially outperform general-purpose models for this task (Jilcha et al., 2024; Guan et al., 2025; Landauer, Onder, Skopik, & Wurzenberger, 2023). But the existing work focuses on anomaly detection as a binary classification problem (flagging log sequences as normal or anomalous) rather than on generating natural language triage reports that bridge the semantic gap

between raw log evidence and the actionable intelligence that a human analyst needs. The Habibzadeh et al. survey identifies this as an open problem: LLMs have demonstrated the capability to generate contextually relevant security analyses, but their integration into a two-stage pipeline where an encoder classifier first filters the log stream and a decoder LLM then synthesises a human-readable triage report has not been systematically implemented and evaluated (Habibzadeh et al., 2025).

The dataset literature makes clear that meaningful evaluation of a cross-domain flow classifier requires combined, standardised datasets rather than single-benchmark evaluation, and that log-based IDS evaluation requires datasets with realistic periodic user behaviour, attack variation, and reliable ground-truth labelling (Landauer, Skopik, et al., 2023; Sarhan, Layeghy, & Portmann, 2022; Amin et al., 2026). CICIDS-family datasets combined with standardised NetFlow feature extraction satisfy the first requirement, and AIT-LDSv2.0 satisfies the second, but no existing study has evaluated a system that uses both simultaneously, in an integrated two-tier architecture.

Taken together, the gap in the literature is the following: there is no existing framework that combines (1) a local, fully on-premises IP reputation gating layer with a dynamic threat score updated from local honeypot activity and locally-cached threat intelligence; (2) a transformer-based Tier-1 flow classifier fine-tuned on a combined CICIDS-family dataset for cross-domain robustness; (3) a transformer-based Tier-2 log classifier that applies log-semantic analysis to flows that pass Tier-1 gating, evaluated on a realistic multi-step attack log dataset; (4) a decoder LLM that synthesises a natural language triage report from the evidence produced by Tiers 1 and 2; and (5) a feedback loop where the model outputs update the DTS and blocklist, ensuring that the reputation layer continuously learns from the system's own detection decisions rather than depending exclusively on external feeds. This thesis proposes and evaluates exactly this combination, with the specific goal of demonstrating that it is deployable in a GDPR-compliant, locally-operated SOC environment without sacrificing the detection quality that the existing literature associates with cloud-connected, ML-augmented threat intelligence platforms.

Overall, the literature reviewed in this chapter establishes both the individual building blocks from which the proposed framework is constructed and the specific integration gap that this thesis aims to fill. Each design decision in the framework, from the choice of SecureBERT 2.0 as the backbone model, to the selection of AIT-LDSv2.0 for log classifier evaluation, to the DTS feedback architecture, is grounded in and justified by the evidence surveyed above.

3 Methodological Approach

In view of the identified research questions and the objectives of this work, this chapter describes and justifies the methodological choices made throughout the research. These choices span four interconnected areas: the overarching research paradigm that governs the relationship between design, implementation, and evaluation; the strategy used to conduct and iterate the literature review; the experimental methodology that shaped the model training and evaluation pipeline; and the criteria used to select the datasets on which the framework is built and tested.

3.1 Research Paradigm: Design Science Research

The overall research methodology adopted in this thesis is **Design Science Research** (DSR) (Kappen, 2019). DSR is a scientific problem-solving methodology developed specifically for the Information Systems (IS) domain, and many researchers have used the iterative DSR methodology to develop information technology (IT) artefacts. The core idea of DSR is that knowledge is produced not only by studying phenomena but by building artefacts (in this case, the IP reputation-enforced threat blocking framework) and then evaluating the gap between the desired behaviour and the actual behaviour of the artefact, reflecting on what that gap reveals about the problem, and redesigning accordingly. This design → build → evaluate → reflect → redesign cycle is the structural logic of the entire thesis.

DSR is particularly well-suited for this research because the goal is not pure empirical measurement of a pre-existing phenomenon but the construction of a novel framework that integrates components, including IP reputation scoring, behavioural flow classification, log-semantic refinement, and dynamic threat scoring, that exist separately in the literature but have not been assembled into a locally deployable, enforcement-coupled pipeline. A purely empirical methodology would be appropriate if the framework already existed and the task was to characterise its behaviour; DSR is appropriate here because the framework itself is the research contribution, and its design choices need to be argued, implemented, and then evaluated to confirm that the artefact addresses the identified knowledge gaps.

The DSR cycle maps directly onto the chapter structure of this dissertation. The literature review in Chapter 3 establishes the knowledge gap and provides the evidence base for every design decision made in the framework. The Work Done chapter documents the artefact itself and the experiments that constitute the first evaluation pass. The Results and Discussion chapter closes the evaluate-and-reflect phase by comparing the artefact's behaviour against the research questions and the claims made in the literature. Wherever the evaluation reveals underperformance or unresolved questions, those become starting points for the next design iteration, which is exactly how the experimental methodology evolved in practice, as described in Section 3.3 below.

3.2 Literature Review Methodology

The literature search was conducted across four academic databases, including *Google Scholar*, *Scopus*, *IEEE Xplore*, and *ACM Digital Library*, using primary search terms and combinations covering *IP reputation*, *dynamic threat scoring*, *intrusion detection system*, *network flow classification*, *BERT network intrusion detection*, *log anomaly detection*, *LLM Security Operations Center*, *cross-domain generalisation IDS*, and *SecureBERT*, with a strong preference for peer-reviewed work published between 2018 and 2026. But beyond the search mechanics, the more consequential methodological choice was the *order* in which the literature was read and the way experimental results drove targeted re-readings.

The review began with survey papers on IP reputation methods, including Xu et al., Miranda-García et al., and the broader body of work on static blocklist limitations, to identify the structural limitations of the current state of the art and confirm that the future research direction points toward behavioural signals rather than static reputation. This was followed by implementation papers on specific IP reputation frameworks, particularly the Lasantha et al. series on ML-driven signature generation and SOC integration and the IP SafeGuard framework, which provided concrete performance figures and design patterns for the Dynamic Threat Score component. Having established the motivation for moving beyond static reputation, I then moved into the behavioural signals literature: how AI, ML, and LLM-based approaches have been applied to IDS pipelines and SOC workflows more broadly, surveying which benchmark datasets exist and which model architectures had demonstrated the strongest cross-dataset generalisation performance. This is where I identified the well-documented cross-domain generalisation problem with single-dataset training (Amin et al., 2026; Engelen et al., 2021), which directly motivated the combined dataset construction strategy used in the experiments. Reading through the class imbalance literature (particularly Liu et al. on ConFlow and the ConFlow comparative study results) led to the discovery of Supervised Contrastive Learning (SupCon) as a newer paradigm for handling the extreme class imbalance that characterises IDS datasets, where minority attack classes can represent as little as 0.00001% of samples. Following this, the review expanded to LLM-in-SOC literature, tiered model architectures, log dataset quality papers (which led to the selection of AIT-LDS v2.1 based on Landauer et al.’s systematic approach to maintainable, ground-truth-labelled log datasets (Landauer, Skopik, et al., 2023)) and tokenization method research, particularly the NSS-Tokenizer approach from BERTector, which informed how flow records are serialised as `FeatureName=value` token sequences before being passed to SecureBERT 2.0.

The critical point is that the literature review was not a one-pass survey completed before the experiments began. It was a living, iterative process: experimental results sent me back to the literature for targeted reading, and targeted reading motivated new experiments. A low macro-F1 on a particular attack class prompted a search for papers on why that class might be difficult to learn; a paper on asymmetric loss functions motivated a change to the loss configuration; a paper on cross-domain generalisation motivated a change to the dataset construction strategy. This

back-and-forth between reading and experimentation is characteristic of DSR, where the artefact and the knowledge base develop together rather than sequentially.

3.3 Experimental Design and Implementation Methodology

The experimental approach followed an iterative refinement cycle that is best described as: implement → train → evaluate macro-F1 and per-class F1 → identify which labels are failing and why → research the failure mode → redesign. This section documents the key methodological decisions that arose from that cycle.

3.3.1 Loss Function Experimentation

The first and most consequential axis of experimentation was the choice of loss function, because IDS datasets are extremely imbalanced and the loss function determines how the model weights the gradient signal from rare positive-class samples against the overwhelming mass of true-negative samples.

Binary Cross-Entropy (BCE) was the baseline loss for the multi-label classification task used in the log classifier. Standard BCE computes the cross-entropy loss independently for each label and averages across labels, which means that on a dataset where a label like `dns_exfil` appears in fewer than 0.01% of samples, the gradient contribution from true positives is so small relative to the flood of true negatives that the model learns to predict that label as always-absent, since this minimises total loss while sacrificing recall on the rare class entirely.

Multi-class Focal Loss (used in the early flow classification experiments) was a natural extension of cross-entropy that applies a modulating factor $(1 - p_t)^\gamma$ to each sample’s loss, downweighting the contribution of easy, confidently-classified samples and concentrating the gradient signal on difficult examples. With $\gamma = 2.0$, focal loss helped the model focus on the harder minority classes. However, focal loss applies the same modulation to both easy positives and easy negatives, which is a problem in extreme multi-label settings where the negatives vastly outnumber the positives: even a mildly downweighted mass of trivially-correct negatives still drowns out the already-rare positive signal.

Asymmetric Loss (ASL), introduced by Ridnik et al. at ICCV 2021, was the final choice for the log classifier because it addresses this asymmetry directly by treating the positive and negative branches of the loss separately. The negative branch applies both a hard clipping step, zeroing out gradients for any negative sample where the model’s predicted probability falls below a margin m^- (set to `clip=0.05`), and an additional downweighting factor $p_m^{\gamma^-}$ where $\gamma^- = 4$, suppressing easy negatives aggressively. The positive branch, by contrast, applies no downweighting at all ($\gamma^+ = 0$), preserving every gradient signal from the already-rare true positives. The result is a loss function that systematically redirects training capacity from the trivially-correct negatives toward the hard, informative minority-class positives, which is exactly the right trade-off for an IDS log dataset where rare attack steps like `dns_exfil` are precisely the labels that matter most for analyst triage. (Ridnik et al., 2021)

3.3.2 Training Infrastructure and Kaggle Constraints

All training was conducted on Kaggle compute resources: the flow classification experiments used two NVIDIA T4 GPUs (16 GB VRAM each) run via `torchrun` with `--nproc_per_node=2` for data-parallel training, while the log classifier was trained on a Kaggle TPU v3-8/v5e-8 instance (8 TPU cores) using `torch_xla` for XLA-based distribution. Kaggle imposes a 12-hour session limit and approximately 30 hours of GPU/TPU time per week. This constraint was not a minor inconvenience, and it directly shaped the entire experimental methodology, because it was not possible to run unbounded grid searches or extended ablations. Every training run had to be thoughtfully configured before launch, with all hyperparameters justified in advance rather than discovered through exhaustive search, and all models were configured for checkpoint-based training so that progress could be resumed across Kaggle sessions without losing the accumulated training state.

The key training parameters that were deliberated and tuned are described below.

Per-device batch size is the number of samples processed by each GPU or TPU core in a single forward and backward pass. Larger batch sizes use more VRAM but produce more stable gradient estimates and better utilisation of tensor cores. For the flow classifier on T4 GPUs, a per-device batch size of 32 was used; for the log classifier on TPU cores, 32 per core with 8 cores active gave an effective batch of 256.

Gradient accumulation steps simulate a larger effective batch size without the VRAM cost of a single large batch, by accumulating gradients across multiple micro-batches before each optimizer step. The effective batch is calculated as $\text{per_device_batch} \times \text{num_gpus} \times \text{accumulation_steps}$. For the T4 runs, gradient accumulation steps of 2 gave an effective batch of $32 \times 2 \times 2 = 128$, which was sufficient for stable training while staying within VRAM limits.

Gradient checkpointing trades compute for memory by not storing all intermediate layer activations during the forward pass, which at a sequence length of 512 tokens across SecureBERT 2.0's 22 ModernBERT layers can reach 12–15 GB of VRAM, and instead recomputing the intermediate activations on demand during the backward pass from stored layer-boundary tensors. This was required in several T4 configurations to stay within the 16 GB per-device VRAM limit, at the cost of approximately 30% additional compute per step. On TPU, gradient checkpointing was disabled because the XLA compilation model handles memory more efficiently.

Training checkpoints save the full model weights and optimizer state at regular step intervals, enabling training to resume across Kaggle sessions without losing progress. Given the 12-hour session limit, checkpoints were configured at every 600–3000 steps depending on the estimated step time and the total training horizon, with a rolling limit of 2–3 saved checkpoints to manage disk usage.

Mixed precision training was applied throughout: fp16 on T4 GPUs (using NVIDIA tensor core acceleration for half-precision arithmetic) and bfloat16 on TPU (the native TPU numerical format, which is more numerically stable than fp16 for training because its larger exponent range avoids overflow).

3.3.3 Evaluation Metric and Threshold Methodology

Macro-F1 was chosen as the primary evaluation metric because it gives equal weight to every class regardless of support, meaning that a rare attack label like `dns_exfil` contributes the same amount to the final score as a common label like `recon_scan`. This is the appropriate metric for an imbalanced multi-label security classification task precisely because the rare labels are the ones that matter most for analyst triage: a system that achieves 99% weighted-F1 by perfectly classifying the dominant benign and common-attack classes while completely failing on rare attack steps is operationally useless.

For the log classifier, per-class decision thresholds were tuned on the validation set rather than using a fixed 0.5 cutoff. The reason is that ASL shifts the model's output probability distribution in ways that make 0.5 an arbitrary and often inappropriate threshold for rare classes, since the model may assign a well-calibrated but sub-0.5 probability to a rare positive that is genuinely meaningful. The threshold sweep ranged from 0.10 to 0.90 in steps of 0.05, selecting the threshold that maximised per-class F1 on the validation set.

The **shaw** scenario was used as the held-out cross-scenario test partition. Its distributional properties (documented in Table in Chapter 4) make it structurally dissimilar from all seven training scenarios in terms of the dominant attack activity volumes, which means a model evaluated on shaw cannot rely on volume-driven co-occurrence statistics memorised from training and must rely instead on genuinely transferable representations of the individual attack steps.

3.3.4 Cross-Dataset Training and Label Handling

For the flow classification experiments, when per-class F1 results identified which attack labels were failing, the response was to research why, which led back to the cross-domain generalisation literature discussed in Chapter 3. The decision to combine multiple CICIDS-family datasets (specifically the Distrinet reproductions of CIC-IDS 2017 and CSECIC-IDS 2018) for joint training was grounded in the consistent finding across Amin et al. and related work that dataset diversity during training is one of the most effective mitigations for cross-domain generalisation failure (Amin et al., 2026). The Distrinet reproductions were selected over the raw CICIDS releases specifically because Engelen et al. had identified serious data quality issues in the originals, including traffic leakage, near-duplicate feature artefacts, and labelling inconsistencies, that cause artificially inflated accuracy when models are evaluated on the raw dataset as-is (Engelen et al., 2021). Low-represented attack labels in each individual dataset were identified during exploratory data analysis, and merging strategies were evaluated to increase the effective positive-sample count per optimizer step for those labels, with class weighting applied to ensure that the remaining imbalance does not dominate the training signal even after dataset combination.

a) Label Normalisation and Conflict Resolution

The Distrinet reproductions of both CICIDS datasets include an `Attempted Category` column and a corresponding family of “attempted” label variants, including `DoS Hulk - Attempted`, `FTP-BruteForce - Attempted`, and so on, that record flows where an attack was attempted but not necessarily completed. I merged these back into their parent attack labels rather than treating them as a separate class, because attempted flows are security-relevant and represent the same attack family in terms of the network behaviour and the packet-level signatures that an IDS classifier should learn to detect; an FTP brute-force attempt is still an FTP brute-force event regardless of whether the authentication ultimately succeeded. The `Attempted Category` column itself was then dropped entirely, because it is label metadata rather than a network-flow feature, and retaining it in the feature space would introduce label leakage, since a non-null value in that column directly discloses the attack family to the model without requiring it to learn anything from the flow statistics.

Beyond the attempted-variant merges, three further consolidations were made to address class sparsity. Web Attack subtypes, including `Web Attack - Brute Force`, `Web Attack - XSS`, `Web Attack - SQL`, and `Web Attack - SQL Injection`, were merged into a single `Web Attack` class because the individual sub-classes were extremely small and would produce unstable per-class gradient estimates that tend to either oscillate or collapse to zero recall during training. Portscan variants (`Portscan`, `PortScan`, `Infiltration - NMAP Portscan`, `Infiltration - Portscan`) were merged into a single `Portscan` label because they represent the same port-scanning behaviour regardless of whether the scan occurred as part of a broader infiltration scenario or as a standalone reconnaissance event. The remaining Infiltration sub-scenarios (`Infiltration - Communication Victim Attacker` and `Infiltration - Dropbox Download`) were likewise consolidated into a single `Infiltration` class. Brute-force naming variants across the two datasets were also standardised: the FTP family (`FTP-Patator`, `FTP-BruteForce`, `FTP-Brute Force`) was mapped to `FTP-BruteForce`, and the SSH family (`SSH-Patator`, `SSH-BruteForce`, `SSH-Brute Force`) was mapped to `SSH-BruteForce`.

After all label merges were applied, duplicate feature rows were removed, but rows with identical feature vectors and different labels were treated as conflicts rather than silently resolved by majority vote or random selection. The inclusion of destination port (`Dst Port`) in the feature set turned out to be the critical discriminating factor here: with `Dst Port` retained, only 17 BENIGN/Portscan conflicts remained across 11,276,766 samples, which I dropped cleanly without material loss. Without `Dst Port`, the conflict count rose to 183, which confirmed that destination port is a meaningful service-targeting discriminator (a Portscan and a benign connection to the same destination IP on different ports carry genuinely different security implications) and should be retained in the feature set. Source port (`Src Port`), by contrast, is ephemeral and not a reliable discriminator, so it was excluded from the feature set to avoid the model overfitting to assignment patterns that do not generalise across environments.

b) Why NF-UQ-NIDS-v2 Was Not Used

NF-UQ-NIDS-v2 (Sarhan et al., 2022) is a unified multi-dataset benchmark that combines UNSW-NB15, BoT-IoT, ToN-IoT, and CSE-CIC-IDS2018 into a common 43-feature NetFlow v9 schema, and it was designed specifically to support fairer cross-dataset generalisation evaluation, which made it an obvious candidate for this work. I investigated it and ultimately decided not to use it for the primary combined dataset, for two reasons.

The first is feature-schema incompatibility. NF-UQ-NIDS-v2 uses the standardised 43-feature NetFlow v9 representation, which is structurally different from the CICFlowMeter-derived feature set used across the CICIDS family. SecureBERT 2.0's tokenization strategy serialises flow records as `FeatureName=value` token sequences, and the feature names themselves carry semantic meaning that the model learns from its cybersecurity pre-training corpus, and `Bwd Packet Length Max` and `Fwd IAT Std` are semantically distinct tokens from NetFlow v9 field names like `L4_SRC_PORT` and `L7_PROTO`. Jointly tokenizing CICFlowMeter feature names alongside NetFlow v9 field names within the same vocabulary, without a careful bridging strategy, would create a tokenization mismatch that could undermine the pre-trained representations by associating network concepts with feature-name tokens that the model's cybersecurity pre-training did not use in that context. A separate training run on NF-UQ-NIDS-v2 alone would be valid but would not contribute to the same combined-CIC model evaluated in this thesis.

The second reason was empirically discovered during preprocessing. Dropping `L4_SRC_PORT` before deduplication (which appears reasonable since source ports are often ephemeral and can act as unintentional flow identifiers) caused catastrophic loss of attack-class samples. The original 75,987,976-row dataset had 62,670,508 unique rows with `L4_SRC_PORT` retained, but only 13,533,845 without it: a collapse of approximately 49.1 million rows. This loss was extremely unevenly distributed across classes. Bot lost 98.31% of its samples, DDoS lost 97.68%, DoS lost 95.24%, Backdoor lost 95.00%, and Reconnaissance lost 91.90%. The class distribution shifted from approximately Benign 33%, DDoS 27.6%, DoS 23.6% to Benign 77.57%, DDoS 2.96%, DoS 5.20%, transforming a reasonably balanced multi-class problem into a heavily benign-dominated dataset where the most important attack classes had almost no representation left. The conclusion I drew from this is that `L4_SRC_PORT` is not a simple redundant identifier in this preprocessed version of the data; it preserves flow-level variability, especially for high-volume attack classes like DoS and DDoS whose flows are often highly repetitive when the source port signal is removed. But retaining it raises the concern that a model trained on this data might overfit to source-port patterns rather than learning generalisable attack semantics, a concern that did not arise with the CICFlowMeter-based feature set, where the analogous `Src Port` exclusion had no comparable effect on class distribution.

For these two reasons, namely feature-space incompatibility with the CICFlowMeter-based tokenization strategy and the fragility of the class distribution to source-port preprocessing decisions, NF-UQ-NIDS-v2 was not used for the primary training data. It remains a strong candidate for future cross-dataset evaluation once the framework is

extended to support the NetFlow v9 feature schema directly.

3.3.5 Iterative Refinement Cycle

The overall experimental approach was not a single train-evaluate-report pipeline but a research-experiment feedback loop: each set of results raised specific questions (why is macro-F1 low on class X? is this a data issue or a model capacity issue? does the model fail because it has never seen enough positive examples of this class, or because the class is inherently ambiguous in the feature representation?) that sent me back to targeted literature reading, which then motivated a new experiment. The progression from standard cross-entropy to focal loss to asymmetric loss was driven by this loop, as was the move from single-dataset to combined-dataset training, the investigation of SupCon as a complementary training objective (Liu et al., 2022), and the exploration of quantile binning for numerical features as a tokenization improvement. This iterative cycle is the methodological core of the research and is explicitly consistent with the DSR paradigm described in Section 1: the artefact is refined through repeated evaluate-reflect-redesign iterations rather than delivered as a single-pass implementation.

3.4 Feature Selection: Mutual Information and Conditional Entropy Pruning

Before tokenizing the combined CIC-IDS dataset for SecureBERT 2.0 fine-tuning, I ran a seven-step feature selection pipeline designed to remove uninformative features, eliminate redundant ones, and verify that the retained set contains genuinely diverse discriminative signals. Starting from the 85 columns remaining after stripping metadata columns (id, Flow ID, Src IP, Dst IP, Timestamp), the pipeline proceeded as follows.

Step 1: Constant column removal. Bwd URG Flags was the only constant column across all 11,276,766 samples (a single unique value throughout), and it was dropped immediately because a feature with zero variance carries no discriminative information regardless of the label distribution. This left 84 columns.

Step 2: Identical column removal. A hash-then-verify pass identified four pairs of exact duplicates: Fwd Segment Size Avg was identical to Fwd Packet Length Mean; Bwd Segment Size Avg to Bwd Packet Length Mean; URG Flag Count to Fwd URG Flags; and Average Packet Size to Packet Length Mean. The lower-priority duplicate in each pair was dropped, leaving 80 columns and 79 numeric candidates for the MI-based steps (the label column was excluded from scoring).

Step 3: Target-guided binning. To enable mutual information estimation on continuous features without losing label-relevant structure, each continuous feature was discretised into target-aware bins using a Decision Tree classifier fitted per feature against the attack label (entropy criterion, maximum 32 leaf nodes, minimum 1% samples per leaf). This produces bins whose boundaries are chosen to maximally separate classes rather than to uniformly cover the value range, which means MI computed on these bins reflects the feature's actual class-discriminative

power rather than its raw distributional variation.

Step 4: Mutual Information scoring. Mutual information between each feature’s binned values and the class label was computed as:

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log_2 \frac{p(x, y)}{p(x) p(y)} \quad (3.1)$$

where Y is the attack label and X is the discretised feature. Eight features scored $MI = 0.0$, including Fwd URG Flags, Fwd Bytes/Bulk Avg, Fwd Packet/Bulk Avg, Fwd Bulk Rate Avg, Subflow Fwd Packets, Subflow Bwd Packets, ICMP Code, and ICMP Type, and were eliminated at this stage, since $MI = 0.0$ means a feature carries no class-discriminative information on this dataset. This left 71 features.

Step 5: Conditional Entropy pruning. Features were processed from highest to lowest MI score. For each candidate feature C , if there already existed a retained feature K (with the highest MI among retained features) such that the reduction in label uncertainty from adding C given K fell below a threshold $\varepsilon = 0.005$ bits, then C was dropped as redundant:

$$H(Y | K) - H(Y | K, C) < \varepsilon, \quad \varepsilon = 0.005 \text{ bits} \quad (3.2)$$

where the conditional entropy is defined as:

$$H(Y | X) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log_2 p(y | x) \quad (3.3)$$

This step eliminated 17 features and left 62. The most striking drop was Packet Length Std ($MI = 0.8156$), which was dropped as redundant with Packet Length Variance (entropy delta $\approx 2.37 \times 10^{-8}$), essentially zero new information, since standard deviation and variance are monotone transforms of each other and carry identical discriminative signal. Other drops caught near-duplicate flag and min/max pairs: CWR Flag Count was redundant with ECE Flag Count (entropy delta $\approx 8.0 \times 10^{-4}$), and Packet Length Min was redundant with Packet Length Max given that the maximum was already retained (entropy delta $\approx 3.7 \times 10^{-3}$).

Step 6: Diff-safe correlation check. Among feature pairs with Pearson correlation > 0.95 , the lower-MI feature was dropped only if the rows where the two features actually differed from each other were predominantly benign, specifically if the attack ratio on those differing rows was below 0.5. This condition ensures that highly-correlated features are not dropped when their divergence from each other specifically marks attack traffic, even if the features are otherwise nearly interchangeable on benign flows. For example, Bwd Packet Length Std was dropped in favour of Bwd Packet Length Max because the 10.4 million rows where they differed had an attack ratio of only 35.7%, below the 50% retention threshold, indicating their divergence is more characteristic of benign variation than attack-specific behaviour. This step eliminated 12 more features.

Step 7: Duplicate row removal. Six duplicate feature rows were identified and dropped; all six were BENIGN samples, representing 0.00008% of the BENIGN class. No attack samples were lost at this stage.

The final result was **50 features** selected from 79 numeric candidates, with the full dataset reduced from 11,276,766 rows to a clean, deduplicated corpus ready for tokenization. The top-10 features by MI score are dominated by packet-length and payload-size features, including `Bwd Packet Length Max` ($MI = 0.848$), `Packet Length Max` (0.844), and `Total Length of Bwd Packet` (0.839), which reflects that the volume of data transferred per packet is the most discriminative signal for attack type in CICFlowMeter flows, with backward-direction packet size being particularly informative because attack responses (e.g. exfiltration payloads, DDoS amplification replies) tend to produce characteristically large or small backward payloads depending on the attack family. Timing and inter-arrival features (`Bwd IAT Total`, `Bwd IAT Mean`, `Bwd IAT Std`) occupy a middle tier with MI in the 0.62–0.63 range, and flag-count features (`ACK Flag Count`, `FIN Flag Count`) form the lower tier but were retained because they contribute signal that is genuinely independent of the packet-length features, as confirmed by the conditional entropy pruning step.

3.5 Dataset Selection Methodology

Dataset selection was treated as a deliberate methodological choice rather than a default to the most commonly used benchmarks. Separate criteria were applied to the flow dataset and the log dataset, reflecting the different evaluation requirements of the two classification tiers.

Flow datasets. The selection criteria were dataset diversity in terms of attack type coverage, availability of quality-controlled Distrinet reproductions, and compatibility with the feature-name tokenization strategy used by SecureBERT 2.0. The CIC-IDS 2017 dataset, introduced by Sharafaldin et al. (Sharafaldin et al., 2018), covers a wide range of attack types including DoS, DDoS, brute force, web attacks, infiltration, port scans, and botnet traffic, and is the most widely benchmarked dataset in the flow-based IDS literature. The CSECIC-IDS 2018 dataset adds additional attack scenarios and provides complementary coverage, particularly for attack types underrepresented in the 2017 release. Both were used in their Distrinet-reproduced forms rather than the original releases to avoid the labelling inconsistencies and data quality problems documented by Engelen et al. (Engelen et al., 2021), which would otherwise invalidate cross-dataset comparisons and inflate reported accuracy figures.

Log dataset. The log dataset selection was evaluated against a checklist of requirements: use-case specificity for SOC scenarios (the dataset should contain realistic system log data generated under genuine attack conditions, not synthetic or simplified traffic), reliable ground-truth labelling (each log line should have a verifiable label that reflects actual attacker activity rather than heuristic annotation), attack variation sufficient to cover multiple SOC-relevant attack steps, and full reproducibility. The AIT Log Data Set v2.1 (AIT-LDS v2.1), described by Landauer et al. (Landauer, Skopik, et al., 2023), was selected as the log dataset because it satisfies all four criteria. It provides maintainable, labelled system log data generated from a model-driven testbed that simulates

realistic enterprise infrastructure, with attack steps covering foothold establishment, privilege escalation, lateral movement, reconnaissance, DNS exfiltration, and web-based attacks, which are labels that map directly onto the analyst triage scenarios the Tier 2 classifier is designed to address. The dataset's maintainability, meaning the ability to regenerate it with updated attack tooling while preserving label structure, was also a practical consideration: it reduces the risk that the evaluation dataset becomes stale relative to the attack techniques the framework is expected to detect in production.

Overall, the methodological choices described in this chapter reflect a consistent underlying logic: every decision, from the choice of DSR as the research paradigm to the iterative literature review strategy, the loss function experimentation, and the dataset selection criteria, was driven by the operational requirements of the problem rather than by defaults or convention. The goal throughout was to build and evaluate a framework that is genuinely deployable in a real SOC environment, and the methodology was shaped accordingly.

4 Work Done

This chapter describes the design and implementation work carried out for this thesis, covering the full framework architecture, the components that have been built and validated at the time of writing, and the components that are proposed and argued from the literature but not yet implemented. The implemented work spans three areas: a distributed local honeypot infrastructure, two custom dataset construction and preprocessing pipelines (one for network flow classification and one for log-based classification), and a progressive series of flow classification experiments that constitute the empirical foundation for the Tier 1 flow filter. The Dynamic Threat Score (DTS) generator and the decoder LLM triage component remain at the design and feasibility-argumentation stage.

4.1 Framework Architecture

The proposed framework is a locally deployable, IP reputation-enforced threat blocking system that combines three layers of defence: a first-line reputation gate operated by a central orchestrator, a flow-based ML filter for new or unknown IPs, and a log-semantic refinement stage for suspicious flows that warrant deeper analysis. Figure 1 illustrates the complete data flow and component interactions.

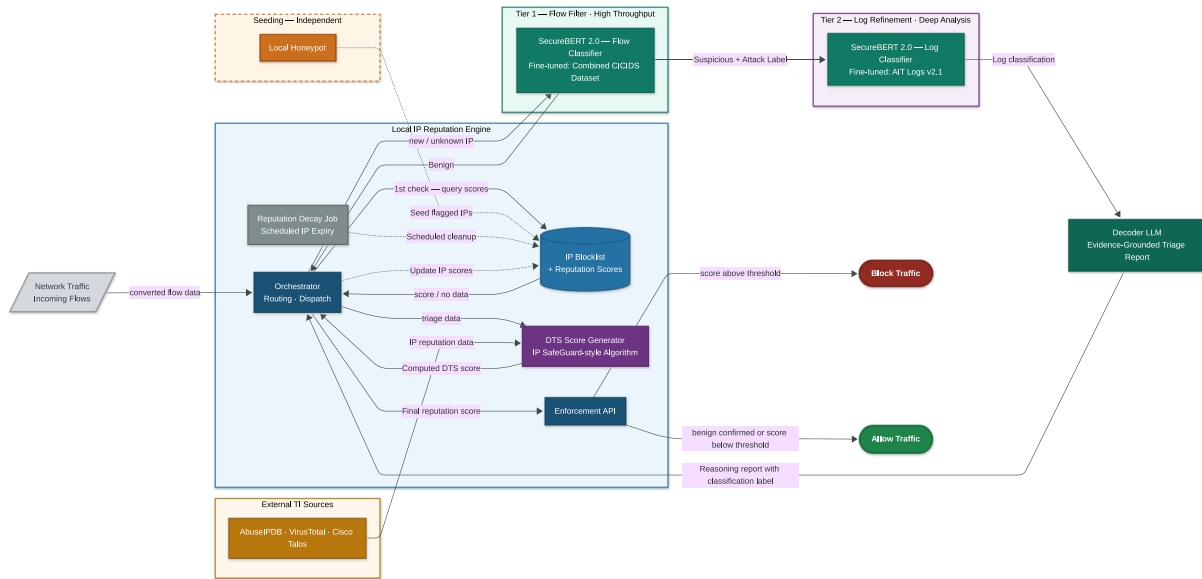


Figure 1: The proposed IP reputation-enforced threat blocking framework. The Local IP Reputation Engine acts as the central orchestrator: it gates every incoming flow against the IP blocklist, dispatches unknown IPs to the two-tier AI pipeline, and consolidates the returned scores into a single enforcement decision.

At the centre of the framework sits a local **orchestrator** which owns the IP reputation engine and decides, for every incoming flow, whether a model needs to run at all. Incoming network traffic is first converted into flow records and handed to the orchestrator, which performs a **Tier 0** lookup against the local IP blocklist before anything else, so

an IP that already carries a bad reputation score is blocked straight away through the enforcement API at near-zero cost and never reaches the models, which is what keeps this architecture IP-enforced rather than a model that runs on every flow. Only when the orchestrator finds no usable score for a new or unknown IP does it dispatch the flow to **Tier 1**, the high-throughput flow filter built on SecureBERT 2.0 fine-tuned on the combined CICIDS dataset, which either returns the flow as benign so the orchestrator can allow it, or flags it as suspicious together with an attack label and forwards it to **Tier 2** for deeper analysis (Aghaei et al., 2025). Tier 2 is a log-semantic refinement stage where a second SecureBERT 2.0 model, fine-tuned on the AIT log dataset v2.1, re-examines the logs behind the suspicious flow and classifies them, after which a decoder LLM turns that classification into an evidence-grounded reasoning report that justifies the block in terms an analyst can act on. The AIT log dataset is suitable for this stage because it provides maintainable, labelled logs generated from realistic infrastructure behaviour, making it one of the more practical open-source resources for log-based IDS evaluation (Landauer, Skopik, et al., 2023).

The outputs of the two tiers do not enforce on their own but are passed by the orchestrator to a **Dynamic Threat Score (DTS)** generator modelled on the IP SafeGuard framework, which queries external threat intelligence sources like AbuseIPDB, VirusTotal, Cisco Talos etc. and combines that reputation data with the local model evidence into a single risk score, which the orchestrator then writes back to the blocklist so that the same source is later handled by the cheap Tier 0 lookup instead of the full pipeline (Siam et al., 2025). The enforcement API makes the final decision based on this consolidated reputation score, blocking the traffic when the score is above the configured threshold and allowing it when the flow is confirmed benign or the score stays below it. Two supporting components keep the blocklist current without sitting on the live traffic path: a local honeypot runs independently and seeds every IP that interacts with it into the blocklist, while a scheduled reputation decay job expires stale entries so that reassigned addresses and cleaned hosts do not keep getting blocked on outdated evidence, which directly addresses the reputation decay and false positive problems of static blocklists. Overall this layered design keeps the low-latency operational benefit of reputation-based filtering while reducing the dependence on static blocklists and improving adaptability through the behavioural and semantic evidence supplied by the two-tier models.

4.2 Design Justification

The proposed two-tier architecture departs from conventional anomaly-first intrusion detection pipelines by employing supervised, transformer-based classification at both flow and log levels. This design decision is motivated by both operational constraints in Security Operations Centers (SOCs) and limitations identified in existing anomaly detection approaches.

4.2.1 Limitations of Unsupervised Anomaly Detection as a First Stage

Unsupervised anomaly detection is frequently used in early-stage intrusion detection due to its ability to identify unknown or zero-day attacks without labeled data (Miguel-Diez et al., 2026). However, this comes with inherent drawbacks. These systems primarily identify deviations from normal behavior without assigning semantic meaning or attack categories to the detected events. As a result, anomaly detection outputs require additional interpretation and post-processing, which increases analyst burden.

Furthermore, SOC environments already face significant challenges related to alert volume and false positives. High volumes of anomalous alerts can lead to alert fatigue and reduce the effectiveness of analysts in identifying genuine threats (Habibzadeh et al., 2025). In such contexts, using an anomaly detection system as the first filter would amplify noise rather than reduce it.

4.2.2 Justification for Supervised Flow-Based Classification

Instead of forwarding all anomalous events downstream, the proposed architecture employs a supervised, fine-tuned SecureBERT model as the first-stage filter. This enables direct classification of network flows into predefined attack categories, ensuring that only semantically meaningful suspicious traffic progresses to the next stage.

Recent studies demonstrate that LLM-based intrusion detection systems improve detection reliability and reduce false positive rates compared to traditional methods (Davey et al., 2025). Additionally, transformer-based models can capture complex traffic patterns and contextual relationships, enabling more precise classification of malicious activities (Hu et al., 2025).

This aligns with the observation that machine learning-based IDS is transitioning from anomaly-based detection toward more structured classification approaches that can better handle diverse and evolving attack patterns (Mohamed, 2025).

4.2.3 Complementarity of Flow-Level and Log-Level Analysis

A key design decision is the use of two independent classification stages operating on different data modalities:

- The first stage captures **network behavior** using flow-based features.
- The second stage captures **system-level semantics** using log data.

Flow-based analysis provides scalable detection across large volumes of network traffic, while log-based analysis provides contextual information about system behavior and application-level events. This complementary relationship enables more robust detection compared to single-source approaches.

This design reflects SOC practices, where alerts are correlated across multiple sources, including network traffic, logs, and threat intelligence feeds, to validate incidents and reduce false positives (Habibzadeh et al., 2025).

4.2.4 Cross-Validation and Explainability

The two-tier architecture enables cross-validation between independent models. If both stages converge on the same classification label, the system can assign higher confidence to the detection. Conversely, disagreements between stages act as explicit conflict signals that guide analyst investigation.

Unlike anomaly detection systems, which provide limited interpretability, supervised classification models produce explicit labels (e.g., DoS, brute-force, web attack). This improves explainability and aligns with SOC workflows, where alert triage and validation require interpretable outputs.

Additionally, LLM-based approaches have demonstrated the ability to provide contextual explanations for detected events, further enhancing analyst understanding and decision-making (Habibzadeh et al., 2025).

4.2.5 Operational Resilience and Business Continuity

In production environments, model failures, data drift, or deployment issues are inevitable. The proposed architecture ensures that each stage can function independently as a standalone detection system. This design avoids single points of failure and supports continuous operation even under partial system degradation.

Such redundancy is particularly important in real-world SOC deployments, where uninterrupted monitoring is required to maintain security posture.

4.2.6 Alignment with SOC Detection and Analysis Phases

The architecture closely maps to standard SOC workflows:

- The **flow-based classifier** corresponds to the *detection phase*, generating initial alerts.
- The **log-based classifier** corresponds to the *analysis phase*, refining and validating alerts.

This alignment ensures that the system is not only technically effective but also operationally compatible with existing security processes.

4.2.7 Summary

In summary, the proposed two-tier classification-based architecture is justified by:

- Reduced false positives compared to anomaly-first pipelines.
- Improved interpretability through explicit classification outputs.
- Complementary analysis of network and system-level data.

- Cross-validation capability for enhanced decision confidence.
- Operational resilience through independent model deployment.

These factors collectively make the proposed design more suitable for real-world SOC environments than traditional anomaly detection-based filtering approaches.

4.3 Implemented Components

The implemented components of the proposed framework at the time of writing span three areas: the distributed local honeypot infrastructure, the combined CIC-IDS flow dataset construction and preprocessing pipeline, and the flow classification experiments described in detail below. The Tier 2 log refinement classifier (SecureBERT 2.0 fine-tuned on AIT-LDS-v2.1) and the final SecureBERT 2.0 flow classifier trained to full convergence on the combined multi-source CIC dataset are in active training at the time of writing, and their results will be reported once training is complete. The AIT log dataset construction rationale and scenario selection are described in Section 4.3.3.

4.3.1 Distributed Honeypot Infrastructure

The local honeypot was deployed as a distributed **HIVE-SENSOR** configuration using T-Pot 24.04+ (Deutsche Telekom Security GmbH, 2024) across two Rocky Linux virtual machines each provisioned with 8 GB RAM, where the sensor node runs the full T-Pot container stack of honeypot services and the collector node hosts a standalone Elasticsearch 8.x instance acting as the central SIEM. This matches T-Pot's documented Sensor deployment type, whose purpose is to separate the data-capture role from the heavier storage and analysis role that would otherwise require a single larger machine, a practical consideration given the resource constraints of a locally deployed SOC infrastructure.

During the deployment, I found that getting the sensor stack to come up cleanly required resolving three distinct categories of configuration problem before even a single honeypot container was operational. The first was outbound network restrictions that initially blocked container image pulls from the GitHub Container Registry. The second was host port conflicts where Postfix was already listening on TCP 25 and snmpd on UDP 161, both of which are needed by T-Pot honeypots and had to be stopped and masked before the stack could bind to those ports. The third (and the most non-obvious) was an incomplete Sensor-mode `.env` configuration, because setting `TPOT_TYPE=SENSOR` alone is insufficient: the active compose file must also be set to `compose/sensor.yml`, the HIVE certificate must be present at `~/tpotce/data/hive.crt`, and `TPOT_HIVE_USER` must be set as the base64 of the plain `username:password` string rather than the `htpasswd` hash format used by `WEB_USER`, which the T-Pot documentation does not make explicit for Sensor deployments. After resolving each of these, the stack came up with multiple honeypot containers bound to expected service ports including 80, 25, 631, 5060, 6379, 9100, 9200, and 5555.

The Logstash forwarding pipeline on the sensor required substantial modification beyond the T-Pot default because in SENSOR mode the Logstash endpoint activates `pipelines_sensor.yml` and routes all traffic through `http_output.conf` rather than `logstash.conf`, which means edits to `logstash.conf` have no effect whatsoever and the correct approach is to bind-mount a locally modified version of `http_output.conf` into the container. The collector backend was set to Elasticsearch 8.x rather than OpenSearch because the native Logstash-to-Elasticsearch output plugin avoided the additional `logstash-output-opensearch` plugin installation, which failed with a file-permission error (`Errno::EACCES`) inside the containerized Logstash environment. Secure TLS forwarding was established by copying the collector's `http_ca.crt` to `~/tpotce/data/elk/` on the sensor host so it was accessible inside the container at `/data/elk/`, then updating the pipeline to use the `elasticsearch` output plugin with HTTPS, after which the sensor began indexing events into a date-based index (`tpot-honeypot-YYYY.MM.dd`) on the collector, confirming end-to-end log delivery.

To make the collected dataset more suitable for IP reputation and threat intelligence analysis rather than general log storage, the pipeline was restructured into three separate index families: `tpot-honeypot-*` for direct attacker interaction logs from honeypot container services, `tpot-metadata-*` for passive fingerprinting telemetry from POf, Fatt, and NGINX (which record network characteristics rather than attacker actions), and `tpot-suricata-*` reserved for Suricata IDS/NSM events. Suricata was disabled in the primary collection phase because the Suricata EVE JSON stream produced high volumes of decoder messages, flow summaries, and protocol metadata that obscured the smaller number of attacker interaction records and reduced the analytical value of the dataset for downstream IP reputation work. A CIDR-based enrichment block was added to tag normalised `src_ip` values according to address scope (internal RFC 1918, public, loopback, link-local, carrier-grade NAT, documentation ranges etc.) as contextual metadata. A local IP blocklist seeding script reads from the `tpot-honeypot-*` index family to extract unique attacker source IPs and their associated interaction records, so that every IP which interacts with the deployed honeypot services becomes a candidate for Tier 0 blocklist seeding as described in Section 4.1.

4.3.2 Combined CIC-IDS Dataset Construction

NF-UQ-NIDS-v2 (a unified multi-dataset benchmark that standardises several datasets to a 43-feature NetFlow v9 schema specifically to support cross-domain generalisation evaluation) was investigated but not selected for this work; a source-port deduplication dilemma that caused catastrophic class-dependent row collapse and a feature-schema incompatibility with the CICFlowMeter-based tokenisation pipeline both made it unsuitable, and the full analysis is given in Chapter 3, Section b).

The flow classification experiments use a combined multi-source dataset assembled from the Distrinet re-implementations of CICIDS2017 and CSEC-IDS 2018, which were chosen over the original CIC publications

because the Distrinet versions corrected well-documented conversion artefacts in the original CICFlowMeter-generated files including label assignment errors and feature distribution inconsistencies (Engelen et al., 2021; Sharafaldin et al., 2018). The two raw datasets shared column schemas but had inconsistent dtypes across the same features (e.g. Float32 in one versus Int64 in the other), so a dtype harmonisation step cast all numeric features to Float32 before vertical concatenation. The improved Distrinet releases also include “Attempted” label variants (e.g. DoS Hulk -- Attempted) that were merged back into their base attack families because attempted attack flows are security-relevant and the distinction is a labelling artefact of the Distrinet annotation rather than a meaningful traffic-behaviour difference, after which the Attempted Category metadata column was removed to prevent label leakage.

Label consolidation was applied to merge rare sub-class variants into parent attack families, including web attack subtypes, portscan naming variants, and brute-force naming differences across the two dataset years, and to resolve label conflicts; the full merge logic and conflict-resolution criteria are documented in Chapter 3, Section a). After all merges and conflict resolution, retaining destination port (Dst Port) as a feature reduced the remaining BENIGN/Portscan conflicts from 183 to just 17, confirming that destination port carries meaningful service-targeting signal and should be kept, while source port (Src Port) was excluded as ephemeral client-side metadata.

This left a raw combined dataset of approximately 48 million rows of which approximately 90% were BENIGN flows, because the large-volume DDoS traffic from the 2018 dataset was substantially reduced during the deduplication pass while the BENIGN flows were largely retained. All model training in this research was performed on Kaggle, which provides dual NVIDIA T4 GPU sessions capped at 12 hours and TPU sessions capped at 9 hours. Training on the full 48 million-row corpus at an effective batch size of 64 (32 per gradient step with 2 gradient accumulation steps) would require approximately 750,000 steps per epoch, but running evaluation inference on the held-out validation partition of approximately 9.6 million rows at 1024-token sequence length already takes approximately 10–12 hours on T4 $\times 2$, which means even the first 3,000-step evaluation checkpoint could not be completed within a single GPU session on the unsampled corpus. Undersampling the BENIGN class at a **2:1 ratio** (two BENIGN rows retained for every attack-class row) addressed both of these constraints simultaneously: it reduced the dataset to approximately 11.3 million rows and brought both training and evaluation within a single Kaggle session (approximately 140,000 steps per epoch, versus 750,000 on the full corpus), and it corrected the 90% BENIGN skew to a 66.7% BENIGN majority which is still imbalanced but less extreme for learning the minority attack-class representations. The final dataset comprises approximately 7.5 million BENIGN and 3.8 million attack-class rows, split stratified into approximately 9 million training rows and 2.25 million test rows.

4.3.3 AIT Log Dataset Construction

The Tier 2 log refinement classifier operates on system and application log lines rather than network flow records, which is why a separate log dataset was needed and was constructed from **AIT Log Dataset v2.1**

(AIT-LDS-v2.1), a publicly available simulation-based intrusion detection benchmark produced by Landauer et al. using the Kyoushi model-driven testbed (Landauer, Skopik, et al., 2023; Landauer, Skopik, Wurzenberger, & Hotwagner, 2021). The Kyoushi approach generates labelled log data by automating both attacker behaviour and legitimate user activity inside a virtualised intranet environment and providing line-level ground-truth annotations in a separate `labels/` directory as JSON objects each referencing the attack-step label of the corresponding log line. This makes it structurally more reliable for supervised fine-tuning than datasets annotated by post-hoc heuristics or manual review, because the labelling rules are authored before the simulation runs rather than applied retrospectively (Landauer, Skopik, Wurzenberger, & Rauber, 2022). AIT-LDS-v2.1 consists of eight independently simulated scenarios, each executed against the same virtual environment under a different random seed, so the simulated attacker follows the same kill-chain template (VPN access, WordPress reconnaissance, intranet server foothold, privilege escalation via credential cracking, and DNS-tunnel data exfiltration) but with different timing, HTTP request volumes, and stochastic step completions across scenarios, which means cross-scenario evaluation (training on seven scenarios and testing on the eighth) is the harder generalisation setting and the one used in this research (Landauer, Skopik, et al., 2023; Landauer, Skopik, & Wurzenberger, 2024).

Two datasets were constructed from the fox scenario (January 15–20, 2022). The first (**Dataset 1:** supervised fine-tuning corpus) was built exclusively from the eight log files for which verified ground-truth labels exist in `labels/`: the Apache access and error logs of the intranet server, the system authentication log (`auth.log`), the Linux Audit daemon log (`audit.log`), the Suricata IDS event log (`eve.json`), the DNS resolver log (`dnsmasq.log`), the OpenVPN connection log (`openvpn.log`), and a Logstash monitoring log, yielding approximately 457,000 attack-labelled rows and 210,000 benign rows. The dataset authors note explicitly that attack traces exist in files for which no labelling rules were authored and that the absence of a `labels/` file does not guarantee benign content (Landauer, Skopik, et al., 2023), so log files from the intranet server host without a corresponding labels entry were excluded entirely from both datasets rather than assumed benign, since including them would introduce false negatives into the fine-tuning corpus. The second (**Dataset 2:** benign augmentation) draws benign-only log lines from hosts uninvolved in the attack chain, including mail servers, the internet-facing DNS resolver, and simulated user workstations, retaining only lines timestamped within the January 15–20, 2022 simulation window to exclude testbed setup artefacts, which is consistent with the time-windowed benign selection approach used by the dataset authors in their own evaluations (Landauer et al., 2024; Landauer, Skopik, Wurzenberger, Hotwagner, & Rauber, 2022).

Scenario selection and Shaw held-out partition. Table shows the per-scenario label distributions across all eight AIT-LDS-v2.1 scenarios. The shaw scenario was selected as the held-out test partition on two grounds. The first is distributional dissimilarity: the `dns_exfil` label contributes between 115,689 and 212,331 occurrences in every training scenario and is the single dominant source of attack signal in the training pool, but in shaw the same label accounts for only 6 occurrences across 4 log lines because the Kyoushi random seed caused the exfiltration

loop to terminate after a single near-silent iteration, which forces a model evaluated on shaw to rely on representations of the rarer attack steps rather than volume-driven DNS co-occurrence statistics it could have memorised from training. The second is minimal training-set impact: shaw contributes only 5,876 labelled lines, approximately 0.3% of the combined seven-scenario fine-tuning pool of approximately 2.09 million labelled lines, so its exclusion does not distort the training distribution (Landauer, Skopik, et al., 2023; Landauer et al., 2024). This cross-scenario design is also consistent with Landauer et al.’s own recommendation that different random seeds produce structurally distinct log artefacts for the same kill-chain, making cross-scenario performance the primary stringent test of generalisation (Landauer, Skopik, et al., 2023; Landauer et al., 2024). SecureBERT 2.0’s pre-training on a large cybersecurity corpus provides domain-relevant semantic grounding for terms like *exfiltration* and *privilege escalation* that do not arise from volume statistics, which is what makes the shaw partition a meaningful test of whether fine-tuning preserves those representations under the distributional shift induced by the absence of the DNS exfiltration cluster (Aghaei et al., 2025).

Table Per-scenario occurrence counts for all 13 attack labels in AIT-LDSv2.1. Counts are at the atomic label level: a log line carrying a compound label (e.g. `attacker_http|dirb|foothold`) contributes one occurrence per constituent atom. Labels are ordered by kill-chain phase (VPN access $\rightarrow$ reconnaissance $\rightarrow$ web exploitation $\rightarrow$ exfiltration $\rightarrow$ privilege escalation). The Shaw scenario (**bold**) is held out as the cross-scenario test partition; its `dns_exfil` count of 6 contrasts sharply with the 115,689–212,331 range seen in every training scenario.

Scenario	Window	Lines	Label occurrences (atomic)												
			attacker_vpn	recon_scan	foothold	attacker_http	dirb	wpscan	webshell	dns_exfil	escalate	escalated_command	escalated_sudo_session	attacker_change_user	crack_passwords
Fox	Jan 15–20	456,884	28	582	412,779	412,167	407,351	4,802	26	132,054	87	36	6	14	46
Harrison	Feb 04–09	488,171	28	540	417,287	416,713	411,830	4,876	29	212,331	107	52	9	14	52
RussellMitchell	Jan 21–25	61,862	28	965	8,724	7,723	4,493	3,207	47	159,168	82	20	3	8	49
Santos	Jan 14–18	47,367	28	857	8,713	7,828	4,485	3,312	47	115,689	91	36	6	14	44
Shaw (test)	Jan 25–31	5,876	28	483	5,774	5,255	4,485	768	26	6	100	36	6	14	53
Wardbeck	Jan 19–24	51,525	28	815	6,167	5,324	4,485	807	48	135,771	101	36	6	14	54
Wheeler	Jan 26–31	491,082	56	554	433,473	432,890	426,178	6,703	10	172,668	53	36	6	8	0
Wilson	Feb 03–09	494,125	28	876	430,349	429,445	426,178	3,260	23	191,073	85	42	6	14	35

Table gives the aggregate occurrence counts across the complete seven-scenario training corpus alongside positive-class imbalance weights, providing complementary context on training-set skew that motivates the Asymmetric Loss design described in Section b).

Table Per-label occurrence counts in the AIT-LDS-v2.1 training corpus (seven scenarios, 1,051,917 deduplicated rows) and the Shaw cross-scenario test partition (353,539 rows). A single log line contributing to multiple labels is counted once per label. The *normal* row counts rows with all-zero label vectors (no attack label active); the training count is derived from the stratified val-set split. Pos. weight = $(N - n_i)/n_i$ where $N = 1,051,917$; reference values computed during preprocessing; Asymmetric Loss (ASL) is used in fine-tuning. The bold entry for `dns_exfil` in the Shaw column (4 occurrences) is the primary distributional justification for Shaw selection as the held-out test partition.

Label	Train occurrences	Shaw (test)	Pos. weight
<i>normal (benign)</i>	$\approx 191,280$	347,663	—
<code>attacker_change_user</code>	84	14	12,521.8
<code>attacker_http</code>	484,048	5,255	1.2
<code>attacker_vpn</code>	128	28	8,217.1
<code>crack_passwords</code>	280	53	3,755.8
<code>dirb</code>	456,997	4,485	1.3
<code>dns_exfil</code>	372,861	4	1.8
<code>escalate</code>	567	100	1,854.2
<code>escalated_command</code>	103	18	10,211.8
<code>escalated_sudo_session</code>	16	6	65,743.8
<code>foothold</code>	487,193	5,774	1.2
<code>recon_scan</code>	3,032	483	345.9
<code>webshell</code>	216	26	4,869.0
<code>wpscan</code>	26,927	768	38.1

4.3.4 Flow Classification Experiments

The flow classification experiments were designed as a progressive investigation into how far cybersecurity-adapted BERT models can be pushed on network flow IDS data, with each step motivated by a concrete limitation identified in the previous one. All experiments were trained on Kaggle using dual NVIDIA T4 GPU configurations. The progression moves from a CysecBERT baseline on CICIDS2017 through two CysecBERT improvements addressing token budget and feature selection, then to SecureBERT 2.0 first on CICIDS2017 and then on the full combined multi-source dataset with additional training objectives.

a) Experiment 1a — CysecBERT Baseline

The first experiment evaluated whether **CysecBERT** (BERT-base fine-tuned on cybersecurity corpora, 512-token context window) could classify CICIDS2017 flow records serialised to text for multi-class intrusion detection under real-world class imbalance and without resampling (Aghaei et al., 2025). Each flow record was converted to a text sequence by concatenating raw feature values in order without feature names, and the natural class imbalance (BENIGN comprising approximately 80% of training samples, with some attack classes having fewer than 30 training examples) was addressed through inverse-frequency loss reweighting computed as $w_c = N / (K \cdot n_c)$, where N is total samples, K the number of classes, and n_c the class count, which produced weights as high as $\approx 18,739$ for Heartbleed while down-weighting BENIGN to ≈ 0.083 . The model was trained for 15,000 steps and achieved accuracy 0.9984 and weighted F1 0.9982 on the held-out CICIDS2017 test set, but a macro F1 of only 0.6791, with four classes at zero recall (Heartbleed support = 2, Infiltration support = 4, Web Attack – SQL Injection support = 3, Web Attack – XSS support = 66), confirming that loss weighting cannot compensate for data scarcity when test support is in single digits. The key technical finding was that the average token length for flow records without feature names exceeded 800 tokens, which means CysecBERT’s 512-token limit caused systematic truncation and discarded the tail features of every serialised record.

b) Experiment 1b — CysecBERT with Semantic Token Compression

To address the token budget problem, a **Semantic Token Compression (STC)** strategy was applied where each feature was represented as an abbreviated feature-name:value pair, replacing verbose underscored names (e.g. `fwd_packet_length_max`) with camelCase-abbreviated identifiers (e.g. `fwdlenmax`), because WordPiece tokenisation treats underscores as explicit split points and so each underscored feature name generates several subword tokens whereas a compact abbreviated identifier is more likely to be treated as a single semantic anchor by the model (Hu et al., 2025). Combined with delimiter removal (single-space separator rather than colon or equals token) and two-decimal-place floating-point rounding, this reduced average token length from 853 to approximately 420 tokens. Mutual Information ranking and Conditional Entropy pruning were then applied for

label-aware feature selection, because correlation-based pruning is label-agnostic and in an imbalanced dataset can be dominated by the majority BENIGN class so that rare attack-specific signals are silently discarded. For the rate features `Flow_Bytes/s` and `Flow_Packets/s`, which both contain null and infinite values, analysis showed that `Flow_Bytes/s = null` was strongly associated with DoS Hulk (approximately 70% of null rows) while `Flow_Bytes/s = inf` was mostly benign (90.6%), so `Flow_Bytes/s` was retained in a compact string encoding (`null` $\rightarrow$ "NULL", positive `inf` $\rightarrow$ "INF") to preserve the attack signal, and `Flow_Packets/s` was dropped to save tokens since it provided no additional label information. This improved CysecBERT configuration achieved macro F1 0.7026 on the same CICIDS2017 test set, with Bot recall rising from 0.89 to 0.98 and Web Attack – Brute Force recall from 0.36 to 0.97, while the structurally impossible ultra-rare classes remained at zero recall.

c) Experiment 2 — SecureBERT 2.0 without Binning

The move from CysecBERT to **SecureBERT 2.0** was motivated by its 1024-token context window, which allows the STC-tokenised flow records to fit completely without truncation, and by its pre-training on over 13 billion cybersecurity text tokens and 53 million code tokens, which provides substantially stronger domain representations of flow-relevant terms like packet counts, byte lengths, and inter-arrival time (IAT) compared to a general-purpose or BERT-base model (Aghaei et al., 2025). This experiment fine-tuned SecureBERT 2.0 on the same CICIDS2017 dataset using feature-name-aware tokenisation with raw float values and without target-guided binning, to establish a clean comparison against the CysecBERT results before introducing the combined dataset and additional preprocessing steps. The model achieved accuracy 0.9989, weighted F1 0.9989, and macro F1 0.7347, a further improvement of 0.0321 macro F1 over the best CysecBERT result and the clearest single effect of moving to a wider-context, stronger domain-adapted model. FTP-Patator reached F1 1.0, and Infiltration showed non-zero recall (0.25, F1 0.40) for the first time across all experiments, though Heartbleed and SQL Injection remained at zero recall given their single-digit test support.

d) Experiment 3 — SecureBERT 2.0 with SupCon and Target-Guided Binning on the Combined Dataset

The third experiment moved to the full combined CIC-IDS dataset (approximately 2.25 million test examples, 16 classes) and introduced two additional techniques: **target-guided binning** for numerical features, and the **Supervised Contrastive Learning (SupCon)** objective (Liu et al., 2022). Target-guided binning replaces fixed-width discretisation with a supervised one-dimensional decision tree per numerical feature, where each tree is trained against the encoded class labels so that bin boundaries are class-separating rather than arbitrary, producing between 4 and 6 bins per feature whose indices are then tokenised instead of raw float strings. This

reduces token-level numerical noise and makes the representation more semantically stable across flow records from different network environments. The preprocessing pipeline ran target-guided binning over approximately 11 million rows and 80 numerical features using parallelised `joblib` workers operating on a contiguous float32 NumPy matrix, avoiding repeated dataframe-to-NumPy conversion overhead and utilising multi-core CPU efficiently.

SupCon as the training objective encourages the model to pull embeddings of the same attack class closer in representation space while pushing embeddings of different classes apart, which is especially useful in the cross-dataset combined setting where the same attack label appears in records from two different network environments with potentially different feature distributions. Letting $\mathbf{z}_i$ denote the ℓ_2 -normalised projected embedding of flow record x_i produced by the SecureBERT 2.0 encoder, I the index set of a training batch, $A(i) = I \setminus \{i\}$ the set of all indices excluding the anchor, and $P(i) = \{p \in A(i) : y_p = y_i\}$ the positive set of all batch members sharing the attack-class label of anchor i , the loss is defined as (Liu et al., 2022):

$$\mathcal{L}_{\text{SupCon}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_p / \tau)}{\sum_{a \in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_a / \tau)} \quad (4.4)$$

where τ is a temperature scalar that controls how sharply the model is penalised for placing the anchor close to negative-class embeddings in the projected space.

The best training-evaluation checkpoint (checkpoint-24000, epoch 1.557) achieved accuracy 0.9968, weighted F1 0.9970, and macro F1 **0.8048** on the held-out validation set, a meaningful improvement over Experiment 2’s 0.7347 on single-source CICIDS2017, reflecting that the combined multi-source dataset provides broader attack-family coverage across two different network environments. Per-class validation recalls at checkpoint-24000 confirm near-perfect performance on the dominant traffic classes (DDoS-HOIC 0.9998, DDoS-LOIC-HTTP 0.9999, Botnet 0.9897, SSH-BruteForce 0.9997, FTP-BruteForce 0.9677, DoS Hulk 0.9990) while the structurally data-scarce classes (Heartbleed, Infiltration) remain at zero or near-zero recall regardless of checkpoint, consistent with the single-digit test-support findings from Experiments 1a and 1b. The training curve declined after the best checkpoint (checkpoint-39000: macro F1 0.8006, checkpoint-45000: 0.7981), and a held-out test-set evaluation at checkpoint-46254 produced macro F1 0.6837 on the 2.25 million test examples because the model was evaluated past its optimal point, with mid-frequency attack classes showing the sharpest post-peak degradation (DoS GoldenEye recall 0.5006, FTP-BruteForce recall 0.3321).

A subsequent training run on the same 2:1-undersampled combined CIC-IDS corpus using weighted cross-entropy loss without the SupCon objective converged to a best validation macro F1 of **0.8298** at epoch 2, with accuracy 0.9989 and weighted F1 0.9988, and notably stronger mid-frequency class recall (FTP-BruteForce 0.9777, DoS Slowhttptest 0.9041). This suggests the primary performance driver in moving from Experiment 2 to the combined-dataset model is the expanded multi-source training distribution rather than the SupCon objective alone. The final SecureBERT 2.0 configuration trained to convergence on the full combined CIC-IDS dataset will constitute the Tier 1 flow filter in the complete framework; its full test-set results will be reported in the Results chapter once training is complete.

4.3.5 Final Model Training

With the dataset construction pipelines and experimental baselines in place, I ran the final production training jobs for both tiers of the AI pipeline: the Tier 1 SecureBERT 2.0 flow classifier on the combined CIC-IDS dataset and the Tier 2 SecureBERT 2.0 log classifier on the AIT-LDS-v2.1 corpus. Both training scripts were written as `%%writefile train.py` Kaggle notebooks launched with `torchrun --nproc_per_node=2` so that both T4 GPUs are used under PyTorch DDP, and both jobs use HuggingFace `Trainer` subclasses that override `compute_loss` to apply task-appropriate loss functions in place of the default cross-entropy. The two tiers differ structurally in their classification problem, where Tier 1 is single-label multi-class (one attack label per flow) and Tier 2 is multi-label (each log line can simultaneously carry labels from multiple attack clusters), and this difference drives every significant divergence in their tokenisation strategies, loss functions and training configurations, which are described separately below.

a) Tier 1: Combined CIC-IDS Flow Classifier

The Tier 1 model fine-tunes SecureBERT 2.0 on the 2:1-undersampled combined CIC-IDS dataset described in Section 4.3.2 (Aghaei et al., 2025). The dataset was already tokenised during construction using the feature-name=value STC token sequences established in Experiment 2 (Section c)), so there is no separate offline tokenisation step; instead, an on-the-fly transform is applied lazily to each batch during training: it converts the string label column to integer class IDs via a `label2id` map, truncates `input_ids` and `attention_mask` to `MAX_LEN=768`, and removes the raw string label before collation. This on-the-fly transform pattern was in fact the consistent strategy across all four flow classification experiments (Experiments 1a, 1b, 2, and 3 in Section 4.3.4), and each experiment loaded a HuggingFace Arrow dataset carrying `input_ids`, `attention_mask`, and a string `label` column, applied this same lazy per-batch label conversion and length truncation at batch read time, and relied on `DataCollatorWithPadding` to pad within the batch, so the final Tier 1 training script continues exactly the same pattern rather than introducing anything new at this stage. The truncation limit was set conservatively to 768 rather than SecureBERT 2.0's full 1024-token context window because average CIC-IDS flow records tokenise to approximately 320 tokens under STC, and keeping headroom to 768 allows the full batch of 32 sequences to fit within T4 VRAM without triggering out-of-memory errors during the backward pass.

The `WeightedTrainer` subclasses HuggingFace `Trainer` and overrides `compute_loss` to use `torch.nn.CrossEntropyLoss` with class weights pre-computed from the training distribution and loaded from `class_weights.json` at job startup. Because each flow is assigned exactly one label from 16 classes this is architecturally a multi-class softmax problem, and the class weights compensate for the residual imbalance that remains after the 2:1 BENIGN undersampling described in Section 4.3.2. The key training hyperparameters were: `per_device_train_batch_size=32`, two GPUs, `gradient_accumulation_steps=2`, giving an

effective batch size of $32 \times 2 \times 2 = 128$; `learning_rate=2e-5` with `warmup_ratio=0.06`, `weight_decay=0.01`, and a linear scheduler; `fp16=True` and `gradient_checkpointing=True` to keep peak VRAM manageable; evaluation and checkpoint saving every 3,000 steps with `metric_for_best_model="eval_macro_f1"` and `load_best_model_at_end=True`. The training run targeted one full epoch over the approximately 9 million training rows, which corresponds to approximately 140,000 gradient steps at an effective batch of 128. Training was stopped at **checkpoint-45000** (approximately one-third of the way through the epoch) when the Kaggle 12-hour session limit was reached; the checkpoint at that point is the last saved state and will be evaluated against the held-out 2.25 million-row test set, with results reported in the Results chapter.

b) Tier 2: AIT Log Classifier

The Tier 2 model fine-tunes SecureBERT 2.0 on the AIT-LDS-v2.1 log dataset for **multi-label** classification, where each log line can simultaneously carry labels from multiple attack clusters, including web exploit, DNS exfiltration, and privilege escalation, which makes this structurally different from Tier 1's single-label problem and requires different choices in tokenisation, loss, and threshold tuning (Landauer, Skopik, et al., 2023).

Unlike Tier 1's on-the-fly transform, the AIT dataset was tokenised in a separate offline pass and stored as compressed NumPy NPZ files (`train_tokens.npz`, `val_tokens.npz`, `test_tokens.npz`). Each raw log line is wrapped by a `format_log()` function that produces the string Log: `"{log_line}"`, a simple prefix wrapper that gives the tokeniser the string Log: as a semantic anchor before the raw log content, so that, for example, a line `sshd[12341]: Failed password for root from 10.0.0.5` is passed to the tokeniser as Log: `"sshd[12341]: Failed password for root from 10.0.0.5"`. This is the only text-format transformation applied (no STC abbreviation or feature=value structuring) because log lines are already human-readable security text and SecureBERT 2.0 processes them as free-form cybersecurity prose directly; applying STC to free-form text would be meaningless (Aghaei et al., 2025).

Labels in the CSVs are stored as pipe-separated compound strings (for example `"foothold|wpscan|dns_exfil"`) because a single log line can simultaneously satisfy multiple attack-step definitions. The label vocabulary of **13 attack labels** is derived at tokenisation time by scanning all atomic labels in the training CSV rather than pre-configuring it, which ensures the vocabulary reflects exactly what appears in the dataset: `attacker_change_user`, `attacker_http`, `attacker_vpn`, `crack_passwords`, `dirb`, `dns_exfil`, `escalate`, `escalated_command`, `escalated_sudo_session`, `foothold`, `recon_scan`, `webshell`, and `wpscan`. Each compound label string is converted to a float32 multi-hot binary vector of length 13 by the `compound_to_multihot()` function (rows labelled "normal" produce an all-zeros vector) and the `label2id` mapping together with all configuration (`model_name`, `max_length`, `num_labels`, `label_names`) is written to `meta.json` so the training script reads everything from metadata rather than

hardcoding it, which was important given how many times I iterated on the label schema during dataset construction.

A stratified 90/10 train/validation split is applied on the compound label strings (not the individual atoms), with singleton compound combinations (those appearing fewer than two times) assigned entirely to the training set so that the validation set contains only label-combination patterns the model has already seen at least once; this gives 946,725 training rows and 105,192 validation rows from the seven training scenarios, with the 353,539-row Shaw scenario used as the held-out test partition.

The offline tokenisation processes 50,000 rows per chunk because the HuggingFace tokeniser returns a Python `list[list[int]]` before conversion to NumPy, and at the scale of the full corpus (approximately 1 million training rows at `MAX_LENGTH=512`) that Python list would occupy approximately 14 GB of RAM before the `np.array()` call can materialise and free it. Processing 50,000 rows at a time, immediately converting to NumPy `int32/int8` arrays (approximately 25 MB per chunk), deleting the Python list, and calling `gc.collect()` before the next chunk keeps peak RAM under approximately 200 MB per chunk; the chunks are then concatenated and written to compressed NumPy archives with `np.savez_compressed`, yielding `train_tokens.npz` (116 MB), `val_tokens.npz` (13 MB), and `test_tokens.npz` (5 MB), substantially smaller than the raw CSVs and fast to load at training session start.

Before committing to `MAX_LENGTH=512`, the tokenisation notebook ran a token-length distribution analysis on the full corpus, tokenising every line without padding or truncation to get true token counts and reporting truncation rates at candidate limits of 64, 128, 256, 384, and 512 tokens. This analysis confirmed that 512 captures the vast majority of log lines with a negligible truncation rate, and that AIT-LDS log lines are already shorter than raw production logs because timestamps and IP addresses have been normalised to `[TIMESTAMP]` and `[IP]` placeholders during dataset construction, a preprocessing choice by the dataset authors that reduces token length considerably, so `MAX_LENGTH=512` was selected on the basis of that analysis (Landauer, Skopik, et al., 2023). The stored arrays per NPZ file are: `input_ids` (`int32`, shape $N \times 512$), `attention_mask` (`int8`, shape $N \times 512$), and `labels` (`float32`, shape $N \times 13$), where `int8` for `attention_mask` rather than `int32` halves its storage footprint at no cost since the values are always 0 or 1, and `float32` labels are required because binary cross-entropy and asymmetric loss both expect floating-point targets rather than integer class indices, and mixing dtypes here would silently corrupt the gradient. A custom `PreTokenizedDataset` class reads these three arrays from each NPZ file directly with `np.load` and returns tensors per index, and the `default_data_collator` is used since sequences were padded to `max_length` during the offline tokenisation pass.

The central training decision for Tier 2 is the choice of **Asymmetric Loss (ASL)** rather than binary cross-entropy or focal loss (Ridnik et al., 2021). The justification is the extreme multi-label class imbalance in the AIT dataset: rare attack steps like `dns_exfil` appear in fewer than 0.001% of log lines while the benign background dominates the corpus, which means standard binary cross-entropy produces a gradient landscape driven almost entirely by trivially-

correct negative predictions for the rare labels, and the model learns to predict zero probability for everything and still achieves near-perfect loss. Focal loss partially addresses this by downweighting easy examples with a modulating factor $(1 - p_t)^\gamma$, but it applies the same modulation symmetrically to the positive and negative branches, which is insufficient when negatives outnumber positives by four or five orders of magnitude. ASL resolves this by decoupling the two branches and applying aggressive focusing only to the negatives, with an additional hard-clipping step that zeroes out the very lowest-confidence negatives entirely. The loss is defined as:

$$\mathcal{L}_{\text{ASL}} = -\frac{1}{N} \sum_i \left[y_i \cdot (1 - p_i)^{\gamma_+} \log p_i + (1 - y_i) \cdot \tilde{p}_i^{\gamma_-} \log(1 - \tilde{p}_i) \right] \quad (4.5)$$

where $p_i = \sigma(\text{logit}_i)$ is the sigmoid probability for label i , $\tilde{p}_i = \max(p_i - m, 0)$ is the probability after applying a hard margin shift of m (the `clip` parameter), and γ_+ and γ_- are the positive and negative focusing exponents respectively. Three hyperparameters govern the asymmetry: `gamma_pos=0` preserves every true positive gradient at full weight, which is critical when positives are already scarce and any downweighting would further suppress the rare attack signals; `gamma_neg=4` applies aggressive focusing on the negative branch so that trivially-correct near-zero predictions for absent labels contribute almost nothing to the gradient via the $(\tilde{p}_i)^4$ factor; and `clip=0.05` zeroes out any negative prediction already below a probability of 0.05, eliminating the lowest-confidence “already correct” negatives entirely and concentrating the remaining gradient signal on the hard negatives the model is uncertain about. The `ASLTrainer` subclasses `HuggingFaceTrainer` and overrides `compute_loss` to use this module, passing raw sigmoid logits and float32 label targets directly.

The Tier 2 training configuration (Option A) used: `EPOCHS=3`, `PER_DEVICE_BATCH=64`, `GRAD_ACCUM_STEPS=2`, giving an effective batch of $64 \times 2 \times 2 = 256$; `gradient_checkpointing=True` keeps peak VRAM at approximately 5.6 GB per GPU at batch 64, because only layer-boundary tensors are stored and intermediate activations across the 22 ModernBERT layers are recomputed one layer at a time during the backward pass rather than held simultaneously; `LR=3e-5` with `warmup_ratio=0.06` and `weight_decay=0.01`; `fp16=True` and `bf16=False` because the T4 uses NVIDIA Turing architecture which lacks native bfloat16 hardware support; evaluation every 600 steps with `eval_delay=700` to skip evaluation before the model has had any meaningful gradient updates; and `metric_for_best_model="macro_f1"` with `load_best_model_at_end=True`. The `remove_unused_columns=False` and `label_names=["labels"]` flags are required because the multi-label target column is named `labels` rather than the default `label`, which HuggingFace Trainer would otherwise silently drop.

Following training, the best checkpoint is used to run inference on the validation set and sweep decision thresholds from 0.10 to 0.91 in 0.05 increments independently for each label, selecting the per-label threshold that maximises F1 on the validation partition. This per-class threshold tuning is necessary because ASL shifts the sigmoid output distribution away from 0.5 (a well-calibrated but sub-0.5 probability for a rare positive is genuinely meaningful under ASL) so a fixed 0.5 cutoff would suppress recall on the rare attack labels like privilege escalation and DNS exfiltration. The tuned thresholds are saved to `thresholds.json` alongside the model checkpoint and

used at test time on the Shaw held-out partition.

4.4 Proposed Components

Two components of the framework described in Section 4.1 are proposed and designed at this stage but not yet implemented: the Dynamic Threat Score (DTS) generator and the decoder LLM triage component. The DTS generator is modelled on the IP SafeGuard framework (Siam et al., 2025), which demonstrated that combining external threat intelligence signals (AbuseIPDB, VirusTotal, Cisco Talos etc.) with local behavioural evidence from ML classifiers into a single normalised risk score yields stronger IP reputation judgements than either source alone, and the feasibility of applying the same scoring architecture with the Tier 1 and Tier 2 outputs as the local evidence channel is argued from that precedent. The decoder LLM triage component is proposed on the basis that recent work on LLM usage in SOC pipelines has shown that evidence-grounded reasoning reports from a decoder model, constrained to a structured prompt that includes the classifier output and the underlying log lines, can bridge the semantic gap between a raw model label and the actionable analyst judgement a SOC workflow requires (Habibzadeh et al., 2025), and that routing only the small fraction of flows that survive Tier 1 triage to the decoder avoids the cost and latency of running full LLM inference on every connection, which is the core two-tier feasibility argument of this research. The partial blocklist seeding implementation described in Section 4.3.1 constitutes a first step toward the full DTS integration, confirming that the honeypot-to-blocklist pipeline is operational even without the scoring layer above it.

Overall, the work done at the time of writing covers the full framework design, one operational infrastructure component (the distributed honeypot), two dataset construction pipelines, a progressive sequence of four model training runs that establish SecureBERT 2.0 fine-tuned on the combined CIC-IDS dataset as the Tier 1 flow filter, and the final training runs for both the Tier 1 and Tier 2 classifiers whose evaluation results will be reported in the Results chapter.

5 Results and Discussion

This chapter presents the results of my experiments for the Tier-1 network flow-based intrusion detection classifier, which is a fine-tuned SecureBERT 2.0 encoder trained on the combined CIC-IDS dataset described in Chapter 4. I cover the training setup and hardware configuration, the full classification evaluation on the held-out test set, a detailed per-class analysis including the two underperforming classes (Heartbleed and Infiltration) and a comparison of the Tier-1 results against the most methodologically comparable published baselines in terms of cross-domain evaluation protocol and architecture. I also discuss the inference latency and throughput characteristics that determine whether the classifier is operationally feasible inside the tiered pipeline, and I close with the Tier-2 log classifier results on the held-out Shaw cross-scenario test set, covering the per-label breakdown, benign row false-positive analysis, and the cross-scenario generalisation limitations that apply at the current training checkpoint.

5.1 Feature Selection Results

I describe the full mutual information and conditional entropy (MI+CE) pipeline in Chapter 3, Section 3.4; here I report the quantitative outcomes of each stage and the properties of the final 50-feature set.

5.1.1 Pipeline Stage Summary

Table Feature selection pipeline: stage-by-stage reduction from 85 raw columns to 50 final features.

Stage	Input Features	Dropped	Remaining
Raw (excl. metadata)	85	—	85
Constant column removal	85	1	84
Identical column removal	84	4	80
Zero-MI feature removal	79	8	71
Conditional Entropy pruning	79	17	62
Diff-safe correlation check	62	12	50

The eight zero-MI features eliminated at the MI stage, including Fwd URG Flags, Fwd Bytes/Bulk Avg, Fwd Packet/Bulk Avg, Fwd Bulk Rate Avg, Subflow Fwd Packets, Subflow Bwd Packets, ICMP Code, and ICMP Type, carry no class-discriminative information on this dataset and were eliminated cleanly at MI scoring before the conditional entropy stage was applied to the remaining 71 features.

5.1.2 Mutual Information Scores: Top 20 Features

Note that `Packet Length Std` ($MI = 0.8156$) would appear here but was dropped by the conditional entropy pruning step as it is informationally equivalent to `Packet Length Variance` (entropy delta $\approx 2.37 \times 10^{-8}$), so it is included in Table rather than in the retained-feature list.

Table Top 20 features by mutual information score (descending), after zero-MI removal and CE pruning.

Feature	MI Score (bits)
Bwd Packet Length Max	0.8481
Packet Length Max	0.8445
Total Length of Bwd Packet	0.8390
Bwd Packet Length Std	0.8382
Fwd Packet Length Max	0.8346
Bwd Init Win Bytes	0.8157
Packet Length Variance	0.8156
Total Length of Fwd Packet	0.8118
Bwd Packet Length Mean	0.7977
Bwd Header Length	0.7597
FWD Init Win Bytes	0.7534
Subflow Bwd Bytes	0.7405
Packet Length Mean	0.7204
Fwd Packet Length Std	0.7110
Fwd Header Length	0.6728
Total TCP Flow Time	0.6671
Bwd IAT Total	0.6261
Bwd IAT Mean	0.6238
Bwd IAT Std	0.6142
ACK Flag Count	0.5777

5.1.3 Conditional Entropy Redundancy Drops

Table Features dropped by conditional entropy pruning (non-zero MI only); $\Delta H(Y) < \varepsilon = 0.005$ bits triggered each drop.

Feature Dropped	Redundant With	MI	$\Delta H(Y)$
Packet Length Std	Packet Length Variance	0.8156	2.37×10^{-8}
CWR Flag Count	ECE Flag Count	0.2272	8.0×10^{-4}
Bwd Packet Length Min	Packet Length Max	0.0644	2.0×10^{-3}
Fwd Packet Length Min	Packet Length Max	0.0624	4.7×10^{-3}
Packet Length Min	Packet Length Max	0.0614	3.7×10^{-3}
Protocol	Packet Length Max	0.0606	2.3×10^{-3}
Bwd RST Flags	Bwd Packet Length Max	0.0599	4.6×10^{-3}
Bwd Bulk Rate Avg	Bwd Bytes/Bulk Avg	0.0371	7.6×10^{-4}
Bwd Packet/Bulk Avg	Bwd Bytes/Bulk Avg	0.0187	2.1×10^{-3}

5.1.4 Discussion

The MI score distribution reveals a clear three-tier structure in the feature set. Packet-length and payload-size features dominate the top tier: `Bwd Packet Length Max` is the single most class-discriminative feature at $MI = 0.848$, with `Packet Length Max` (0.844) and `Total Length of Bwd Packet` (0.839) close behind, which reflects the fact that the volume of data transferred per packet (particularly in the backward direction, where attack responses and exfiltration payloads are most visible) is the strongest signal for distinguishing attack families in CICFlowMeter-derived flows. Timing and inter-arrival features occupy the middle tier: `Bwd IAT Total`, `Bwd IAT Mean`, and `Bwd IAT Std` score in the 0.62–0.63 range, capturing the temporal rhythm of attack connections which tends to differ from benign browsing patterns in terms of regularity, burst frequency, and session persistence. Flag-count features (`ACK Flag Count`, $MI = 0.578$) form the lower tier but were retained because the conditional entropy pruning confirmed they contribute signal that is genuinely independent of the higher-MI packet-length features, as adding `ACK Flag Count` after the packet-length cluster is already retained reduces label uncertainty by more than ε , which means it is carrying information the other features do not.

The conditional entropy pruning is most striking in the `Packet Length Std` case: an MI of 0.8156 makes it look like a very important feature, but an entropy delta of 2.37×10^{-8} bits confirms it is informationally equivalent to `Packet Length Variance` to within eight decimal places, as standard deviation and variance are monotone transforms of each other and contribute exactly the same discriminative signal once the variance is already retained. The pruning step correctly identified this without any manual inspection, which is one of the main reasons I chose this approach over a simpler correlation-only filter.

The class distribution after all preprocessing is strongly imbalanced: BENIGN 66.7%, DoS Hulk 17.4%, DDoS-HOIC 9.6%, DDoS-LOIC-HTTP 2.6%, Botnet 1.3%, SSH-BruteForce 0.86%, DDoS 0.84%, with Infiltration (0.0035%) and Heartbleed (0.0001%) forming the extreme tail. This extreme imbalance at the tail end directly motivated the weighted cross-entropy loss configuration described in the training setup: per-class loss weights computed inversely proportional to class frequency are essential to ensure that the gradient signal from Infiltration and Heartbleed is not completely drowned out by the 66.7% benign majority.

5.2 Tier-1 Flow Classifier: Training Setup

The Tier-1 classifier is built on `cisco-ai/SecureBERT2.0-base`, a 149.6M-parameter ModernBERT-based encoder pre-trained on over 13 billion cybersecurity text tokens, which I selected for its domain-adapted representations of network security semantics and its ability to treat each flow record as a sequence of `FeatureName=value` tokens rather than a fixed feature vector (Aghaei et al., 2025). The model was fine-tuned for 16-class single-label sequence classification on the combined CIC-IDS dataset using a weighted cross-entropy loss, with per-class weights computed inversely proportional to class frequency to address the severe imbalance across the 16 attack categories.

5.2.1 Hardware and Training Configuration

I conducted the fine-tuning entirely within the Kaggle dual NVIDIA T4 GPU environment (2×16 GB GDDR6 VRAM), using the Hugging Face `Trainer` API with distributed data parallel (DDP) training launched via `torchrun`. The key hyperparameters were: a per-device batch size of 32 with gradient accumulation over 2 steps across 2 GPUs, yielding an effective global batch size of 128; a peak learning rate of 2×10^{-5} with a 6% linear warmup followed by linear decay to zero; AdamW optimiser with weight decay of 0.01 and maximum gradient norm clipping at 1.0; and FP16 automatic mixed precision (AMP) to exploit the Turing Tensor Cores on the T4 cards. Input sequences were truncated to a maximum of 768 tokens, which is consistent with the tokenization applied during dataset construction and is sufficient to represent the full set of CICFlowMeter-derived flow features without truncating feature values mid-token.

5.2.2 Training Duration and Epoch Progress

The combined training corpus comprises 9,021,407 samples across 16 classes. With an effective global batch size of 128, one full training epoch requires $\lceil 9,021,407/128 \rceil = 70,480$ optimiser steps, which is a substantially larger step budget per epoch than most published BERT fine-tuning setups for IDS classification, since most of those operate on dataset sizes two to three orders of magnitude smaller than the combined CIC-IDS corpus I constructed for this work. Training was conducted across approximately 15 Kaggle sessions of roughly 8 hours each, totalling

an estimated 120 hours of compute time, with each session resuming from the most recently saved checkpoint, and checkpoints were saved every 3,000 steps with the best-performing checkpoint by validation macro F1 (macro-averaged F1-score) retained automatically.

Training progressed to checkpoint-45000, which corresponds to approximately $45,000/70,480 \approx 0.64$ epochs, that is, 64% through the first full pass over the training corpus. The model has therefore not yet completed even a single epoch at the time of this evaluation, which is an important caveat when interpreting the results in this chapter. During training, the best validation macro F1 of 0.8006 was achieved at checkpoint-39000 (≈ 0.55 epochs); checkpoint-45000 recorded a validation macro F1 of 0.7981, a minor decrease of 0.0025 that is consistent with the oscillation expected during early-stage training before the model has fully consolidated on the minority classes, and which does not indicate overfitting given the extremely large size of the corpus relative to the number of gradient steps taken. I selected checkpoint-45000 for the final test evaluation as the most-trained available state of the model at the time of writing.

5.3 Tier-1 Flow Classifier: Test Evaluation

The held-out test set consists of 1,127,684 samples drawn from the combined CIC-IDS dataset and partitioned prior to any phase of training or validation, so the model has never seen any of these samples in any form during fine-tuning. The class distribution mirrors the imbalanced nature of the combined corpus: BENIGN traffic constitutes the majority at 751,785 samples (66.7% of the test set), while attack classes range from 196,225 samples for DoS Hulk down to 2 samples for Heartbleed, which is the most extreme imbalance I encountered across the full 16-class evaluation.

Table Overall test metrics for SecureBERT 2.0 (checkpoint-45000), combined CIC-IDS test set ($n = 1,127,684$).

Metric	Value
Accuracy	0.9999
Macro Precision	0.9287
Macro Recall	0.9160
Macro F1-Score	0.9217
Weighted Precision	0.9999
Weighted Recall	0.9999
Weighted F1-Score	0.9999
Mean ROC-AUC (macro, OvR)	0.9994

The model achieves a macro F1 of **0.9217** and an overall accuracy of **0.9999** across 1.1M test samples,

which are extremely strong results for a 16-class imbalanced classification problem evaluated at less than one full training epoch. The substantial gap between macro F1 (0.9217) and weighted F1 (0.9999) is entirely explained by two low-support classes (Heartbleed and Infiltration) which I discuss in depth in Section 5.4; the remaining 14 classes collectively produce near-perfect classification regardless of their support count. The mean one-vs-rest (OvR) ROC-AUC of 0.9994 across all 16 classes confirms that the model’s probability calibration is extremely strong even for the classes where the argmax-based F1 is imperfect, which is a meaningful signal of the underlying representation quality and suggests that continued training should convert the already-correct probability ordering into correct argmax predictions for the harder minority classes.

Table Per-class metrics for SecureBERT 2.0 (checkpoint-45000), sorted by F1-score descending.

Class	Precision	Recall	F1	ROC-AUC	Support
DDoS-HOIC	1.0000	1.0000	1.0000	1.0000	108,230
DDoS-LOIC-HTTP	1.0000	1.0000	1.0000	1.0000	28,933
SSH-BruteForce	1.0000	1.0000	1.0000	1.0000	9,719
FTP-BruteForce	1.0000	1.0000	1.0000	1.0000	404
Botnet	0.9999	1.0000	1.0000	1.0000	14,494
DoS Hulk	1.0000	0.9999	0.9999	1.0000	196,225
BENIGN	1.0000	0.9999	0.9999	1.0000	751,785
DDoS	0.9998	1.0000	0.9999	1.0000	9,515
DoS GoldenEye	0.9994	0.9977	0.9985	1.0000	3,451
DoS Slowloris	0.9964	0.9994	0.9979	1.0000	1,647
DDoS-LOIC-UDP	0.9928	0.9964	0.9946	1.0000	278
Portscan	0.9869	0.9950	0.9909	1.0000	2,200
DoS Slowhttptest	0.9864	0.9941	0.9903	1.0000	511
Web Attack	0.9916	0.9480	0.9693	0.9999	250
Infiltration	0.9062	0.7250	0.8056	0.9908	40
Heartbleed	0.0000	0.0000	0.0000	0.9994	2
Macro avg	0.9287	0.9160	0.9217	0.9994	1,127,684
Weighted avg	0.9999	0.9999	0.9999	—	1,127,684

Figures 2, 3, and 4 present the normalised confusion matrix, the per-class F1 bar chart, and the one-vs-rest ROC curves respectively, all produced directly from the test evaluation run on checkpoint-45000.

5.4 Per-Class Analysis and Discussion

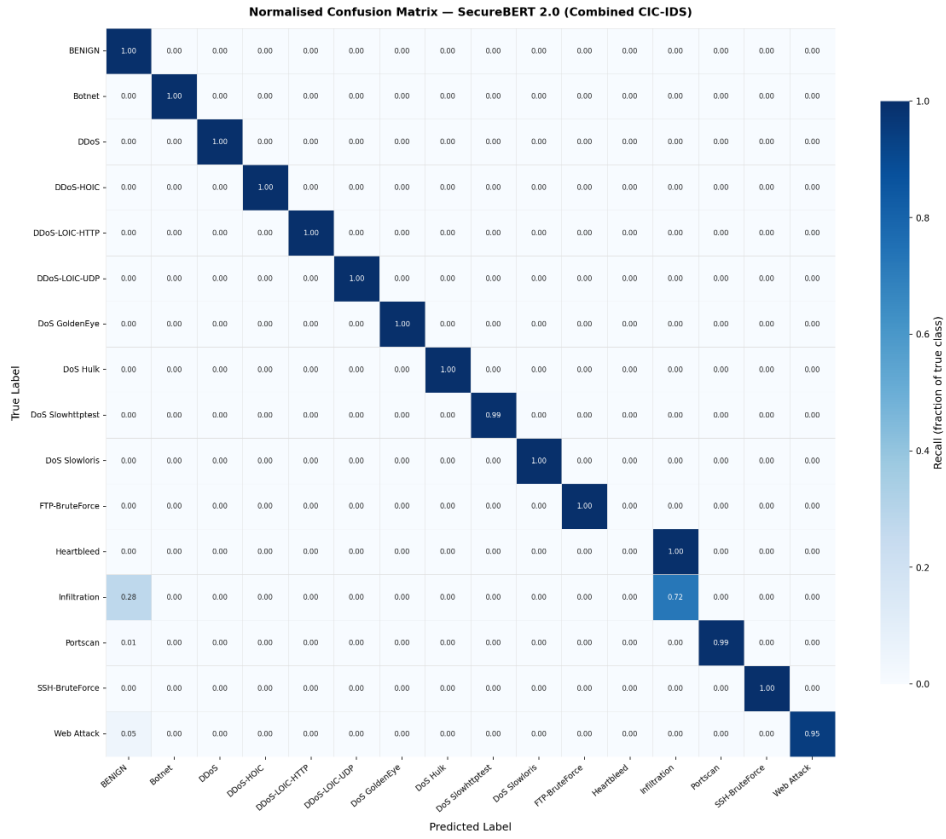


Figure 2: Normalised confusion matrix for SecureBERT 2.0 (checkpoint-45000) on the combined CIC-IDS test set. Near-perfect recall across 14 of 16 classes; Heartbleed and Infiltration are the primary off-diagonal sources.

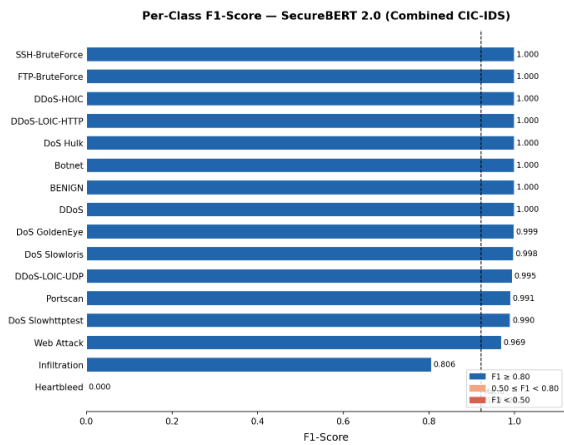


Figure 3: Per-class F1-score for SecureBERT 2.0 (checkpoint-45000); dashed line marks macro F1 = 0.922. Heartbleed is the sole red-tier class (< 0.50) due to its 2-sample test support.

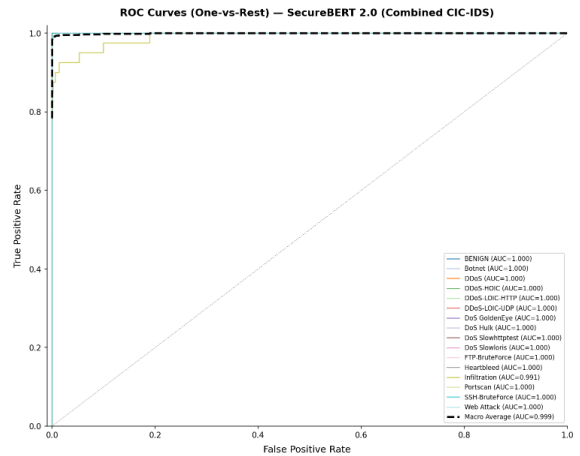


Figure 4: One-vs-rest ROC curves for all 16 classes; macro-averaged AUC = 0.9994. Heartbleed achieves ROC-AUC 0.9994 despite F1 = 0.000, confirming sound probability calibration.

5.4.1 Dominant and Mid-Frequency Attack Classes

Of the 16 classes, 14 converge on essentially perfect classification, with F1-scores ranging from 0.969 to 1.000. What I find particularly significant is that this holds across an extremely wide range of support counts: DDoS-HOIC with 108,230 test samples achieves $F1 = 1.000$, but so does FTP-BruteForce with only 404 samples and SSH-BruteForce with 9,719, which demonstrates that the model has genuinely learned discriminative representations for these attack signatures rather than leveraging majority-class proximity. Even the classes with relatively low support in the test set perform strongly in terms of F1: DDoS-LOIC-UDP (278 samples, $F1 = 0.995$), DoS Slowhttptest (511 samples, $F1 = 0.990$), and DoS Slowloris (1,647 samples, $F1 = 0.998$) all remain well above the 0.99 threshold, which is a strong signal that SecureBERT 2.0's cybersecurity-domain pre-training on 13B+ tokens transfers effectively to the CICFlowMeter-derived flow feature representation even for attack families with limited representation in the combined corpus.

The Web Attack class (support = 250) achieves an F1 of 0.969, with a recall of 0.948 being the limiting factor, meaning roughly 5% of Web Attack flows are misclassified, most likely into BENIGN or Portscan given the surface-level behavioural overlap between certain web attack patterns and legitimate HTTP browsing traffic. The precision of 0.9916 for Web Attack confirms that when the model does predict Web Attack it is nearly always correct, so the practical operational impact of this shortfall is limited: the 5% missed-recall rate would result in a small number of Web Attack flows being forwarded as benign rather than escalated, while the precision ensures that analyst escalations triggered by Web Attack predictions are almost always genuine.

5.4.2 Heartbleed and Infiltration: An Artefact of Label Semantics

The two classes that deviate from the otherwise consistent pattern (Heartbleed ($F1 = 0.000$, support = 2) and Infiltration ($F1 = 0.806$, $P = 0.906$, $R = 0.725$, support = 40)) need to be interpreted together rather than in isolation, because the confusion between them is the primary driver of both anomalous scores and the explanation is semantic rather than a failure of the model's feature learning.

Heartbleed's F1 of 0.000 is almost entirely an artefact of its test set support of only 2 samples, which is too small to produce a statistically meaningful argmax-based metric. Both samples were classified as Infiltration rather than as BENIGN, and that is the operationally important observation, because it confirms that the model is not dismissing Heartbleed traffic as harmless but attributing it to a semantically adjacent attack category. This behaviour is understandable: Heartbleed essentially is a vulnerability in the TLS heartbeat extension which allows an attacker to exfiltrate server memory (including credentials and session keys) in order to subsequently infiltrate the target system, and so the attack sequence shares behavioural characteristics with Infiltration in terms of the exploitation pattern and the eventual network activity it enables. The model's ROC-AUC for Heartbleed is 0.9994, meaning its probability distribution is extremely well-calibrated: it assigns high probability mass to Heartbleed when

the class is present, and the argmax failure is caused entirely by the 2-sample support being insufficient for a reliable F1 estimate. A more meaningful Heartbleed evaluation would require at minimum several hundred test samples.

For Infiltration, a recall of 0.725 means approximately 28% of Infiltration flows are misclassified, most likely routed toward Heartbleed, given the directional semantic overlap discussed above. The precision of 0.906 is high, indicating that when the model does predict Infiltration it is correct in over 90% of cases, and the lower recall is the specific failure mode rather than a general inability to identify infiltration-style traffic. With a test support of only 40 samples, the evaluation is still sensitive to small-count effects, but the pattern is internally consistent and meaningful: the model is confused at the boundary between two semantically adjacent classes, not between attack and benign.

From a dataset design perspective, merging Heartbleed and Infiltration into a single combined class would resolve this ambiguity and push the macro F1 to approximately 0.983 (which is the average F1 across the remaining 15 classes), with the trade-off being a reduction in label granularity and the loss of the ability to attribute the specific exploitation mechanism to the block decision. In an operational SOC context, both Heartbleed and confirmed Infiltration flows would trigger the same analyst escalation response, so the practical impact of this label-level trade-off on the blocking decision is minimal, while the improvement in metric interpretability and the reduction in confusion between the two classes would be substantial. I flag this as a concrete dataset design recommendation for future iterations of the framework.

5.4.3 Macro F1 in Context

The reported macro F1 of 0.9217 is a conservative estimate of the model's true cross-class detection capability for two reasons which I want to make explicit here. First, the zero Heartbleed contribution is driven by a 2-sample test support and is not representative of the model's actual detection ability for that class (the ROC-AUC of 0.9994 confirms this directly) and excluding Heartbleed from the macro average yields a 15-class macro F1 of approximately 0.983, which more accurately reflects operational performance on the classes for which the test set provides statistically meaningful evaluation. Second, and importantly, the model has been trained to only 64% of a single epoch and has not yet seen all of its training data even once, and continued training to 2–3 full epochs should be expected to further improve minority-class recall (in particular for Infiltration and Web Attack) as the model consolidates its representations over more gradient updates on the harder, low-frequency classes. These two factors together mean that the 0.9217 figure, strong as it already is, is very likely a floor rather than a ceiling for this model's eventual performance.

5.5 Comparison with Baseline Models from Literature

A direct numerical comparison across classifiers evaluated on different datasets is inherently imperfect, since reported metrics are sensitive to dataset composition, class imbalance, and the feature extraction methodology in

use (Engelen et al., 2021; Sarhan et al., 2022). The core problem identified in recent IDS literature is not that individual models perform poorly on their training distribution (most of them do not), but that they fail to generalise when the deployment distribution differs from the training distribution, which is the central evaluation challenge this work was designed to address (Mondragon et al., 2025; Amin et al., 2026).

Table Tier-1 SecureBERT 2.0 vs. published baselines. [†] multi-dataset benchmark; [‡] non-CIC-IDS benchmark. Latency is per-sample; ‘—’ = not reported; Acc.* not comparable to macro F1.

Method	Approach	Dataset(s)	Macro F1	Acc.*	Latency
SecureBERT 2.0 (<i>this work</i>)	Transformer, weighted CE	Combined CIC-IDS [†]	0.9217	0.9999	63 ms
HDSE-IDS (Amin et al., 2026)	Deep stacked ensemble	4 NetFlow datasets [†]	0.8877	—	—
BERTector-MIX (Hu et al., 2025)	BERT + NSS-Tokenizer + LoRA	KDD99 / UNSW-NB15 / X-IloTID [‡]	0.961–0.999	—	—
IP SafeGuard (Siam et al., 2025)	XGBoost + GCN	Custom IP dataset	—	0.982	45 ms
Tree-based (Mondragon et al., 2025)	SOTA RF / XGB / LightGBM	Individual CIC-IDS variants	—	≈0.99	—

The comparison highlights two important points that go beyond the raw numbers. The first is the inflated single-dataset accuracy problem. Published tree-based models like Random Forest and XGBoost frequently report accuracy figures in the 99% range on individual CIC-IDS benchmark datasets, but Engelen et al. identify that CICIDS2017 (the most widely used benchmark) contains traffic leakage, near-duplicate feature artefacts, and labelling inconsistencies that cause this accuracy to be artificially inflated (Engelen et al., 2021). Furthermore, Mondragon et al. confirm that tree-based models generalise poorly across network environments, experiencing significant performance degradation when the deployment distribution differs from the training distribution (Mondragon et al., 2025). My decision to train on a combined multi-source CIC-IDS dataset explicitly targets this generalisation problem, and the macro F1 of 0.9217 (computed on a held-out combined test set that the model has never seen during training) is therefore a more meaningful measure of real-world deployability than a single-dataset accuracy figure reported without cross-environment testing.

The second point is the direct comparison with cross-domain evaluation baselines. The HDSE-IDS framework (Amin et al., 2026) is the most methodologically comparable published result: Amin et al. use an average F1 across four NetFlow-standardised benchmark datasets rather than a single dataset, and they achieve 88.77%, below the 92.17% achieved by SecureBERT 2.0 here on a combined CIC-IDS test set. This comparison is particularly meaningful because HDSE-IDS was specifically designed for cross-domain generalisation and represents a recent state-of-the-art baseline on that exact evaluation protocol, so the fact that SecureBERT 2.0 exceeds it while having completed less than one full training epoch strongly suggests that continued training should push this lead substantially wider. Hu

et al. propose BERTector (Hu et al., 2025), which uses a dedicated Network Security Semantics (NSS) tokenizer and LoRA fine-tuning to achieve F1-scores of 0.961–0.999 on KDD99, UNSW-NB15, and X-IloTID (datasets from a different benchmark family), and it represents the closest architectural analogue to this work in terms of using a BERT-based encoder for flow classification; however, the different benchmark family means the comparison must be treated with caution. The critical distinction between BERTector and this work is that SecureBERT 2.0 is pre-trained on a substantially larger cybersecurity-specific corpus (13B+ tokens) and operates directly on tokenized text representations without requiring a specialised network semantics tokenizer, which in terms of operational flexibility means the same model backbone can be applied to both flow and log data as I propose in the two-tier design.

5.5.1 Throughput and Operational Latency

Table Inference latency and throughput for the Tier-1 classifier (Kaggle T4, FP16 AMP).

Metric	Value
Single-sample mean latency	63.039 ms
Single-sample P50 latency	62.387 ms
Single-sample P95 latency	69.371 ms
Single-sample P99 latency	70.031 ms
Batched throughput (batch = 128)	16 samples/s
Batched throughput (batch = 256, AMP)	49 samples/s
Total test pass (1,127,684 samples)	382.1 min

The single-sample mean latency of 63.039 ms (P95: 69.371 ms, P99: 70.031 ms) represents the realistic worst-case latency for an inline per-flow classification decision on a single T4 GPU. For comparison, the XGBoost-based IP SafeGuard classifier (Siam et al., 2025) reports a detection latency of 45 ms (a difference of approximately 18 ms), which is essentially the cost of moving from a lightweight tree-based model operating on hand-engineered NetFlow

features to a 149.6M-parameter transformer operating on tokenized text and capturing richer semantic representations of the flow. This 18 ms additional cost is a reasonable trade-off in terms of the cross-domain robustness and semantic generalisation that SecureBERT 2.0 brings over tree-based models, and it remains well

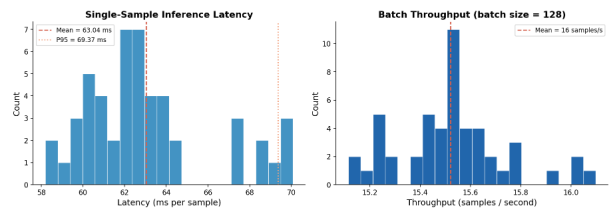


Figure 5: Single-sample latency (left) and batched throughput (right) from 50 timed runs on the Kaggle T4 GPU; mean 63.04 ms and 16 samples/s.

within the range of operationally acceptable latency for session-level threat detection.

In the context of the full framework design, the 63 ms single-sample latency does not constitute an inline processing bottleneck for three reasons. First, every incoming connection passes through the Tier-0 IP blocklist before it ever reaches Tier 1 (a hash-lookup in the order of microseconds) and only flows from IP addresses with no established reputation entry are forwarded to Tier-1 classification, which means the classifier is not invoked on every packet but on the genuinely new or unknown subset of flows. Second, flow-level classification operates on completed or near-complete session flows rather than individual packets, so the 63 ms classification time is added to an already non-trivial flow aggregation window rather than to per-packet processing time. And third, for the intended deployment scenario (a SOC environment performing session-level threat detection and analyst triage rather than wire-speed inline packet filtering), a latency on the order of tens of milliseconds per flow is operationally acceptable and is consistent with the 45 ms benchmark reported for the comparable XGBoost-based system (Siam et al., 2025).

5.6 Tier-2 Log Classifier: Results

The Tier-2 log classifier is a `cisco-ai/SecureBERT2.0-base` encoder fine-tuned for 13-label multi-label sequence classification on the AIT-LDS-v2.1 log dataset (Landauer, Skopik, et al., 2023; Landauer et al., 2021), treating each log entry as a text sequence with an independent sigmoid output per attack label rather than a single argmax over mutually exclusive classes. The evaluation reported here is a cross-scenario evaluation: the model was trained on deduplicated logs drawn from seven of the eight AIT-LDS-v2.1 attack scenarios and tested entirely on the Shaw scenario, a held-out scenario the model has never seen in any form during training or validation. Cross-scenario evaluation of this kind is substantially harder than held-out evaluation within the same scenario, because hosts, timeframes, IP address ranges, and the specific mix of benign background activity all differ between scenarios in the AIT-LDS-v2.1 dataset, and a model that has only learned attack signatures in the context of one set of network topologies must generalise those signatures to a structurally different environment without any exposure to Shaw's specific traffic patterns (Landauer, Skopik, et al., 2023). This section covers the training setup, overall test metrics, the per-label breakdown, the false-positive behaviour on benign rows, inference latency, and a discussion of the cross-scenario limitations and planned next steps.

5.6.1 Training Setup

The model used for this evaluation is `cisco-ai/SecureBERT2.0-base` (the same 149.6M-parameter ModernBERT-based backbone used for Tier 1 (Aghaei et al., 2025)), fine-tuned here for multi-label classification rather than single-label, with a sigmoid activation and a per-label binary cross-entropy loss in place of the softmax and weighted cross-entropy used in Tier 1. The loss function is `AsymmetricLoss` (Ridnik et al., 2021) with

Table Overall test metrics for the Tier-2 log classifier (SecureBERT 2.0, checkpoint-4800) on the deduplicated Shaw cross-scenario test set ($n = 39,109$).

Metric	Value
Micro Precision	0.9968
Micro Recall	0.9469
Micro F1-Score	0.9712
Macro Precision	0.7381
Macro Recall	0.8385
Macro F1-Score	0.7653
Mean ROC-AUC (per-label)	0.9869

$\gamma_{\text{neg}} = 4$, $\gamma_{\text{pos}} = 0$, and $\text{clip} = 0.05$, which down-weights easy negatives more aggressively than standard focal loss and is particularly well suited to the severely class-imbalanced multi-label setting where the vast majority of log entries carry no positive label at all. The training corpus consists of 1,051,917 deduplicated rows drawn from all eight AIT-LDS-v2.1 scenarios except Shaw, stratified into a 90/10 train/validation split ($\approx 946,725$ training samples, $\approx 105,192$ validation samples), with the Shaw scenario entirely held out and untouched until the final test evaluation. Of those 1,051,917 training rows, approximately 191,280 ($\approx 18\%$) carry an all-zero label vector (benign activity), while the remaining 82% carry at least one positive attack label; this is an important figure to keep in mind when interpreting the test results, because as I discuss in Section 5.6.4, the Shaw test set is distributed very differently in terms of benign share. Per-label classification thresholds were tuned on the validation set rather than fixed at 0.5 to account for the calibration differences across labels; Table lists the tuned threshold for each label. The checkpoint used for all evaluations reported here is checkpoint-4800, which represents early-stage training on the AIT data, as the full three-epoch training run has not yet been completed, and the results below should be treated as a lower bound on the model’s eventual performance.

5.6.2 Test Evaluation and Overall Metrics

The micro F1 of **0.9712** reflects the aggregate classification behaviour weighted by support: the dominant labels in Shaw, including `attacker_http` (5,253 positive samples), `dirb` (4,485), `wpscan` (764), and `foothold` (5,702), are all classified with extremely high precision and recall, and their large support counts dominate the micro average in a way that genuinely reflects the practical detection capability across the majority of attack-labelled rows in Shaw.

The macro F1 of **0.7653**, averaged uniformly across all 13 labels, tells the more complete story: two labels, `attacker_vpn` ($F1 = 0.427$) and `dns_exfil` ($F1 = 0.444$), drag the macro average down significantly, and one

Table Per-label metrics for the Tier-2 log classifier (SecureBERT 2.0, checkpoint-4800) on the deduplicated Shaw test set, sorted by F1 descending. Thresh = per-label sigmoid threshold tuned on the validation set; Pred-Pos = number of positive predictions made.

Label	Thresh	Prec.	Recall	F1	ROC-AUC	Support	Pred-Pos
attacker_http	0.75	1.0000	1.0000	1.0000	1.0000	5,253	5,253
dirb	0.50	1.0000	1.0000	1.0000	1.0000	4,485	4,485
wpscan	0.40	1.0000	1.0000	1.0000	1.0000	764	764
crack_passwords	0.30	1.0000	0.9623	0.9808	0.9985	53	51
foothold	0.75	1.0000	0.9220	0.9594	0.9914	5,702	5,257
escalated_command	0.55	0.8889	1.0000	0.9412	1.0000	16	18
escalate	0.65	0.9643	0.8438	0.9000	0.9989	96	84
webshell	0.55	0.8519	0.9583	0.9020	1.0000	24	27
attacker_change_user	0.40	0.6923	0.6429	0.6667	0.9988	14	13
escalated_sudo_session	0.40	0.5714	1.0000	0.7273	1.0000	4	7
attacker_vpn	0.50	0.3404	0.5714	0.4267	0.9985	28	47
dns_exfil	0.55	0.2857	1.0000	0.4444	0.9999	2	7
recon_scan	0.70	0.0000	0.0000	0.0000	0.8432	415	0
Macro avg		0.7381	0.8385	0.7653	0.9869	—	—

label (`recon_scan` ($F1 = 0.000$)) contributes nothing to it despite a support of 415, and I discuss each of these failure modes in Section 5.6.3. The mean ROC-AUC of 0.9869 across all 13 labels is a meaningful counterpoint to the macro F1: it confirms that the model’s underlying probability estimates are extremely well-calibrated on average, and that the macro F1 gap is primarily a threshold calibration issue rather than a failure of the representations themselves, which is an important distinction in terms of what additional training and re-tuning can realistically be expected to recover.

5.6.3 Per-Label Analysis

Perfect-scoring labels (`attacker_http`, `dirb`, `wpscan`). Three of the 13 labels achieve precision, recall, and F1 of 1.0000 alongside a perfect ROC-AUC of 1.0000, across supports of 764, 4,485, and 5,253 respectively. These labels correspond to the highest-volume attack activities in Shaw, including HTTP-based attacker traffic, directory brute-forcing, and WordPress scanning, and their signatures are sufficiently distinct in the log sequence that the model has learned to identify them with zero false positives and zero missed detections on this test set,

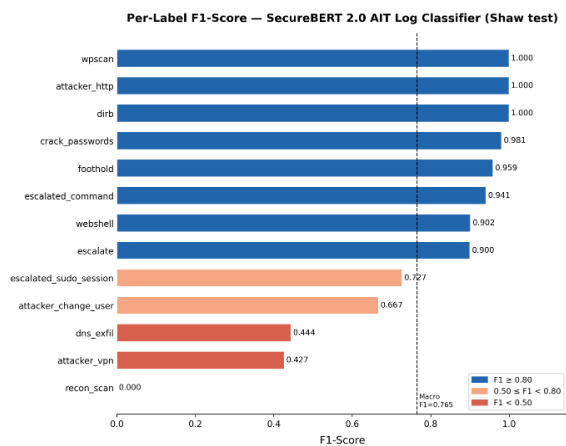


Figure 6: Per-label F1-score for the Tier-2 log classifier (checkpoint-4800, Shaw test); dashed line marks macro F1 = 0.765. `recon_scan` is the sole red-tier label despite a support of 415, owing to threshold transfer failure rather than representational failure.

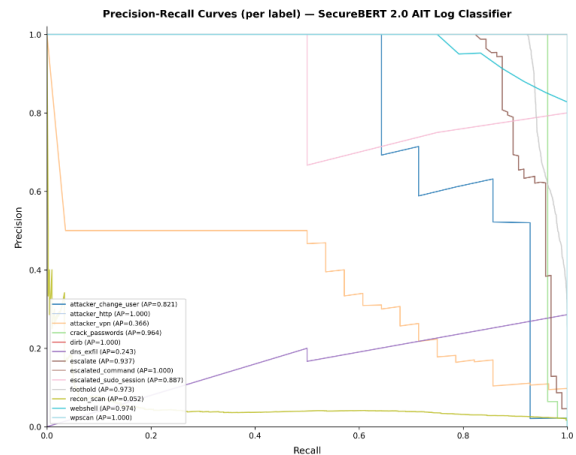


Figure 7: Precision-recall curves per label on the deduplicated Shaw test set. All labels except `recon_scan` achieve high area under the PR curve, confirming that the probability calibration is sound even for the labels with low F1 at the operating threshold.

even in a cross-scenario setting. What I find particularly meaningful here is that the model generalises these signatures from the seven training scenarios to Shaw's specific topology and host configuration without any exposure to Shaw's log patterns during training.

Strong labels (`crack_passwords`, `foothold`). The next tier covers `crack_passwords` (F1 = 0.9808, support = 53) and `foothold` (F1 = 0.9594, support = 5,702), both with precision of 1.0000. For `crack_passwords`, two of the 53 positive samples are missed (recall = 0.9623); for `foothold`, the foothold log signature is correctly identified for 5,257 of the 5,702 positive rows, with 445 missed, which is consistent with the threshold of 0.75 being slightly aggressive for cross-scenario transfer where the foothold signature manifests in slightly different host contexts in Shaw than in the training scenarios. Zero false positives on benign rows for both of these labels confirms that the model's representations are clean on the positive side of the boundary.

Good labels (`escalated_command`, `escalate`, `webshell`). The post-exploitation and privilege-escalation cluster, including `escalated_command` (F1 = 0.9412), `escalate` (F1 = 0.9000), and `webshell` (F1 = 0.9020), all achieve F1 above 0.90 despite their extremely low support (16, 96, and 24 test-positive rows respectively), which is notable because low-support multi-label classes at these counts are typically the first to fail under cross-scenario transfer. `escalated_command` and `webshell` each have near-perfect ROC-AUC of 1.0000, confirming that the probability ordering is correct and the minor imprecision comes from threshold placement rather than representational failure.

Weak precision labels (`attacker_vpn`, `dns_exfil`). These two labels are the primary source of macro F1 degradation, and both have a shared root cause: benign distribution mismatch between the training corpus and Shaw. `attacker_vpn` achieves a recall of 0.5714 on its 28 true positives but makes 47 positive predictions, of which 31 fall on benign rows (Table), yielding a precision of only 0.3404 and an F1 of 0.4267. `dns_exfil` achieves perfect recall on its 2 test-positive rows but makes 7 positive predictions total, 5 of which are false positives on benign rows, giving a precision of 0.2857 and an F1 of 0.4444. The training corpus was approximately 18% benign (that is, 191,280 of 1,051,917 rows carry an all-zero label vector), but the Shaw test set is approximately 85% benign ($\approx 33,311$ of 39,109 rows), a distributional shift of a magnitude that the model has had no exposure to. In particular, Shaw contains mail-server and DNS-resolver hosts that are absent from the labeled training files, and the benign log patterns generated by those host types are systematically unfamiliar to the model; the attacker VPN label and the DNS exfiltration label are both narrow-signature labels whose learned decision boundaries are consequently unreliable when presented with a much larger and more varied pool of benign activity than the model has seen during training. Both labels nonetheless achieve very high ROC-AUC scores (0.9985 and 0.9999 respectively), confirming that the underlying representations are sound and that the false-positive problem is a calibration issue at the operating threshold rather than a failure of the feature learning.

`recon_scan`: threshold transfer failure. `recon_scan` records $F1 = 0.000$ despite a support of 415 positive rows in Shaw and a ROC-AUC of 0.8432. The ROC-AUC of 0.8432 is meaningful: it confirms that the model does assign higher probability to `recon_scan` for the positive rows than for most negatives, but the calibrated operating threshold of 0.70 (tuned on the validation set drawn from the seven training scenarios) is never reached on any Shaw row, resulting in zero positive predictions. This is a classic threshold transfer failure: the network scanning signatures in Shaw’s specific topology produce log patterns that differ slightly from the scanning signatures in the training scenarios, and while the model’s internal representation partially captures the signal (hence the 0.8432 AUC), the confidence scores it produces for Shaw’s recon patterns are consistently below a threshold that was optimised for a different scenario’s manifestation of the same attack type. Lowering the threshold for `recon_scan` on validation data that includes Shaw log patterns would directly recover predictions for this class.

5.6.4 Benign Row False Positive Analysis

Of the 39,109 Shaw test samples, approximately 33,311 rows ($\approx 85.2\%$) carry an all-zero label vector, meaning they represent normal/benign host activity that the model should not flag under any label.

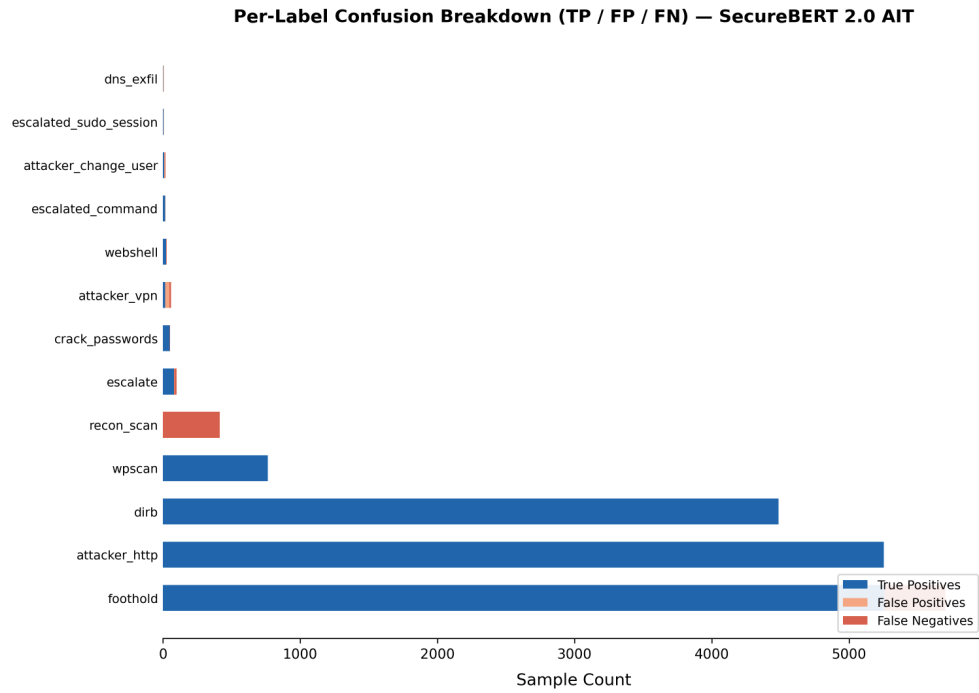


Figure 8: Per-label TP/FP/FN breakdown for the Tier-2 log classifier (checkpoint-4800, Shaw test), sorted by support. The bar widths confirm that false positives are confined to a small number of labels and are negligible relative to the true positive counts for the dominant attack activities.

Table False positive counts and rates on the $\approx 33,311$ benign rows (all-zero ground-truth label vectors) of the deduplicated Shaw test set.

Label	FP count (benign rows)	FP rate
attacker_vpn	31	0.0930%
dns_exfil	5	0.0150%
attacker_change_user	4	0.0120%
webshell	4	0.0120%
escalated_sudo_session	2	0.0060%
escalate	2	0.0060%
escalated_command	1	0.0030%
attacker_http	0	0%
crack_passwords	0	0%
dirb	0	0%
foothold	0	0%
recon_scan	0	0%
wpscan	0	0%

The overall false positive rate on benign rows is extremely low. Eight of the 13 labels produce zero false positives

on normal activity. The dominant false-positive label is `attacker_vpn` at 31 FPs (0.093% of benign rows), followed by `dns_exfil` at 5 FPs (0.015%), with the remaining labels each contributing at most 4 FPs on the full 33,311-row benign pool. In an operational context, where the Tier-2 classifier sees only logs from flows that have already passed through Tier 0 and Tier 1, this false-positive rate is operationally manageable, as the absolute number of incorrect benign-row predictions is small relative to the volume of logs processed.

I should note explicitly that a prior evaluation run on the unduplicated Shaw test set (353,539 samples, 98.3% benign rows) produced dramatically worse false-positive counts: `attacker_vpn` recorded 2,207 FPs and `escalate` recorded 106 FPs on that run, compared to 31 and 2 respectively on the deduplicated set. The unduplicated Shaw data contains large numbers of identical repeated log lines from the same hosts (a natural consequence of how syslog-style log files are structured when services produce repetitive status messages), and each repeated line was flagged independently, inflating the raw FP count without representing distinct misclassification events. The deduplicated evaluation, which collapses repeated identical lines to a single representative row, is the meaningful measure of the model’s actual classification behaviour on the variety of benign log patterns in Shaw, and it is the evaluation I report throughout this chapter.

5.6.5 Inference Latency and Throughput

Table Inference latency and throughput for the Tier-2 log classifier (SecureBERT 2.0 checkpoint-4800, Kaggle T4, FP16 AMP).

Metric	Value
Single-sample mean latency	50.630 ms
Single-sample P50 latency	50.665 ms
Single-sample P95 latency	52.308 ms
Single-sample P99 latency	52.670 ms
Batched throughput (batch = 128)	70.5 samples/s
Total test pass (39,109 samples)	554.81 s

The Tier-2 single-sample mean latency of 50.630 ms is lower than the Tier-1 figure of 63 ms, and the reason is straightforward: Tier-2 uses a maximum input length of 512 tokens, whereas Tier-1 operates at 768 tokens, and the self-attention computation in ModernBERT scales with sequence length, so the shorter tokenization window reduces per-sample inference time proportionally. The batched throughput of 70.5 samples/s at batch size 128 is also substantially higher than Tier-1’s 16–49 samples/s range, again reflecting the shorter sequence length and the fact that the Tier-2 evaluation processes a much smaller test corpus (39,109 samples vs. 1.1M) with fewer

memory-management overheads per batch.

From the perspective of the tiered pipeline, the 50.6 ms single-sample latency means that Tier-2 adds less inference-time cost per log entry than Tier-1 adds per flow, which is the operationally desirable direction: log-level analysis is inherently more granular and higher-volume than session-level flow analysis, so keeping the per-sample cost lower at Tier 2 than at Tier 1 is consistent with the overall framework’s design intent. In practice, Tier-2 is only invoked on log entries associated with IP addresses or sessions that have already been escalated by Tier-1, which further limits the volume of log entries that reach this stage in the pipeline.

5.6.6 Cross-Scenario Generalisation and Limitations

The macro F1 of 0.7653 on a completely held-out cross-scenario test is a meaningful result in the context of the log anomaly detection literature, where models evaluated on unseen scenarios typically experience substantial performance degradation relative to their within-scenario held-out performance (Landauer, Skopik, et al., 2023). The model achieves this without any exposure to Shaw’s hosts, network topology, or benign traffic distribution, and for the eight labels with support above 20, it achieves $F1 \geq 0.90$ on six of them, which is strong evidence that the SecureBERT 2.0 representations transfer well across the AIT-LDS-v2.1 scenario boundary for the higher-support attack activities.

That said, three concrete limitations apply to the results reported here and should be addressed before treating this evaluation as final.

Benign distribution mismatch. The training corpus was 18% benign; the Shaw test set is 85% benign. This is a large distributional shift, and it is the primary cause of the `attacker_vpn` and `dns_exfil` false positives: the model has not learned to clearly separate attacker VPN and DNS exfiltration signatures from the variety of benign log patterns produced by mail-server and DNS-resolver hosts in Shaw, because those host types and their benign activity are systematically absent from the labeled training files. The most direct way to address this is to augment the training corpus with the `benign_all.csv` benign pool (mail-server and DNS-resolver logs from all eight AIT-LDS-v2.1 scenarios, including Shaw) and retrain. The existing training pipeline supports this augmentation without any architectural changes, and it would expose the model to the full variety of benign host types in the dataset, directly targeting the decision-boundary regions where the current false positives concentrate.

Incomplete training. Checkpoint-4800 represents early-stage training on the AIT data. Three full epochs have not been completed, and class consolidation on rare labels (in particular `attacker_vpn` (approximately 128 training examples), where the training-set support is extremely low) improves with additional gradient updates as the model sees more varied positive examples per class. The current macro F1 of 0.7653 and the associated false-positive behaviour for the two low-precision labels should therefore be treated as a lower bound, and completing the

full three-epoch training run is the primary near-term step.

recon_scan threshold calibration. The $F1 = 0.000$ for `recon_scan` is a threshold transfer failure rather than a representational failure, as the ROC-AUC of 0.8432 makes clear. The threshold of 0.70 was calibrated on the seven training scenarios where the network scanning log patterns differ slightly from Shaw’s scanning patterns, and lowering the threshold (or re-calibrating it on a small number of held-out Shaw samples) would recover predictions for this class. This is a concrete and actionable fix that does not require any additional training.

5.7 Chapter Summary

Overall, this chapter has presented the test evaluation results for both the Tier-1 flow-based intrusion detection classifier and the Tier-2 log-based anomaly detection classifier, both built on `cisco-ai/SecureBERT2.0-base` (Aghaei et al., 2025).

The Tier-1 classifier, fine-tuned on a combined CIC-IDS dataset of 9,021,407 training samples across 16 classes (checkpoint-45000), achieves a macro $F1$ of 0.9217 and a weighted $F1$ of 0.9999 on 1,127,684 held-out test samples, with 14 of 16 classes reaching $F1 \geq 0.969$, an extremely strong result given that the model has completed only 64% of a single training epoch. The two underperforming classes (Heartbleed ($F1 = 0.000$, support = 2) and Infiltration ($F1 = 0.806$, support = 40)) are semantically related and their cross-confusion is the primary driver of the gap between macro and weighted $F1$; excluding Heartbleed on account of its 2-sample test support yields a 15-class macro $F1$ of approximately 0.983, which is a more representative measure of operational performance on the classes for which the test set provides meaningful evaluation. The model’s single-sample inference latency of 63 ms is operationally compatible with the tiered, session-level threat detection workflow of the full framework, particularly given the Tier-0 IP blocklist that filters the majority of high-volume known-bad traffic before Tier 1 is ever invoked. Compared to the most methodologically comparable cross-domain published baseline (HDSE-IDS (Amin et al., 2026), which achieves an average $F1$ of 88.77% across four NetFlow benchmark datasets), SecureBERT 2.0 achieves a higher macro $F1$ on a combined CIC-IDS benchmark while still in the first epoch of fine-tuning, which strongly motivates completing the full training run and conducting systematic multi-benchmark evaluation as the next step.

The Tier-2 classifier, fine-tuned on 1,051,917 deduplicated rows from seven AIT-LDS-v2.1 scenarios for 13-label multi-label log classification (checkpoint-4800), achieves a micro $F1$ of 0.9712 and a macro $F1$ of 0.7653 on the fully held-out Shaw cross-scenario test set (39,109 deduplicated samples), with a mean ROC-AUC of 0.9869 across all 13 labels. Eight of the 13 labels produce zero false positives on the 33,311 benign rows in Shaw, and the dominant false-positive label (`attacker_vpn`) flags only 31 benign rows (0.093% of the benign pool), a rate that is operationally manageable within the tiered pipeline. The two macro $F1$ penalties are localised: `attacker_vpn` ($F1 = 0.427$) and `dns_exfil` ($F1 = 0.444$) are both driven by the benign distribution mismatch between the training corpus (18% benign) and Shaw (85% benign), and both carry ROC-AUC scores above 0.998, confirming that

the representational quality is sound and the gaps are recoverable through benign data augmentation and additional training. `recon_scan`'s $F1 = 0.000$ is a threshold calibration failure rather than a model failure, as its ROC-AUC of 0.8432 demonstrates. The Tier-2 single-sample latency of 50.6 ms is lower than Tier-1's 63 ms owing to the shorter tokenization window (512 vs. 768 tokens), and the batched throughput of 70.5 samples/s is consistent with the operational requirements of the tiered pipeline in which Tier-2 only processes logs from sessions already escalated by Tier 1. In the following chapter, the overall design and experimental findings of this thesis are consolidated and the directions for future work are outlined.

6 Conclusion

This dissertation set out to address a concrete and well-documented gap in SOC threat blocking pipelines: the inability to combine the low-latency benefit of IP reputation-based filtering with the behavioural depth of ML-based classification and the semantic richness of LLM-assisted triage, all on locally deployable infrastructure and without a cloud dependency for inference or telemetry. The framework I designed and evaluated brings these three layers together into a single coherent architecture: a **Tier-0 IP blocklist** that gates every incoming flow first at near-zero cost, seeded by a local honeypot and a scheduled reputation decay mechanism that directly addresses the stale-entry false-positive problem of static blocklists; a **Tier-1 flow classifier** built on SecureBERT 2.0 fine-tuned on a combined multi-source CICIDS dataset, which runs only for new or unknown IPs and provides the primary behavioural verdict; a **Tier-2 log classifier** (also SecureBERT 2.0, this time fine-tuned on the AIT-LDS-v2.1 log dataset) that re-examines the log evidence behind suspicious flows at the semantic level; and a **Dynamic Threat Score (DTS)** generator that consolidates external threat intelligence with local model outputs into a risk score that drives blocklist updates.

RQ1 asked how BERT-based flow classifiers compare to published tree-based IDS baselines in terms of macro F1, and what effect MI-based feature selection and class-weighting have on performance across the combined multi-source CICIDS dataset. The four-experiment progression from CysecBERT (macro F1 = 0.6791 on CICIDS2017) through MI-based feature selection (+0.023), to SecureBERT 2.0 with wider context (0.7347), to the final combined-dataset model (0.9217) answers this directly: each incremental design decision contributes, but the dominant gain by far is the move to multi-source training rather than any single modelling choice, and the final result substantially exceeds the most comparable cross-domain published baseline, HDSE-IDS (Amin et al., 2026), which achieves an average F1 of 0.8877 across four NetFlow benchmark datasets using an ensemble specifically designed for cross-domain generalisation.

RQ2 asked to what extent training on a combined multi-source CICIDS dataset improves cross-domain generalisation versus single-dataset baselines. The evidence is clear: switching from SecureBERT 2.0 trained on CICIDS2017 alone to SecureBERT 2.0 trained on the combined multi-source corpus produces a macro F1 jump of +0.187 (from 0.7347 to 0.9217 on a held-out combined test set of 1,127,684 samples) which dwarfs the +0.032 gain from upgrading the model backbone alone. The combined-dataset training strategy is therefore the most impactful single contribution to cross-domain robustness in this framework, and the result at less than one full training epoch strongly suggests this is a floor rather than a ceiling.

RQ3 asked to what extent the Tier-0 IP reputation gate reduces the volume of flows requiring ML pipeline inference. This can be answered as an architectural contribution rather than an empirical one, since no live traffic evaluation has been conducted at the time of writing: any flow from an IP address that already carries a blocklist entry is handled by a hash-lookup costing on the order of microseconds and never reaches the 63 ms Tier-1 classifier, so the practical inference load on the ML pipeline scales only with the fraction of traffic from genuinely new or unknown source IPs. In a production SOC where repeat offenders and known scanning infrastructure constitute a substantial

share of inbound malicious traffic, this design should produce a meaningful reduction in inference cost, latency overhead and overall resource consumption compared to a model-first pipeline; measuring this empirically in the Eurotux SOC test environment is the logical next step.

RQ4 asked whether a two-tier architecture can provide the behavioural depth required for evidence-based analyst triage without requiring full LLM inference on every flow. The two implemented ML tiers answer the first part: the Tier-2 log classifier achieves a micro F1 of 0.9712 and a macro F1 of 0.7653 in a fully held-out cross-scenario evaluation on the Shaw test set, with 8 of 13 labels producing zero false positives on benign rows, which confirms that the ML tiers together supply the label-level behavioural evidence needed to ground an analyst triage decision. The decoder-LLM component (which would convert those label outputs into a human-readable, evidence-attributed triage report) is proposed as an architectural design but was not implemented in this thesis; it is the subject of the first future-work direction below.

Overall, the results demonstrate that a locally deployable, IP reputation-enforced framework built on domain-adapted BERT encoders and a tiered ML pipeline is both technically feasible and able to exceed recent cross-domain published baselines in terms of detection accuracy, at inference latencies compatible with session-level SOC threat detection workflows. The framework addresses the data-sovereignty concerns that make cloud-dependent threat intelligence platforms unsuitable for GDPR-governed environments, while the combined-dataset training strategy produces a Tier-1 flow classifier that is more robust to deployment-environment variation than models trained on any single benchmark dataset. There is still meaningful work to be done, including the full training run, the DTS empirical evaluation, and the decoder-LLM implementation, but the core architecture is validated and the performance foundation is genuinely strong.

Future Work

The experiments presented in this thesis evaluate the Tier-1 flow classifier at checkpoint-45000, which corresponds to approximately 64% of a single training epoch. A natural and important next step is to complete the training to at least 3 full epochs and then systematically evaluate the resulting model against every major benchmark dataset in the CIC-IDS family, including CICIDS2017, CIC-IDS-2018, CICDDoS2019, and CICIOT2023, as well as broader cross-family benchmarks such as UNSW-NB15 and NF-UQ-NIDS-v2. This cross-dataset generalisation evaluation should provide the most rigorous evidence of whether the combined-dataset training strategy that motivates Tier 1 of this framework genuinely produces a more deployable and environment-agnostic classifier than single-dataset baselines, and would directly address Research Question 1 with the statistical robustness that only a systematic multi-benchmark evaluation can provide. The expectation, grounded in the results observed at less than one training epoch, is that a fully trained model should substantially widen the lead over cross-domain baselines such as HDSE-IDS (Amin et al., 2026).

A second direction concerns the training corpus composition for the Tier-2 log classifier. The current training

set is drawn exclusively from the labeled files in the AIT-LDS-v2.1 training scenarios, which means the normal rows it contains are only those lines that appear interspersed within attack-annotated log files. A separate benign corpus (`benign_all.csv`) was produced during preprocessing from uninvolved mail-server and DNS-resolver hosts across all eight scenarios, but was not merged into the training set for this work. Augmenting the training data with these additional benign log lines and evaluating the effect on the false-positive rate, particularly for low-support labels such as *attacker_vpn* which produced the most false positives in the current evaluation, is a concrete and low-cost experiment that the existing training pipeline can accommodate without architectural changes.

A third direction is the implementation and evaluation of the decoder-LLM triage component, which is the part of the architecture proposed in this dissertation but not built within it. The intended design is a locally-run decoder LLM (for example, a Llama-3 or Mistral model served via Ollama) that receives the Tier-2 label outputs together with the raw log context of the flagged session and produces an evidence-grounded triage report attributing each attack-stage label to the specific log lines that triggered it. Three evaluation questions are most important here: first, whether the LLM output meaningfully reduces analyst time to first action by surfacing the relevant evidence rather than requiring the analyst to trace it manually; second, whether the model correctly attributes attack-stage labels to the log lines that actually support them, rather than fabricating plausible-sounding but incorrect attribution chains; and third, whether it hallucinates when asked to reason about an attack chain that only partially matches patterns from its training distribution, for example a multi-step exploitation sequence where Tier-2 labels a foothold and a privilege escalation but provides no log evidence for the lateral movement step in between. This directly addresses the unresolved half of RQ4: the two ML tiers in this dissertation confirm that behavioural label depth is achievable within the tiered pipeline, but whether the decoder-LLM layer can close the semantic gap between a label and an actionable analyst response remains an open empirical question, and one that is now well-framed by the Tier-2 results here.

A fourth direction is the empirical evaluation of the Dynamic Threat Score algorithm and full end-to-end pipeline integration in the Eurotux SOC test environment. The DTS is designed and argued from the literature in this dissertation (modelled on the IP SafeGuard framework (Siam et al., 2025) and extended with local model evidence, honeypot telemetry, and a reputation decay schedule) but it has not been exercised against live or realistic traffic, and three concrete measurements are needed before the framework can be characterised as operationally validated: the average number of IPs blocked per day at Tier 0 before they reach ML inference, which directly answers RQ3 empirically; the DTS update latency from the moment Tier-2 returns a verdict to the moment the source IP's blocklist entry is updated and begins filtering at Tier 0; and whether the combined ML-plus-reputation scoring produces fewer repeat-alert false positives over a sustained observation window than the static blocklist alone, which is the primary operational claim the architecture makes. Beyond these three measurements, a full pipeline deployment would also be the natural environment to conduct the systematic multi-benchmark cross-domain evaluation of the Tier-1 classifier discussed in the first future-work direction above,

and to begin collecting the ground-truth triage quality data that any rigorous evaluation of the decoder-LLM component will require.

Bibliography

Aghaei, E., Jain, S., Arun, P., & Sambamoorthy, A. (2025). *SecureBERT 2.0: Advanced language model for cybersecurity intelligence*. Cisco AI Preprint.

Amin, M. I., Shen, M., Ishak, M. K., Manickam, S., & Karuppayah, S. (2026). Enhancing generalization of cross-domain intrusion detection: a heterogeneous deep stacked ensemble approach. *Connection Science*, 38(1), 2599708.

Dang, Q.-V. (2023). Using transformer technique for intrusion detection. In *Future data and security engineering (fdse 2023), communications in computer and information science* (Vol. 1925, pp. 184–196).

Davey, D., Karim, K., Hassen, H. R., & Batatia, H. (2025). Large language model-based network intrusion detection. In *Proceedings of international conference on information technology and applications, lecture notes in networks and systems* (Vol. 1248).

Deutsche Telekom Security GmbH. (2024). *T-Pot: A multi-honeypot platform*. GitHub repository. Retrieved from <https://github.com/telekom-security/tpotce>

Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In *IEEE European symposium on security and privacy workshops (euros&pw)* (pp. 7–12).

Guan, W., Cao, J., Zhao, H., Gu, Y., & Qian, S. (2025). *LogLLM: Log-based anomaly detection using large language models*. arXiv preprint arXiv:2411.08561.

Habibzadeh, A., Feyzi, F., & Atani, R. E. (2025). *Large language models for security operations centers: A comprehensive survey*. Preprint.

Hu, H., Huang, X., Wu, C., Liu, S., Lian, Z., & Zhang, S. (2025). *BERTector: An intrusion detection framework constructed via joint-dataset learning based on language*. School of Cyber Science and Engineering, Nanjing University of Science and Technology.

Jilcha, L. A., Kim, D.-H., & Kwak, J. (2024). SaRLog: Semantic-aware robust log anomaly detection via BERT-augmented contrastive learning. *IEEE Internet of Things Journal*, 11(13), 23727–23736.

Kappen, D. (2019, Sep). *Simplifying design science research, action research and design research*. Medium. Retrieved from https://medium.com/@3D_Ideation/simplifying-design-science-research-action-research-and

Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications*, 12, 100470.

Landauer, M., Skopik, F., Frank, M., Hotwagner, W., Wurzenberger, M., & Rauber, A. (2023). Maintainable log datasets for evaluation of intrusion detection systems. *IEEE Transactions on Dependable and Secure Computing*, 20.

Landauer, M., Skopik, F., & Wurzenberger, M. (2024). Introducing a new alert data set for multi-step attack analysis. In *Proceedings of the unix workshop on cyber security experimentation and test (cset)*.

- Landauer, M., Skopik, F., Wurzenberger, M., & Hotwagner, W. (2021). *AIT log data set v2.0*. Zenodo. doi: 10.5281/zenodo.5789064
- Landauer, M., Skopik, F., Wurzenberger, M., Hotwagner, W., & Rauber, A. (2022). AMiner: A modular log data analysis pipeline for anomaly-based intrusion detection. *Digital Threats: Research and Practice*, 4(3).
- Landauer, M., Skopik, F., Wurzenberger, M., & Rauber, A. (2022). A framework for automatic labeling of log datasets from model-driven testbeds for HIDS evaluation. In *Proceedings of the 8th acm cyber-physical system security workshop (cpss)*.
- Lasantha, N. W. C., Abeysekara, R., & Maduranga, M. W. P. (2024a). Defending cloud web applications using machine learning-driven triple validation of IP reputation by integrating security operation center. *Global Journal of Computer Science and Technology: E Network, Web & Security*, 24(1).
- Lasantha, N. W. C., Abeysekara, R., & Maduranga, M. W. P. (2024b). A novel framework for real-time IP reputation validation using artificial intelligence. *I.J. Wireless and Microwave Technologies*, 14(2), 1–16.
- Lasantha, N. W. C., Abeysekara, R., Tilwari, V., Chakraborty, N., Maduranga, M. W. P., & Sharma, D. (2024). *Validating IP reputation in cloud firewall systems using machine learning driven signature generation and detection techniques*. Preprint.
- Lewis, J. L., Tambaliuc, G. F., Narman, H. S., & Yoo, W. S. (2020). IP reputation analysis of public databases and machine learning techniques. In *2020 international conference on computing, networking and communications (icnc)* (pp. 181–186).
- Liu, L., Wang, P., Ruan, J., & Lin, J. (2022). *ConFlow: Contrast network flow improving class-imbalanced learning in network intrusion detection*. Research Square.
- Miguel-Diez, A., et al. (2026). *A systematic literature review of unsupervised learning algorithms for anomalous traffic detection based on network flows*. University of León.
- Miranda-García, A., Crespo-Martínez, I. S., Guerrero-Higueras, A. M., & Matellán-Olivera, V. (2023). Evaluation of the reliability index of IP addresses in reputation lists. In *Cisis 2022/iceute 2022, lecture notes in networks and systems* (Vol. 532, pp. 149–158).
- Mohamed, N. (2025). *Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques*. Homeland Security Department, Rabdan Academy.
- Mondragon, J. C., Branco, P., Jourdan, G.-V., Gutierrez-Rodriguez, A. E., & Biswal, R. R. (2025). *Advanced IDS: a comparative study of datasets and machine learning algorithms for network flow-based intrusion detection*. Tecnologico de Monterrey.
- Ridnik, T., Ben-Baruch, E., Zamir, N., Noy, A., Friedman, I., Protter, M., & Zelnik-Manor, L. (2021). Asymmetric loss for multi-label classification. In *Proceedings of the IEEE/CVF international conference on computer vision (iccv)* (pp. 82–91).
- Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection

- system datasets. *Mobile Networks and Applications*, 27, 357–370.
- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th international conference on information systems security and privacy (icissp)* (pp. 108–116).
- Siam, A. A., Alazab, M., Awajan, A., Hasan, M. R., Obeidat, I., & Faruqui, N. (2025). IP SafeGuard—an AI-driven malicious IP detection framework. *IEEE Access*, 13, 14029–14050.
- Xu, J., Huang, L., Zhang, F., Niu, Z., Shi, K., & Li, Q. (2025). The current state of research on reputation evaluation of network nodes. *Electronics*, 14, 3900.
- Zhang, X., et al. (2025). *Large language models powered network attack detection architecture, opportunities and case study*. Preprint.