

Shield-X: Vectorization and Machine Learning-Based Pipeline for Network Traffic Threat Detection [†]

Claudio Henrique Marques de Oliveira ¹, Marcelo Ladeira ¹, Gustavo Cordeiro Galvao Van Erven ¹
and João José Costa Gondim ^{2,*}

¹ Department of Computer Science, University of Brasília, Brasília 70910-900, Brazil; claudio.oliveira@aluno.unb.br (C.H.M.d.O.); mladeira@unb.br (M.L.); gvanerven@gmail.com (G.C.G.V.E.)

² Department of Electrical Engineering, University of Brasília, Brasília 70910-900, Brazil

* Correspondence: gondim@unb.br

[†] Presented at the First Summer School on Artificial Intelligence in Cybersecurity, Cancun, Mexico, 3–7 November 2025.

Abstract

This paper presents an integrative methodology combining advanced network packet vectorization techniques, parallel processing with Dask, GPU-optimized machine learning models, and the Qdrant vector database. Our approach achieves a 99.9% detection rate for malicious traffic with only a 1% false-positive rate, setting new performance benchmarks for cybersecurity systems. The methodology establishes an average detection time limit not exceeding 10% of the total response time, maintaining high precision even for sophisticated attacks. The system processes 56 GB of PCAP files from Malware-Traffic-Analysis.net (2020–2024) through a five-stage pipeline: distributed packet processing, feature extraction, vectorization, vector database storage, and GPU-accelerated classification using XGBoost, Random Forest, and K-Nearest Neighbors models.

Keywords: network security; anomaly detection; machine learning; vector database; real-time systems; cybersecurity

1. Introduction

Cybersecurity threats constantly evolve, making network security an increasingly complex challenge. Traditional detection methods, often based on known signatures and log records, cannot keep pace with the speed and sophistication of modern attack attempts and vectors [1], especially those using evasive techniques [2] or representing zero-day threats [3]. Given this scenario, more dynamic and intelligent approaches become necessary to identify malicious traffic in real-time [4]. The essence of our approach lies in transforming raw network packet data into information-rich vector representations [5], allowing the capture of subtle patterns and complex relationships in traffic that escape conventional analyses [6].

Recent studies highlight different approaches and architectures aimed at improving real-time detection, scalability, and accuracy of network security systems. Komisarek et al. [7] present a machine learning-based approach for anomaly and cyberattack detection in network traffic data transmitted in continuous streams using Big Data frameworks like Apache Kafka and Spark. Srivatsa et al. [8] introduce the LogSense architecture, a scalable solution for real-time log anomaly detection using Large Language Models and advanced stream processing frameworks, integrated with the Qdrant vector database for efficient retrieval. Yu et al. [9] propose LogMS, a log anomaly detection method based



Academic Editors: Luis Javier García Villalba, Héctor Manuel Pérez Meana, Gabriel Sánchez Pérez and Ana Lucila Sandoval Orozco

Published: 2 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

on multi-source information fusion and labeling probability estimation using Long Short-Term Memory models. These works highlight the importance of efficient and scalable methodologies for anomaly detection in computer networks but present significant gaps regarding adaptability in dynamic environments and efficiency in resource usage.

2. Proposed Method

The proposed methodology was developed with a focus on feasibility and validated through the implementation of a complete pipeline (Figure 1a). Data collection was performed from datasets from the Malware-Traffic-Analysis.net website, covering 2020 to 2024 and totaling 56 GB of PCAP files containing raw network data in packet captures initially processed to extract relevant characteristics. Network traffic data was collected using the Scapy tool to capture network packets in PCAP format containing information about network traffic, including source and destination IP addresses, ports, protocols, TCP flags, packet length, payload size, and content [10,11].

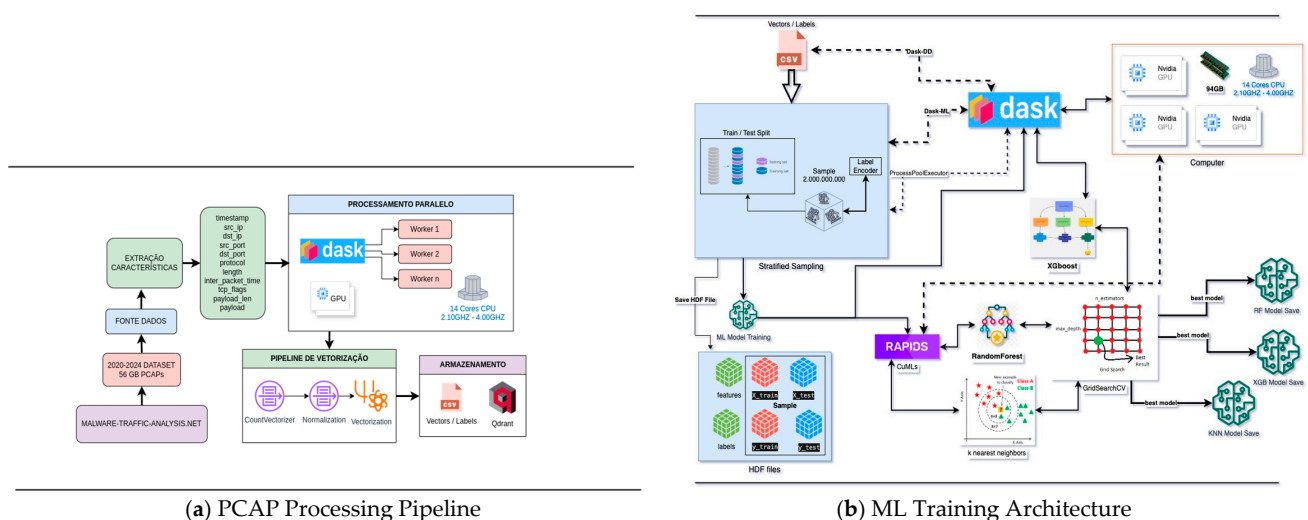


Figure 1. System architecture: (a) complete processing pipeline from PCAP files through Qdrant vector database to threat detection; (b) machine learning model training with acceleration and optimization.

Feature extraction from network packets transforms raw data into a structured format, enabling the creation of a more informative dataset for classification, prediction, and clustering tasks. Selected features include timestamp, src ip, dst ip, src port, dst port, protocol, length, inter-packet time, tcp flags, payload len, among others [6,12–18]. The importance of specific features has been established through extensive research: packet size for traffic pattern identification [6], protocol differentiation for detecting specific patterns [17], IP addresses for tracking traffic origin and destination [18], ports for identifying service types [16], TTL for identifying potentially malicious traffic [14], TCP flags as critical indicators of connection state [12], packet length for differentiating traffic types [13], timestamps for temporal traffic analysis [15], and inter-packet time for understanding traffic behavior [19].

In order to prepare data for machine learning analysis, features are normalized using the Scikit-Learn library, ensuring all features are on the same scale, preventing larger values from dominating during model training (Figure 1b). IP addresses are vectorized using CountVectorizer, transforming addresses into numerical embeddings, while TCP flags are converted to numerical values using MultiLabelBinarizer, applying the one-hot encoding technique. The resulting vectors from normalization and vectorization are stored in Qdrant, optimized for vector search. The collection was configured with a specific vector size and a

cosine distance metric for efficient retrieval [20]. Insertion was performed in batches using process chunk managed by a Dask client, allowing parallel processing [21]. Each point in Qdrant contains the vector and a payload with details of the original packet.

The malicious network traffic detection methodology includes training machine learning models using collected and processed data essential for identifying anomalous patterns. Three main models were trained using a Dask cluster with GPUs configured via LocalCUDACluster, leveraging parallel processing capacity [21]. K-Nearest Neighbors was implemented using the cuML library to leverage GPU acceleration with hyperparameter search, including the number of neighbors and the distance metric [22]. Random Forest was trained using cuML implementation configured for different numbers of estimators and maximum depths, providing robustness against overfitting through aggregation of multiple decision trees. XGBoost utilized the GPU-optimized version trained with different learning rates, numbers of estimators, and maximum depths [23]. Training used GridSearchCV for hyperparameter search, ensuring optimal parameters, with 10 billion stratified samples maintaining proportional class representation.

Real-time analysis of network data is essential for efficiently identifying and mitigating cybersecurity threats using Apache Kafka as a distributed streaming platform acting as a backbone for ingestion, processing, and continuous distribution of large data volumes [10]. The analysis flow operates through continuous ingestion where KafkaProducer captures network packets in real-time, extracts characteristics, and sends information to a dedicated Kafka topic. Distributed consumption and processing employ KafkaConsumer reading messages from topics with processing load distributed using Dask. Each received packet transforms into a feature vector by applying normalization and vectorization techniques from the preprocessing phase.

Generated vectors are compared with data stored in the Qdrant vector database using cosine similarity search to identify previously stored packets vectorially similar to the current packet. Score calculation based on cosine similarity, where 0.9 suggests a small angle between vectors, indicating high similarity [24], effective for detecting patterns and anomalies in high-dimensional data [25]. Results with high similarity indicate potential anomalies selected for the second verification layer using trained XGBoost, KNN, and Random Forest models, generating a final prediction if model confidence exceeds 80%. All packet processing is distributed using Dask with GPU integration via LocalCUDACluster, allowing parallel processing of multiple packets, optimizing available hardware resources, with garbage collection for efficient memory management.

3. Results

Experiments were conducted on hardware featuring Intel Xeon E5-2680 v4 @ 2.40 GHz with 28 cores, 189 GB DDR4 RAM, 2x NVIDIA P40 24 GB GPUs, 1x NVIDIA 3090 24 GB GPU, and NVMe SSD 2 TB storage. Software environment included Ubuntu 22.04 LTS, XGBoost 3.0.0, Rapids cuML 24.04, Dask 2024.5.0, Qdrant 1.12.6, Scapy 2.5.0, and Apache Kafka 3.4.0. The dataset consisted of 56 GB PCAP files from Malware-Traffic-Analysis.net totaling approximately 180 million packets for training, plus 10 GB controlled test data from an isolated environment executing recent malware samples not included in the training set.

All models presented exceptional performance with accuracy above 0.96, as shown in Table 1. The KNN model stood out slightly with the highest accuracy (0.9736) and best values for macro metrics, evaluating average performance among all classes without considering size, which is particularly important in malware detection scenarios where minority classes are as important as majority classes [26]. Random Forest presented the best balance between precision and recall with a macro F1-score of 0.96, while XGBoost obtained slightly lower results in macro metrics but maintained excellent performance

considering class weights. When considering weighted average by class size, all models converge to similar values (0.97), indicating consistent performance between classes of different sizes, suggesting robustness against class imbalance, a common challenge in malicious traffic detection [27].

Table 1. Comparative model performance.

Model	Accuracy	Macro Precision	Macro Recall	Macro F1-Score
XGBoost	0.9700	0.95	0.94	0.94
Random Forest	0.9600	0.98	0.95	0.96
KNN	0.9736	0.9787	0.9775	0.9773

The system achieved a detection rate of 99.9% for simulated malicious traffic using the tcpreplay tool (Table 2), highlighting the model's ability to accurately identify anomalous patterns and cybersecurity threats. False-positive incidence was 1% with simulated traffic and 5% with monitored real traffic, which is considered low, demonstrating system robustness. The distributed Dask environment, combined with GPU computational power, enabled real-time analysis of billions of packets, maintaining low latencies and high throughput. Architecture based on Dask and GPUs optimizes resource usage, allowing efficient workload balancing and better hardware utilization with 85–95% GPU utilization during intensive processing, load balancing across 3 simultaneous GPUs, and 70% reduction in training time compared to CPU-only processing.

Table 2. Summarizes the detection performance metrics.

Metric	Simulated Traffic	Real Traffic
Detection Rate	99.9%	97.8%
False Positives	1%	5%
Average Detection Time	8 ms	12 ms
Throughput (packets/s)	1.2 M	800 K

Compared to traditional intrusion detection methods, the proposed methodology presents significant advantages in precision, efficiency, and scalability. While traditional methods fail to detect zero-day attacks or unknown variants, vectorization and similarity approach in Qdrant identifies subtle deviations from normal behavior, achieving high detection rates in simulated scenarios, including diverse malicious patterns. Rule-based systems generate many alerts for legitimate traffic superficially resembling malicious patterns; a combination of vector search with ML classification resulted in only 1% false positives in simulations and 5% in real traffic, significantly lower than observed in many traditional IDS/IPS systems, alleviating the burden on analysts [19,28]. Architecture demonstrated capability to process billions of packets with low latency, a performance level difficult to achieve with traditional monolithic systems without massive investments in specialized hardware [29,30]. Adding more processing nodes allows scaling analysis capacity as traffic volume increases.

The methodology provides a powerful tool for early threat detection, significantly contributing to critical infrastructure protection. Real-time detection capability of malicious traffic patterns allows for proactive response, mitigating potential attacks before causing significant damage. Reducing false positives alleviates the burden on security analysts, allowing focus on real threats and improving operational efficiency [19,28]. Despite high accuracy, intensive memory usage by models with increasing network traffic presented significant challenges requiring high-performance machines and GPU usage to maintain system efficiency and scalability. Three GPUs were used simultaneously in model training;

nevertheless, the implementation demonstrated a methodology that was highly effective in identifying anomalies and reducing false positives using parallel processing and GPU resources handling large data volumes in real-time.

4. Conclusions

This study presented an innovative approach for detecting malicious network traffic combining network packet feature vectorization with vector databases like Qdrant and advanced machine learning techniques. The methodology achieved a 99.9% detection rate with 1% false positives in simulated environments, establishing new benchmarks for cybersecurity systems. Future work includes memory consumption optimization, incremental learning implementation for adapting to new attack patterns, and integration with existing SIEM systems, enhancing operational workflows while maintaining the detection accuracy required for defending against sophisticated cyber threats.

Author Contributions: C.H.M.d.O., M.L., G.C.G.V.E. and J.J.C.G. contributed equally to the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors gratefully acknowledge the technical and computational support provided by Fundação de Apoio à Pesquisa do Distrito Federal (FAP/DF) through the Tech-Learn grant project “CONSTRUÇÃO DE MODELOS DE LINGUAGEM NATURAL PARA PROCESSAMENTO DE DADOS EM FONTES ABERTAS”.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Santos, V. Sistemas de Detecção de Intrusões (IDS—Intrusion Detection Systems) Usando Unicamente Softwares Open Source. SegInfo. 2010. Available online: <https://seginfo.com.br/2010/06/21/sistemas-de-deteccao-de-intrusoes-ids-intrusion-detection-systems-usando-unicamente-softwares-open-source/> (accessed on 18 August 2025).
2. Palo Alto Networks. O que é um sistema de detecção de intrusões? Cyberpedia, n.d. Available online: <https://www.paloaltonetworks.com.br/cyberpedia/what-is-an-intrusion-detection-system-ids> (accessed on 5 September 2025).
3. Hariharasubramanian, N. Signature Based IDS vs. Anomaly Based IDS: Understanding the Difference. Which is Best for Your Needs? Fidelis Security. 2025. Available online: <https://fidelissecurity.com/cybersecurity-101/learn/signature-based-vs-anomaly-based-ids/> (accessed on 12 September 2025).
4. Lip, Y.P.; Dai, Z.; Leem, S.J.; Chen, Y.; Yang, J.; Binbesh, F. A Systematic Literature Review on AI-Based Methods and Challenges in Detecting Zero-Day Attacks. *IEEE Access* **2024**, *12*, 144150–144163. [CrossRef]
5. Perumal, G.; Subburayalu, G.; Abbas, Q.; Naqi, S.M.; Qureshi, I. VBQ-Net: A Novel Vectorization-Based Boost Quantized Network Model for Maximizing the Security Level of IoT System to Prevent Intrusions. *Systems* **2023**, *11*, 436. [CrossRef]
6. Moore, A.; Zuev, D.; Crogan, M. *Discriminators for Use in Flow-Based Classification*; Technical Report; Intel Research: Cambridge, UK, 2005.
7. Komisarek, M.; Pawlicki, M.; Kozik, R.; Choraś, M. Machine Learning Based Approach to Anomaly and Cyberattack Detection in Streamed Network Traffic Data. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2021**, *12*, 3–19. [CrossRef]
8. Srivatsa, A.; Gudisa, V. LogSense: Scalable Real-Time Log Anomaly Detection Architecture. *arXiv* **2024**, arXiv:2408.13699. Available online: <https://clovlog.com/logsense.pdf> (accessed on 20 September 2025).
9. Yu, Z.; Yang, S.; Li, Z.; Li, L.; Luo, H.; Yang, F. LogMS: A multi-stage log anomaly detection method based on multi-source information fusion and probability label estimation. *Front. Phys.* **2024**, *15*, 1401857. [CrossRef]
10. Akanbi, A.; Masinde, M. A distributed stream processing middleware framework for real-time analysis of heterogeneous data. *Sensors* **2020**, *10*, 3166.

11. McKinney, W. Data Structures for Statistical Computing in Python. *SciPy* **2010**, 56–61. [[CrossRef](#)]
12. Alshammari, R.; Zincir-Heywood, A.N. Investigating two different approaches for encrypted traffic classification. In *Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust, Fredericton, NB, Canada, 1–3 October 2008*; IEEE: Piscataway, NJ, USA, 2008; pp. 156–166.
13. Garcia, M.L.; Marin, L.; Gomez-Skarmeta, A.F. Intrusion detection in wireless ad hoc networks. *J. Netw. Comput. Appl.* **2004**, *25*, 99–114.
14. Jung, J.; Paxson, V.; Berger, A.W. *Fast Portscan Detection Using Sequential Hypothesis Testing*; IEEE S&P: San Francisco, CA, USA, 2002; pp. 211–225.
15. Lakhina, A.; Crovella, M.; Diot, C. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Comput. Commun. Rev.* **2005**, *35*, 217–228. [[CrossRef](#)]
16. Mahoney, M.V.; Chan, P.K. *PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic*. Florida Institute of Technology; Technical Report CS-2001-04; Florida Institute of Technology: Melbourne, FL, USA, 2001. Available online: <https://cs.fit.edu/media/TechnicalReports/cs-2001-04.pdf> (accessed on 20 September 2025).
17. Park, K.; Pai, V.S.; Peterson, L.; Wang, Z. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In *Proceedings of the 6th Symposium on Operating Systems Design & Implementation (OSDI'04)*, San Francisco, CA, USA, 6–8 April 2004.
18. Paxson, V. Bro: A system for detecting network intruders in real-time. *Comput. Netw.* **1999**, *31*, 2435–2463. [[CrossRef](#)]
19. Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.-L.; Iorkyase, E.; Tachtatzis, C. Threat analysis of IoT networks Using Artificial Neural Network Intrusion Detection System. In *Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC)*, Yasmine Hammamet, Tunisia, 11–13 May 2016. [[CrossRef](#)]
20. Johnson, J.; Douze, M.; Jégou, H. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* **2016**, *7*, 535–547. [[CrossRef](#)]
21. Dask Development Team. Dask: Library for Dynamic Task Scheduling. 2016. Available online: <https://dask.org> (accessed on 22 September 2025).
22. Syriopoulos, P.K.; Kalampalikis, N.G.; Kotsiantis, S.B.; Vrahatis, M.N. kNN Classification: A review. *Ann. Math. Artif. Intell.* **2023**, *93*, 43–75. [[CrossRef](#)]
23. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 785–794. [[CrossRef](#)]
24. Huang, A. Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, Christchurch, New Zealand, 29 April 2008; pp. 49–56.
25. Aggarwal, C.C.; Hinneburg, A.; Keim, D.A. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory—ICDT 2001*; Van den Bussche, J., Vianu, V., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 1973. [[CrossRef](#)]
26. Fernández, A.; García, S.; Galar, M.; Prati, R.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer: Cham, Switzerland, 2018. [[CrossRef](#)]
27. Johnson, J.M.; Khoshgoftaar, T.M. Survey on Deep Learning with Class Imbalance. *J. Big Data* **2019**, *6*, 27. [[CrossRef](#)]
28. Puthal, D.; Nepal, S.; Ranjan, R.; Chen, J. A secure big data stream analytics framework for disaster management on the cloud. In *Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Sydney, Australia, 12–14 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1218–1225.
29. Mahapatra, T. Composing high-level stream processing pipelines. *J. Big Data* **2020**, *7*, 81. [[CrossRef](#)]
30. Nazari, E.; Shahriari, M.H.; Tabesh, H. Big data analysis in healthcare. *Front. Health Inform.* **2019**, *30*, 29.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.