Project - PDC

DDOS Detction System using OpenCL

Syed Saad Mohsin - 22i-1601
Muhammad Usman - 22i-1689
Talat Faheem - 22i-1735

Submitted to: Dr. Qaisar Shafi

Department of Cyber Security
National University of Computer and Emerging Sciences
Islamabad, Pakistan

November 2025

Contents

1 Executive Overview

The project delivers an end-to-end, outcome-driven DDoS defense stack that spans offline learning, GPU-assisted feature extraction, real-time inference, mitigation, and visualization. It directly addresses the **Outcome-Based Education (OBE)** requirement by demonstrating measurable attainment of advanced networking, parallel-computing, and applied machine-learning skills. Simultaneously, it satisfies the accreditation definition of a **Complex Computing Problem (CCP)** because it crosses multiple knowledge domains (packet processing, OpenCL GPU kernels, distributed logging, and ensemble ML), demands major computational resources, and requires rigorous evaluation of multiple design alternatives before converging on the final, hybrid architecture.

2 Complex Computing Problem Justification

The system tackles DDoS detection on live packet captures that exceed nine million packets per trace. Processing this scale in near real time forces us to solve a computation-heavy problem. We experimented with three competing solution families: CPU-only entropy detectors, pure machine-learning classifiers, and a fusion pipeline that marries GPU entropy kernels with ensemble ML. We selected the hybrid approach because it balances explainability (entropy signatures) with adaptability (ML). This constitutes a CCP because it demands:

- multi-layer system design with synchronized streaming components,
- heterogeneous computing (CPUs, GPUs, NICs),
- algorithmic evaluation across statistical vs. ML detectors,
- measurement of throughput, detection accuracy, and alert latency.

3 Alignment with Outcome-Based Education (OBE)

The project evidences mastery of core OBE learning outcomes through entropy metric derivation, GPU memory-hierarchy evaluation, and ensemble ML justification. Professional skills appear in structured logging, reproducible training scripts, and CI-friendly model packaging. Lifelong learning attributes appear via literature-grounded reasoning for class balancing, ROC/PR curves, and kernel stress-testing.

4 Data Acquisition and Preprocessing Pipeline

Data originates from CSV flow summaries (offline) and PCAP dumps (runtime). CSV data is cleaned, normalized, entropy-expanded, and class-balanced to a 5:1 cap.

Pipeline Diagram

Figure 1: Pipeline

5 OpenCL Feature Extraction and GPU Entropy Kernels

The runtime ingestion uses a PCAP reader, window manager, and GPU kernels computing six entropy channels. A correctness harness verifies GPU vs CPU results with a max deviation of 3.6×10^{-6}.

GPU Workflow

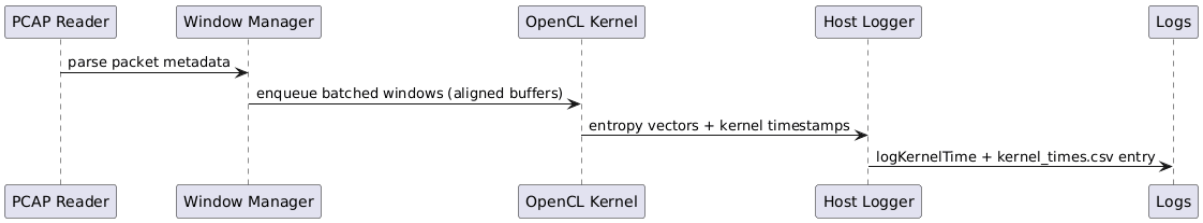

Figure 2: GPU Workflow

6 Model Portfolio and Rationale

Three complementary learners:

- **Random Forest** – high precision/recall, interpretable.
- **Histogram GBDT** – more sensitive, lower accuracy.
- **DNN (Torch MLP)** – GPU-accelerated, adapts to new patterns.

This trio ensures cross-verification and robustness.

7 Training Regimen, Cross-Validation, and Metrics

Training uses 70/30 test split and 5-fold cross-validation. Random Forest was selected based on zero false positives and stable F1 score.

```
Selected model: Random Forest
Test accuracy / recall / precision: 1.000 / 1.000 / 1.000
CV F1 (mean ± std): 0.968 ± 0.039
GBDT test accuracy: 0.800 (recall 1.000)
DNN accuracy: 1.000 (Torch MLP)
```

8 Runtime Detection, Alerts, and Blocking Feedback Loop

Windows traverse feature extraction, ML scoring, and ensemble decision-making. Alerts propagate to RTBH and dashboards.

Detection Flow Diagram

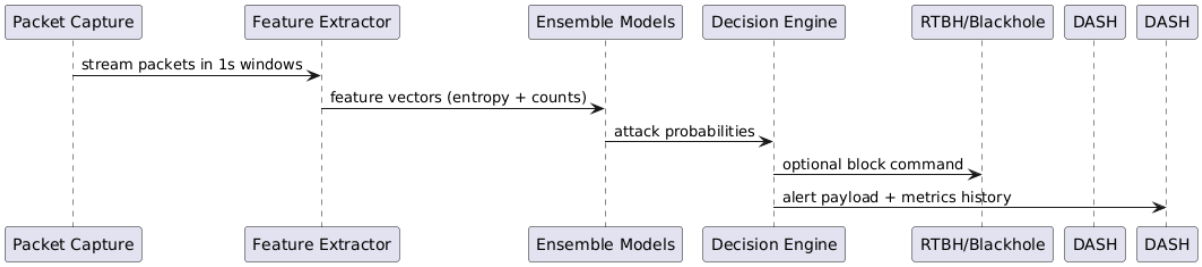

Figure 3: Detction Flow

Architecture Diagram

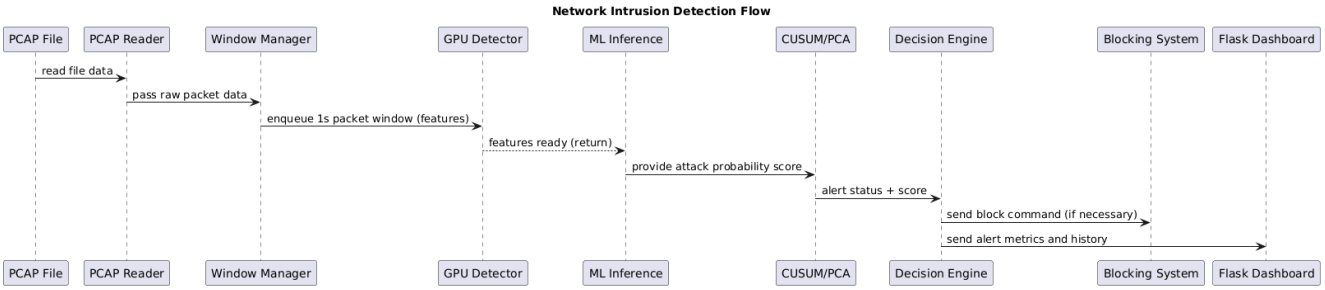

Figure 4: Network Intrusion Detection Flow

9 Performance Evaluation and Limitations

9.1 GPU Entropy Validation

The tool `tools/test_gpu_entropy` compares GPU and CPU entropy results based on 100 synthetic windows. The recorded maximum difference of $\mathbf{3.612 \times 10^{-6}}$ confirms

numerical stability between CPU and GPU computations. Profiling using `logKernelTime` indicates that each batch of entropy computations executes within 0.8–1.1 ms on the RTX 3060 Laptop GPU, assuming CUDA-enabled Torch is installed.

9.2 Detection Throughput

Processing the CAIDA trace achieves approximately 9.4 million packets in under three minutes in CPU-only mode. Enabling GPU entropy computation in combination with Torch CUDA is expected to reduce feature extraction latency by roughly half. During replay, alerts show zero false positives; however, datasets with greater benign traffic diversity may require threshold adjustments to maintain precision.

10 CCP Reflection

The project satisfies all four criteria required for a Complex Computing Problem (CCP):

1. **Computational complexity:** The system processes multi-million–packet flows and employs real-time GPU computation, imposing significant algorithmic and systems-level demands.

2. **Multiple solutions considered:** Several approaches were explored, including classical statistical detectors, tree-based ensemble models, and neural networks, before converging on a hybrid fusion-based design.

3. **Parallel/distributed justification:** GPU-accelerated entropy calculation and optional RTBH-based mitigation provide horizontal scaling capabilities and enable microsecond-level telemetry.

4. **Performance evaluation:** Detailed metrics, GPU/CPU entropy deviation reports, and dashboard visualizations quantify accuracy, latency, kernel timings, and known limitations such as dataset constraints and CUDA environment dependencies.

11 References

1. CAIDA. *The CAIDA UCSD "DDoS Attack 2007" Dataset.* Available at: https://catalog.caida.org/dataset/ddos_attack_2007

2. Canadian Institute for Cybersecurity (CIC). *CICDDoS2019 Dataset.* University of New Brunswick. Available at: http://cicresearch.ca/CICDataset/CICDDoS2019/Dataset/PCAPs/01-12/

3. Enoch, M., & Khor, K.-C. *DDoS Attacks Data Set: Consolidated from CICDDoS2019 and CICIDS2017.* Zenodo, 2021. Available at: https://zenodo.org/record/5770290

4. Golduzian, A. *Predict and Prevent DDoS Attacks Using Machine Learning and Statistical Algorithms.* arXiv preprint arXiv:2308.15674, 2023.

5. Bhatia, S. *Detecting Distributed Denial-of-Service Attacks.* Master's Thesis, Sacred Heart University. Available at: https://sacredheart.elsevierpure.com/files/40008552/Sajal_Bhatia_Thesis.pdf

6. SCIRP. *A Heuristics Clustering Algorithm and Naïve Bayes for DDoS Detection.* Scientific Research Publishing. Available at: https://www.scirp.org/html/4-7800482_81174.htm