International Journal of Safety and Security Engineering
Vol. 16, No. 10, October, 2026, pp. 2389-2409
Journal homepage: http://iieta.org/journals/ijsse

BotDefendX: Extracting Explainable Intelligence from Encrypted Network Data

Harsha Sonune[1,2*], Nilima Kulkarni[1]

[1] Department of Computer Science and Engineering, MIT School of Computing, MIT Art, Design and Technology University, Pune 412201, India
[2] Department of Information Technology, MKSSS's Cummins College of Engineering for Women, Savitribai Phule Pune University, Pune 411052, India

Corresponding Author Email: sonuneharsha30@gmail.com

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ijsse.161016

Received: 25 August 2026
Revised: 18 October 2026
Accepted: 23 October 2026
Available online: 31 October 2026

Keywords:
botnet detection, Distributed Denial of Service, shortcut learning, concept drift, novelty detection, ensemble learning, ANOVA F-test, SHAP explainability

ABSTRACT

Flow-based botnet detectors can sometimes claim 100% accuracy, which may be due to the way the traffic was captured, and may only be accurate for the captured traffic and not others. In this paper, we propose BotDefendX, an ensemble of soft voting (Random Forest (RF) plus support vector machine with radial basis function kernel (RBF-SVM)) with fixed hyperparameters, no synthetic oversampling and a drift-aware safety layer at the flow level. The high accuracy is inherent in the data because, on the earlier subset (FLNET2023 with 3 classes), a single shortcut is nearly 100 per cent accurate (shortcut index = 0.997). In the more realistic CSE-CIC-IDS2018 benchmark (shortcut index 0.56; 25,019 test flows; eight traffic types), the ensemble achieves 99.0% accuracy, a 0.989 Matthews correlation coefficient (MCC) and a weighted area under the receiver operating characteristic curve (ROC-AUC) of 0.99985, which is a little lower than LightGBM, XGBoost and RF. Detectors trained on one network fail to notice the presence of another network when evaluated with near-chance balanced accuracy (0.49–0.63), while the safety layer, consisting of a prediction-conditioned Wasserstein drift score, class-conditional novelty detector, and uncertainty monitor with null-calibrated thresholds, alerts to the presence of another network in all the cross-network tests. It raises false alarms in 0.3–0.4% of windows of known-attack or mixed traffic windows, and once it is given one labeling cycle (197–567 flagged flows), recall on the previously unseen attack increases from 0% to 49–99.6%. The timing features are most important for the behavioral signal, per SHapley Additive exPlanations (SHAP) and permutation importance, and a 'replay dashboard' illustrates this in real time. BotDefendX is thus not the most precise classifier, but rather a framework for evaluation and safety.

1. INTRODUCTION

Botnets are groups of infected computers and Internet of Things (IoT) devices that are remotely controlled by attackers to launch attacks—such as massive Distributed Denial of Service (DDoS) attacks. Attackers now throw Artificial Intelligence (AI) into these networks to scan for vulnerabilities on their own, respond to network defenses in real time, and operate very evasively. The 5G connectivity and poorly secured IoT devices have helped to bypass the traditional cloud-based mitigation of hyper-volumetric botnets like Aisuru and Kimwolf. The threat is also more sophisticated: DDoS-as-a-service solutions are making it easier than ever to launch a cybercrime campaign.

The commands and control lines behind such attacks are no longer readily visible. They change constantly, are encrypted and hide in traffic streams that are not independent and identically distributed (IID), and are non-homogeneous and heteroskedastic. The attack surface has grown to cover IoT devices, enterprise networks and cloud-edge infrastructure. Things are made worse by little or no payload inspection, and noisy and time-varying extra flow metadata. To implement coordination, operators must therefore be able to distinguish between coordination and payload visibility and deal with very high-class imbalance, and make decisions in real time that are auditable. False positives and false negatives have a negative impact on critical services and missed attacks are an invitation to adversarial penetration. In this study, they thus focus on flow-level detection, based on connection-level statistics instead of the contents of packets.

There are two strands of previous studies. Deep temporal models are accurate but opaque and heavy to compute and are therefore difficult to deploy at the edge and overfit the data. Simple lightweight classical learners work well but are based on the assumption of IID data, which is rarely satisfied. For both strands, probability calibration as well as explainability can be limiting, making the triage and post-mortem analysis difficult in the Security Operations Center (SOC). These weaknesses come to light when things are compared: popular baselines might appear best at first and then not carry over between them, and traffic drift and class skew impact all of them.

The methodological rigor requires accuracy, speed and transparency of each decision. The reported accuracy is a poor criterion for the usefulness of a detector for two reasons. First, public flow datasets frequently include some pieces of information unrelated to the attack behavior, such as the capture session, host or port number, and the labels and features of flows have been known to be erroneous [1-4]. Secondly, a detector that is deployed on a new network could lose accuracy without ever warning because a classifier will only return a known label, typically Benign, when it encounters an attack it has never seen [5-7].

To overcome these issues, in this paper we revisit BotDefendX—a flow level and explainable detection pipeline. We will stick with a conventional classifier—a soft voting ensemble of the Random Forest (RF) and support vector machine (SVM) with a radial basis function (RBF) kernel with 20 features selected using the analysis of variance (ANOVA) F-test, without any hyperparameter tuning and without any synthetic oversampling. Around it, we develop a data audit to quantify the amount of a task that a single feature can address as well as a drift-aware safety layer to alert the operator when the classifier's decisions should no longer be trusted, which flows exhibit suspicious behaviour, and how much labelling is required to restore trust. Results are presented in both balanced accuracy and macro-F1, as well as the Matthews correlation coefficient (MCC) and area under the receiver operating characteristic curve (ROC-AUC) based on the real class scores and the baseline majority class. SHapley Additive exPlanations (SHAP) and permutation importance are provided to show which features contribute to the decision.

This paper does not present a more precise classifier. The incremental deep and edge-oriented detectors mentioned in Section 2 [8-10] intend to achieve higher in-domain accuracy. BotDefendX, however, provides a framework with moderate in domain accuracy, but measures what the detector does not know and the false-alarm rate it can claim. Both are evaluated and reported and the framework can be combined with any classifier that can output class probabilities.

As seen in Figure 1, the publications have increased consistently since 2019 and reached its climax in 2025, corresponding to the proliferation of IoT systems and the growing worry about their security. The other publication types are significantly less common, and reviews and book chapters are in second place, with research articles 4 times more common. The curve, together with the distribution, indicates an already well-developed field with many experiments rather than surveys or short communications.

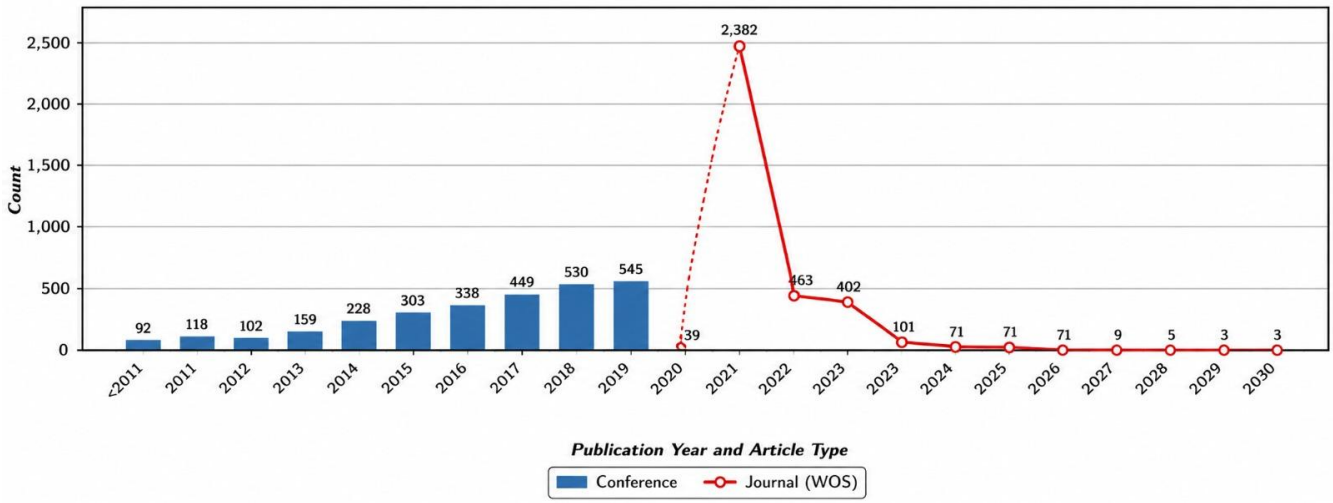

Figure 1. Year-wise growth and publication-type distribution of Internet of Things/botnet research

The main contributions are:

•Shortcut audit (C1): The data-quality audit for flow datasets, which includes correct de-duplication after identifier removal, a train-test overlap check, a single-feature probe and a shortcut index that indicates that near-perfect results on the FLNET2023 emulated subset are recording artifacts.

•Prediction-conditioned drift score (C2): Wasserstein drift score where the flows assigned to each class are compared with the training flows of that class, such that the drift alarms are not raised for known attacks which are correctly classified.

•Null-calibrated alarm thresholds (C3): The results of this study suggest that the threshold changes significantly when using a null calibration, while thresholds influenced by other factors change only slightly. The results of this study indicate that the threshold changes significantly with a null calibration, whereas the threshold changes with other factors only slightly.

•Class-conditional novelty detection (C4): A nearest-neighbor novelty detector that can detect more unseen-attack flows than Isolation Forest at a particular false-flag rate.

•Drift-triggered labeling loop (C5): An active-labeling feedback loop with a measured labeling budget and a disjoint re-evaluation.

•Multi-dataset and cross-network evaluation (C6): Evaluation on three public datasets, a cross-network matrix to demonstrate the lack of transferability of detectors, an ANOVA justification, a classifier comparison and a real-time demonstration using a replay.

The remainder of the paper is organized as follows. Section 2 reviews related work, including dataset quality and drift. Section 3 describes the datasets, the pipeline, the drift-aware layer and the experimental protocol. Section 4 reports the results, Section 5 discusses implications, future directions and limitations, and Section 6 concludes.

2. RELATED WORK

Previous research on botnet detection converges on three aspects that inform our design. First, foundation and hybrid learning approaches, such as incremental boosting and

continual-update ensembles, seek stability under non-IID, changing traffic, but at the cost of instance-level transparency and calibrated thresholds, which makes triage difficult at Internet service provider (ISP) scale. Second, federated and streaming detection models, such as recurrent neural network (RNN)/long short-term memory (LSTM) pipelines and graph neural models, capture temporal staging (scan–command-and-control (C2)–attack) and inter-host dependencies. However, they are opaque, expensive to train and impose on-host overhead, which makes it hard to run them over encrypted data with edge-grade functionality. Third, advanced methods and special domains such as smart homes and the Industrial IoT (IIoT) reduce central data exposure and latency, but they report inconsistent generalization at the intersection of device heterogeneity and label sparsity, and they seldom treat explainability as a first-class requirement.

2.1 Edge-middleware, surveys, and smart-home

Recent advancements include distributed deep neural network (DNN) middleware that partitions inference between the cloud and heterogeneous edge nodes (e.g., Raspberry Pi, Jetson and microcontrollers). These systems demonstrate real-time intrusion detection system (IDS) capabilities with low CPU and memory footprints, device-specific pipelines and stable latency under concurrent workloads. Owen et al. [11] provided a systematic taxonomy of the field, covering typologies, infection vectors, the contrast between anomaly-based and signature-based paradigms and between supervised and unsupervised baselines, and in doing so revealed structural gaps in explainability, calibration, encrypted-traffic processing, dataset bias and scalability. Shalan et al. [12] addressed the privacy issue with a smart-home architecture that integrates federated learning, blockchain-integrated role-based access control (RBAC) and knowledge distillation. Models are pre-seeded via transfer learning, adapted locally to device idiosyncrasies and periodically distilled by sharing class-score summaries. Mutambik [13] proposed the genetic algorithm (GA) with hybrid deep learning-based anomaly detection (GA-HDLAD) technique, in which a GA selects the features and a hybrid deep learning model identifies botnet traffic in IoT environments. Reported gains in per-device accuracy (e.g., doorbells, webcams and thermostats) illustrate a template for secure, label-efficient adaptation in non-IID homes while preserving auditability through on-chain governance.

Graph- and sequence-based representations have been used to recover the temporal structure that flat tabular models discard. Representing IoT traffic as a temporal multilayer graph, with devices as nodes and temporally annotated flows as edges, and propagating information through recurrence-gated graph convolutions exposes sequential phases of adversarial activity — scanning, C2, flooding — that standard classifiers treat as independent events [14]. These methods perform well in binary regimes, but multiclass settings reveal persistent ambiguity between temporally proximate behaviors under class imbalance, motivating drift-aware decision thresholds and per-class calibration. A complementary line restricts detection to short temporal windows (~1 s) and pairs unsupervised clustering with supervised classification (k-nearest neighbors (k-NN), SVM, multilayer perceptron (MLP)) in a protocol-agnostic pipeline, which is useful for recovering micro-temporal flow regularities and localizing C2 activity in encrypted or obfuscated traffic, including decentralized peer-to-peer variants [15].

Feature- and search-driven approaches form a second strand. Combining unsupervised device-log clustering with human-centric intent inference built on distance, angle, speed and efficiency features links human–computer-interaction behavior to network-security risk and supports proactive policy tuning against signature-less zero-day attacks [16]. For IIoT specifically, a two-step feature extraction pipeline — a Fisher-information/grasshopper optimization algorithm (GOA) filter-wrapper scored by k-NN fitness, followed by a chaos-initialized, Harris-Hawks-optimized neural network — balances exploration and exploitation under the highly dynamic workloads where fixed hyperparameters stop working [17]. At the feature-selection level, a metaheuristic combining K-medoids with Scatter Search reduces UNSW-NB15 to its most discriminative subset and outperforms deep-learning baselines trained on the full feature set [18], reinforcing that careful feature curation can substitute for model depth.

2.2 Federated/streaming detection and adversarial robustness

Federated and streaming architectures aim to preserve data locality while adapting to non-IID concept drift in real time. Federating graph neural networks lets clients train locally on graph-structured traffic and share only securely aggregated updates, improving F1 on peer-to-peer behaviors while preserving privacy, with adaptive client selection stabilizing convergence on rare patterns [19]. Pairing lightweight edge classifiers with a server-side LSTM/RNN that aggregates votes over temporal sequences achieves high F1 across sequence lengths and generalizes across datasets and sequence granularities [20]. Grounding the evaluation in real microcontroller-generated ping-flood traffic (ESP-01S) shows that, on minimal-feature datasets, a simple k-NN can outperform heavier models, underscoring the value of fit-for-purpose features over raw model capacity [21]. A Temporal Convolutional Network–Gated Recurrent Unit backbone tuned via the enhanced gorilla troops optimizer (EGTO) and the marine predator's algorithm (MPA) extends this to horizontal federated learning, reporting strong multiclass performance at sub-millisecond runtimes, though with communication and computation overheads that motivate more communication-efficient federated designs for large client populations [22].

A complementary research line aims to stress test these detectors in the presence of adversarial perturbation and work towards deployable accuracy at the edge. It is shown that evasion transfers across classifiers, and that adversarial training, margin sensitive thresholding and ensembles can resist small coordinated perturbations [23] by reinforcement-learning based adversarial generation that perturbs timing and volume features while preserving the malicious function in its entirety. As for deployment, an XGBoost–LightGBM ensemble with 17 high-signal features extracted from 77 features of CICIDS2017 achieves macro accuracy of more than 99%, while significantly reducing the training time, making it suitable for deployment on edge gateways [24]. With DDoS traces, about 99% or higher classification accuracy can be maintained with a weighted combination of Decision Tree, Naive Bayes and SVM while keeping false positives to a minimum to maintain the trust of the analysts [25]. Modularization of the early botnet activity detection at

the multi-access edge using parallel binary classifiers (scanning versus C2) achieved almost 100% accuracy, near zero false positives and approximately 16 ms latency, demonstrating that modularization and placement at the multi access edge can significantly reduce the time to detection (TTD) [26].

2.3 Advanced methods, datasets, and special domains

Work in this space increasingly targets specialized data modalities and deployment constraints rather than generic flow features. Detecting botnets from Transport Layer Security (TLS) metadata alone (handshake, size, timing) shows convolutional neural networks (CNNs) outperforming RF, but at higher inference cost, while RF retains an edge in interpretability via feature importances as a human-in-the-loop backstop [27]. For federated deployments under tight compute budgets, combining XGBoost-based selection with principal component analysis (PCA) compresses N-BaIoT to a handful of principal components, letting a differentially private MLP trained via Federated Averaging (FedAvg) reach near-100% metrics on only a few thousand floating-point operations (FLOPs), which is evidence that private, constrained-node IDS is achievable without large parameter budgets [28, 29]. Fusing payload language (Transformer token streams) with traffic images (CNN+bidirectional LSTM (BiLSTM)) by maximizing their mutual information extends detection to unseen botnet families and maps flows to severity classes to prioritize response [30]. Ensemble classification over botnet traffic features has likewise been used to predict botnet presence and its impact on IoT networks [31]. A GA-optimized, rule-directed k-NN over standard deviation of round-trip time (SDRTT), Google-hit and IP-churn features achieves sub-1% error with explanation-friendly features [32], and coupling a generative adversarial network (GAN) with a deep Q-network (DQN) agent to generate semantically consistent evasive examples converges faster and improves zero-day robustness over earlier GAN-based approaches [33]. Beyond these, the literature has also explored utility-driven microservice tactics for 5G/6G core-network mitigation [34] and the viability of quantum-hybrid deep learning for threat detection in noisy environments [35].

2.4 Dataset quality, concept drift and open-set detection

Another body of work deals with the evaluation of detectors. The studies [1-3] reported the presence of labelling errors, feature-extraction errors and duplicate records in CIC-IDS2017 and CSE-CIC-IDS2018, and how correcting these errors alters the reported detection performance. Among the problems of machine learning in security that Arp et al. [4] identified sampling bias and snooping and spurious correlation. The results presented in this section provide the impetus to conduct the data audit of Section 3.1. A change in the distribution of the data, or of the label given the data, is called concept drift and is surveyed in the studies [5, 6]. Most drift detectors are based on error rates, and hence require labels, whereas label-free drift detectors use a comparison of feature distributions between two windows, a reference and a current window, an empirical study comparing two-sample tests for drift detection on raw and reduced representations is presented in this study [7]. The related problem of rejection of inputs from classes not represented in training is addressed by open-set recognition [36]. We extend these notions and make the distributional comparison dependent on the result of the classifier, set alarm thresholds on a collection of null data held out for this purpose, and evaluate on network-intrusion flows using leave-one-attack-out and cross-network evaluation.

2.5 Research gap

Based on the preceding survey, we synthesize the critical limitations and opportunities in recent botnet and IoT IDS literature into a compact decision-grade matrix. The gaps cut across data, models, deployment and assurance: adversarial robustness, early scan/C2 detection, feature bloat on constrained nodes, privacy-preserving federation, calibration and explainability. For each gap, the table lists representative advances from the literature (with citations) and translates them into concrete, action-oriented insights. Literature from 2024 to 2026 has increasingly focused on the practical limitations of these advanced methods: recent works advocate bridging the sim-to-real gap for constrained IoT deployments [37], addressing the privacy–utility trade-offs inherent in federated learning architectures [38], and developing online adaptive learning mechanisms to counteract zero-day concept drift and severe class imbalance [39].

The gap analysis in Table 1 directly informs the scope and design boundaries of the proposed system. The model operates on flow-level, protocol-agnostic features and does not rely on deep packet inspection, so it remains applicable to encrypted traffic.

Table 1. Bridging the divide in botnet defense: research gaps, innovations, and actions

Research Gap	Key Insight	Recent Advancements
Adversarial robustness	Evasive samples often break attack semantics or need heavy retraining	GAN+DQN crafts valid evasions faster than prior methods [33]
Feature bloat vs. real-time IoT IDS	High-dimensional features hurt latency and memory	Hybrid/bio-inspired selectors reach ≥ 99% with 14–31 features [40, 41]
Dataset curation and leakage	IP/port fields inflate scores; imbalance and protocol mix persist	Flow-timing statistics (duration, inter-arrival time) generalize better than identifier fields [42]
Code-level vs. network-only IDS	Network-only IDS can miss obfuscated mobile payloads	Model checking localizes C2 logic with perfect accuracy [43]
Concept drift and online adaptation	Static models decay as traffic and attacks evolve	Elastic retraining with feature pruning sustains > 99.5% under drift [44, 45]
Calibration under imbalance	High accuracy does not imply calibrated scores or stable alerts	Expected calibration error/Brier score and threshold reporting are largely absent across pipelines [46, 47]
Sim-to-real gap	Simulated accuracy degrades in live, resource-constrained deployment	Aggressive feature selection and validation on realistic benchmarks (e.g., Bot-IoT) [37, 38]
Drift, imbalance, and opacity	Opaque alerts lack operator trust for automated prevention	Online adaptive learning coupled with explainable AI [38, 39]

Note: Generative Adversarial Network (GAN); Deep Q-Network (DQN); Internet of Things (IoT); Intrusion Detection System (IDS).

Dataset quality is addressed by a data audit with correct de-duplication, a shortcut index and evaluation on three datasets; drift is addressed by a calibrated, prediction-conditioned safety layer with a labeling loop; feature selection is kept compact (20 features) for edge viability; and explanations are provided by permutation importance and SHAP. The calibration pursued here is that of the alarm thresholds rather than of the class probabilities. These design decisions are reflected in the sections that follow, which describe the data, the pipeline and the drift layer and evaluate them against the gaps identified above, rather than attempting a generic solution to all network-security problems.

3. METHODOLOGY

BotDefendX consists of 3 parts that cooperate. The offline stage cleans the data and trains the classifier and calibrates monitoring thresholds. A real-time stage classifies all flows and monitors the stream. The analyst feedback loop goes back to training with labeled flows. Section 3.1 contains a description of the data, section 3.2 of each component and section 3.3 of the experimental protocol. All equations are collected in this section.

3.1 Datasets and data-quality controls

3.1.1 Dataset identity and provenance

Three public flow datasets with different recorders on different networks (Table 2) are used. All of them together provide us with the opportunity to listen to the data, to run a detector on a realistic source, and to observe what happens when a detector trained on one recording encounters a foreign network.

Dataset A is the set of three comma-separated values (CSV) files that were used in the previous version of this work (Dataset-1-BOT.csv, Dataset-1.csv and Dataset-2.csv). They are a three-file slice of FLNET2023 [48], which is an intrusion dataset collected in an emulated ISP network for federated learning. The files are not a part of the CIC-IDS-2017 release, and the authors were not responsible for capturing them, which we refer to as the FLNET2023 subset. Only these files were used as the full set of files could not be downloaded. There is one class per file, with these recorded in the following sessions (Table 3): Normal (6 days in May, 2023), DDoS-bot (5 mins on 13th May, 2023), DoS-slowhttp (4 mins on 27th April, 2023). Classes hence vary in the day on which they were recorded, number of source addresses (two, three and two), and in destination port (5001, 5050 and 7200). Such classes would be separated without any effort, using any identifier, like a timestamp or a port number. Sorting the flows by time and taking a 70% cut, the cut would be in the middle of the classes, not between past and future. It's for this reason that the identifier columns are eliminated first (Section 3.1.2) and why we report no chronological split.

No row contained missing or infinite values. Secure Hash Algorithm 256 (SHA-256) digests of the files used: Dataset-2.csv, 09ee77fee3cf39342abed1111290ae9dd4a79638b3979b6b6e2fb50a4f91ab34; Dataset-1-BOT.csv, d5e70728453d362087e07aa17ac3720e3827b8c9a90baaae55532ceeab902390; Dataset-1.csv, 6fc3b1a9a49e2141f746f57207e195b5e2b7f17667412bb7a32cdcd5d9ba70af.

Table 2. Datasets used in this study, their origin and their role

Dataset	Origin	Role
FLNET2023 subset (Dataset A)	Emulated ISP network (CORE emulator, GEANT-2012 topology); benign traffic generated with hping3, attacks with MHDDoS and slowhttptest [48]	Shortcut audit; second in-domain dataset; one network in the cross-dataset test
CSE-CIC-IDS2018 (Dataset B)	Testbed of several hundred machines on Amazon Web Services; benign traffic generated from user-behavior profiles on the same network and days as the attacks [49]	Main benchmark: classifier, drift layer, feedback loop
CIC-DDoS2019 (Dataset C)	Separate testbed with 17 DDoS attack types, mostly reflection and amplification floods [50]	External network; cross-dataset test only

Table 3. Capture provenance and curation of the three FLNET2023-subset files (one class per file)

Class (File)	Capture Window (First to Last Flow)	Raw Rows	Exact Dup.	Confl.	Unique Kept
Normal (Dataset-2.csv)	2023-05-11 14:54:04 to 2023-05-17 14:54:25 (6 days)	104,146	99,972	0	4,174
DDoS-bot (Dataset-1-BOT.csv)	2023-05-13 15:48:13 to 15:53:13 (5 min)	87,808	158	1	87,649
DoS-slowhttp (Dataset-1.csv)	2023-04-27 10:00:51 to 10:04:52 (4 min)	10,590	0	1	10,589
Total		202,544	100,130	2	102,412

The raw and unique counts are reported separately. The biggest file (104,146 rows) is Normal, but only contains 4,174 unique flows (95.99% of all rows are exact copies, as is to be expected from synthetic hping3 traffic). After removing duplicates and conflicting flows, the classes contain 4,174 Normal, 87,649 DDoS-bot and 10,589 DoS-slowhttp flows, 102,412 in total. In this set of unique flows, Normal is the minority class with 4.1% and DDoS-bot the majority class with 85.6% (Figure 2).

Normal appears to be the largest class only in the raw counts, because of duplication; we therefore report raw and unique counts separately throughout. Experiments were conducted on a working subset of up to 40,000 sampled rows per label; duplicated rows were removed, and the number of unique flows per class (Normal: 2,723, DDoS-bot: 12,000, DoS-slowhttp: 10,589) was capped at 12,000. Figure 2 compares, on a logarithmic vertical axis, the raw rows as delivered, the unique flows after removal of identifier columns, duplicates and conflicting labels, and the flows used in the experiments (at most 40,000 rows sampled per label and 12,000 flows per class).

Dataset B (CSE-CIC-IDS2018) was collected using the methodology and set of features introduced in CICFlowMeter [49]. It has 16,233,002 flows in its ten daily files. The streaming was chunked into 250k-row chunks and uniform random sampling was performed to keep a max of 40k flows

per label from each file (813,707 flows total). We do not accept Brute Force traffic, Web Attack / Infiltration traffic is off scope, just Benign, Bot, DDoS and DoS families. Incomplete rows were discarded, leaving 810,716 rows, of which 610,450 rows were from the in-scope families. The number of exact duplicates was 105,063 and the number of conflicting labels was 278, and the number of unique flows in each class was capped at 12,000 (Table 4). DoS-SlowHTTPTest is excluded, as almost all the flows are copies: only 52 out of 40,000 sampled flows are unique (0.13%). This dataset has also been reported in the literature for labelling and feature extraction issues [1-3].

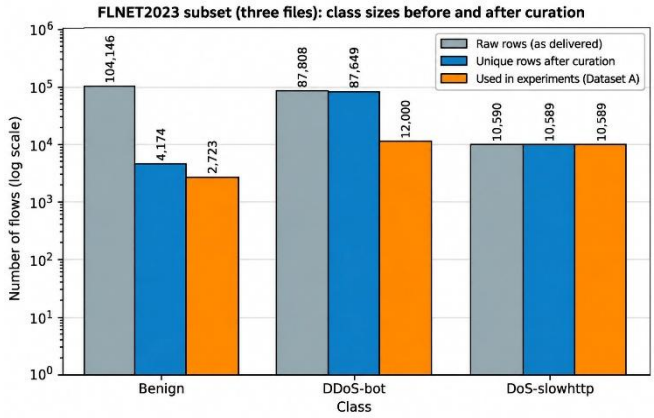

Figure 2. Class sizes of the three FLNET2023-subset files

Table 4. CSE-CIC-IDS2018 class composition after cleaning (eight classes, 68 features, 83,394 flows used)

Class	Raw Rows	Unique Rows	Unique (%)	Used
Benign	397,730	342,223	86.0	12,000
Bot	40,000	20,885	52.2	12,000
DDoS-HOIC	40,000	24,320	60.8	12,000
DDoS-LOIC-HTTP	40,000	39,992	100.0	12,000
DDoS-LOIC-UDP	1,730	1,730	100.0	1,730
DoS-GoldenEye	40,000	39,890	99.7	12,000
DoS-Hulk	40,000	26,635	66.6	12,000
DoS-Slowloris	10,990	9,712	88.4	9,664

Dataset C, CIC-DDoS2019 [50], contains Benign traffic and 17 types of DDoS attacks. It is de-duplicated and conflicting labels are removed, resulting in 93,659 flows and 64 features, with a max of 12000 per class. It is only used in the cross-dataset test, where all the labels become benign or attack. All three datasets are part of the CICFlowMeter feature family but the names of the columns and labels vary (fwd_pkt_len_max, FwdPktLenMax, FwdPacketLengthMax and Normal, BENIGN, Benign). Each label is thus also associated with a family (Benign, DDoS, DoS or Bot) as well as with a binary label (attacks or not). Only the 58 features shared by all three datasets are used for the cross-dataset test.

3.1.2 Cleaning and class handling

There are 7 steps in cleaning. (i) Identifier columns (IP addresses, ports, timestamps, flow identifiers and the protocol field) are dropped. This makes the detector learn behavior of traffic and not who sent it or when. (ii) All features are converted to numbers, infinite values are replaced with missing values, and repeated header lines in the CSE-CIC-IDS2018 files are removed and rows with missing values are deleted. (iii) One copy of exact duplicates is eliminated. This is for steps that follow step (i); prior to (i), each row is distinct in terms of timestamp or port. (iv) When two rows are the same feature but with different labels, we delete both rows. (v) Too many copies: classes with too many copies are skipped (DoS-SlowHTTPTest). (vi) 12,000 different flows are randomly sampled uniformly from each class. (vii) Constant (unchanging) columns are eliminated. The cap in step (vi) combined with the balanced accuracy, macro-F1 and MCC addresses the problem of class imbalance. Interpolation between the actual flows can generate invalid values (such as half a packet), so we do not use the synthetic minority oversampling technique (SMOTE).

3.1.3 Shortcut audit

The audit asks: what can one feature solve before any model is trained? For every feature, a decision tree of depth at most three is trained on that feature alone ($70/30$ split, at most 40,000 flows), and its balanced accuracy BA_j on held-out data is recorded. The shortcut index is

$$SI = \frac{b^* - 1/K}{1 - 1/K}, b^* = \max_j BA_j \quad (1)$$

where, K is the number of classes and $1/K$ is the chance level. An index value of 0 indicates no feature helps. An index of 1 indicates that one feature can solve the complete task and therefore a high accuracy does not imply high detection capability. The audit also includes percentages of rows in the test data that are in the same row in the training data and percentages of unique data rows in each class.

3.1.4 Data splits

All splits are stratified and use the fixed seed 42 (Table 5).

Table 5. Data splits used by each experiment

Experiment	Split
Main hold-out result	70% training, 30% test
Model comparison	Stratified 5-fold cross-validation on up to 40,000 flows
Feature-selection comparison	Stratified 5-fold cross-validation on up to 20,000 flows; 70/30 hold-out for the full ensemble
Leave-one-attack-out drift tests	Known traffic: 50% training, 20% calibration, 15% test, 15% evaluation. Unseen attack: 50% for the live stream, 50% for final evaluation. Training capped at 30,000 flows
Cross-dataset test	Source dataset: 60% training, 20% calibration, 20% test. Other datasets: up to 20,000 flows each, used only for testing
Robustness test	70/30, training capped at 30,000 flows

3.2 Proposed solution—BotDefendX

Figure 3 shows the system as implemented. It is read in a snake pattern: the first row (data), the second row from right to left (learning and calibration) and the third row (real-time detection). The light-blue band runs offline in a notebook, the light-green band in the real-time engine, and the dashed arrows are the feedback loop. The offline part comprises the data sources, ingestion and harmonization, cleaning (de-duplication after identifier removal), the data audit, stratified

splitting, the training pipeline fitted on training data only, drift-layer calibration on a held-out split, and evaluation. The real-time part comprises the model bundle, the real-time engine, the drift-aware monitor and the dashboard. In the feedback loop, only flagged flows from alarmed windows are labeled and used for retraining, which is evaluated offline. The labels C1–C6 in Figure 3 mark the contributions of this work.

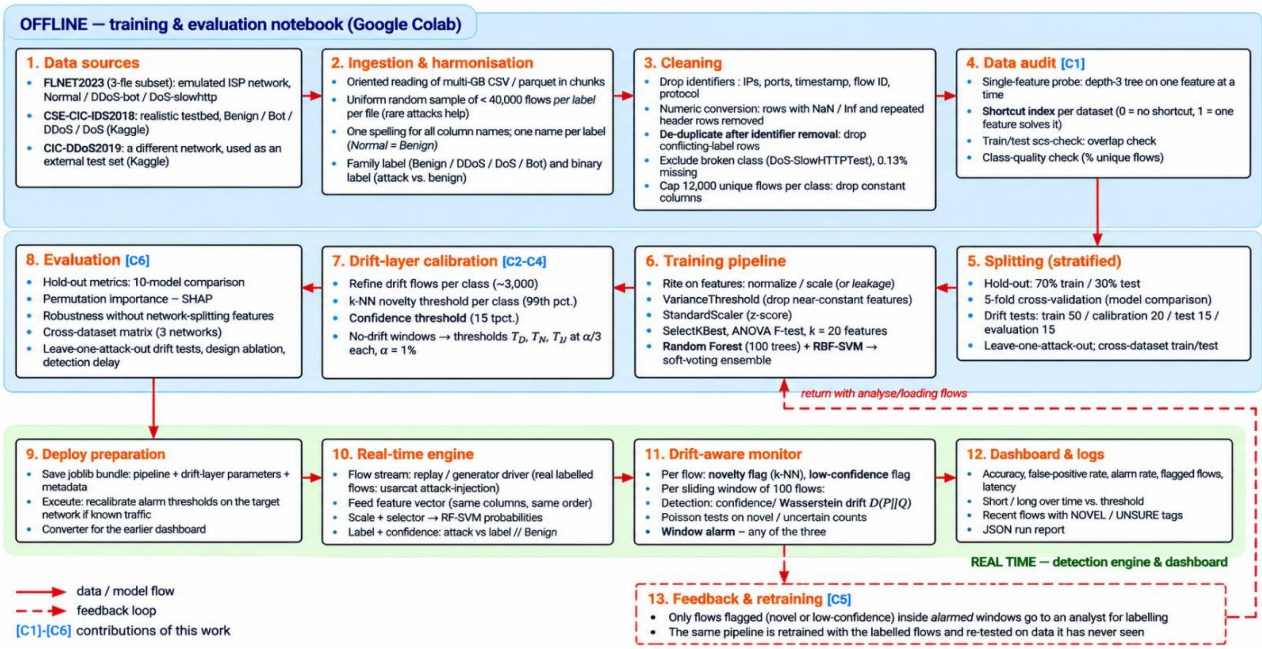

Figure 3. BotDefendX architecture

3.2.1 Offline training pipeline

(1) Preprocessing and feature selection. The three steps are chained in a scikit-learn [51] pipeline so that they are fitted on training data only and are merely applied to test data. A variance threshold (10^{-10}) discards near-constant features, and each remaining feature is standardized:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{2}$$

where, μ_j and σ_j are estimated on the training split. Each feature is then scored using an ANOVA F-test [52] which is a ratio of the between-class variance and within-class variance:

$$F_j = \frac{\sum_{c\in\mathcal{Y}} n_c \left(\bar{z}_{c,j} - \bar{z}_{\cdot,j}\right)^2 / (|\mathcal{Y}| - 1)}{\sum_{c\in\mathcal{Y}} \sum_{i\in c} \left(z_{ij} - \bar{z}_{c,j}\right)^2 / (N_{\text{train}} - |\mathcal{Y}|)} \tag{3}$$

where, n_c is the number of training flows of class c, $\bar{z}_{c,j}$ is the class mean of feature j and $\bar{z}_{\cdot,j}$ is the overall mean of feature j. The top 20 of the p features that remain after the variance threshold are retained,

$$\mathcal{S} = TopK\left(\{F_j\}_{j=1}^{p}, k\right), k = 20 \tag{4}$$

and every later step works on the projection of the flow onto this set. Section 4.5 compares the method of ANOVA with three other selectors, and demonstrates that the number of selected features, k = 20, is reasonable.

(2) Classifier. We use two base classifiers: an RF [53] with 100 trees, a maximum depth of 15 and a minimum number of flows per leaf of at least five, and an SVM [54] with RBF kernel, default regularization and Platt scaling (probability outputs) [55]. Soft voting is where they average their class-probability vectors and return the most probable class.

$$p^{ens} = \frac{1}{2}(p^{RF} + p^{SVM}), p^{RF}, p^{SVM} \in [0,1]^{|\mathcal{Y}|} \tag{5}$$

$$\hat{y} = \underset{c\in\mathcal{Y}}{argmax}\, p_c^{ens}, \kappa = \underset{c\in\mathcal{Y}}{max}\, p_c^{ens} \tag{6}$$

where, κ is the decision confidence. Flows that are not predicted as Benign are considered attacks. The hyperparameters are fixed and not tuned for in-domain or searched for, so there is not an opportunity for in-domain score inflation. On a Google Colab central processing unit (CPU), training the hold-out model takes 1.1 s for Dataset A (17,718 flows) and 121 s for Dataset B (58,375 flows); most of this time is spent on the SVM.

(3) Evaluation metrics. Since the classes are imbalanced, accuracy is always reported along with that for a majority-class predictor, and balanced accuracy (mean per-class recall), macro-F1 and the multi-class MCC [56] are also used. The flow-level false-positive rate (FPR) is the percentage of Benign flows that are classified as a class of any attack. The false-negative rate (FNR) is the percentage of attack flows that are classified as Benign, but are actually an attack; a classified attack with the name of another attack is not a false negative. ROC-AUC is calculated from all soft-vote probability levels of all test flows, and a support-weighted average of the per-class one-versus-rest (OvR) AUCs is used:

$$AUC_{OvR} = \sum_{c\in\mathcal{Y}} w_c \; AUC_{c\ vs\ \neg c}, w_c = \frac{n_c}{N_{test}} \tag{7}$$

where, n_c is the number of test flows of class c and N_{test} is the total number of test flows. A binary attack-versus-benign AUC is computed with the attack score $1 - p^{\text{ens}}_{\text{Benign}}$. For monitoring windows, the term false-alarm rate is used to distinguish it from the flow-level FPR.

(4) Explainability. Global importance is computed using permutation importance on 3,000 test flows (five repetitions, using balanced accuracy as score). A complementary view can be obtained from SHAP values [57] of the RF component, calculated using TreeExplainer on 1,000 flows. All the trained pipeline and the parameters of the drift layer are packaged as one model bundle for deployment.

3.2.2 Drift-aware safety layer

If there is an attack that is not included in the classes that the classifier has been trained on, the classifier still assigns a known class, usually Benign. The safety layer then monitors the traffic surrounding the classifier and triggers an alarm when it is no longer safe to rely on the decisions made by the classifier. It is based on windows W of 100 consecutive flows, in the model's own feature space (the 20 selected, standardized features), and combines three signals.

Signal 1: prediction-conditioned drift. For every class c we keep a fixed sample of at most 3,000 flows for training, R_c. We represent the flows of window W that the classifier c labeled as W_c. The drift score is

$$D(W) = \min\left\{1, \frac{1}{2.5}\sum_{c:\,|W_c|\geq m}\frac{|W_c|}{|W|}\cdot\frac{1}{F}\sum_{f=1}^{F}\mathcal{W}_1\left(W_c^{(f)}, R_c^{(f)}\right)\right\} \quad (8)$$

where, W_1 is the Wasserstein distance [58] calculated per feature, $F = 20$ is the number of features, and $m = 10$ is the minimum number of flows a class must have in the window to be scored. The factor $1/2.5$ and the cap at 1 make $D(W)$ be in the interval [0, 1]. A correct classification of a known attack is similar to the training flows of its class and contributes little to the score. A new attack assigned to a known class does not match the expected appearance of that class; it increases the score. A monitor that compared each window only with benign traffic alone would label all attacks as drift.

Signal 2: per-flow novelty. The novelty score of a standardized flow z is the average distance from z to its 5 closest neighbours in the reference flows of the same class.

$$\nu(z) = \frac{1}{5}\sum_{r\in NN_5(z;R_{\hat{y}})} d(z,r) \quad (9)$$

where, $d(\cdot,\cdot)$ is the distance between two standardized flows and $NN_5(z;R_{\hat{y}})$ is the set of the five reference flows of the predicted class that are closest to z. A flow is identified as NOVEL if it has a score above the per-class novelty score threshold θ_c^N which is defined as the 99th percentile of novelty scores for the calibration data.

Signal 3: uncertainty. A flow is marked UNSURE when the confidence κ is below θ^U, the first percentile of the calibration confidences. This is an application-tuned confidence rule which replaces a fixed hand-picked confidence rule. The flags also indicate to the feedback loop suspicious flows in an alarmed window.

$$NOVEL(z) = \mathbb{1}\left[\nu(z) > \theta_{\hat{y}}^N\right],\quad UNSURE(z) = \mathbb{1}[\kappa < \theta^U] \quad (10)$$

This replaces a predetermined confidence rule with one that is tuned to the application. The flags also indicate which flows of an alarmed window are suspicious, which the feedback loop uses.

Poisson tests. If traffic doesn't change, approximately 1% of flows are flagged by construction, and some classes (diverse benign traffic, for example) are flagged more frequently than others. An alarm should thus only be triggered for a window which has more flagged flows than expected for the window's combination of predicted classes. On the calibration data, let π_c be the flag rate of class c. The mean number of flags in window W is E and the number of flags observed in W is N, so the surprise is

$$S(N) = -log_{10}Pr[Poisson(E) \geq N],\quad E = \sum_{i\in W}\pi_{\hat{y}_i} \quad (11)$$

Calibration of the alarm thresholds. Adjusting alarm set points. All thresholds are derived from a calibration split, which is neither used for training nor reporting. The first half consists of the novelty and confidence thresholds for each class. The second half provides the flag rates π_c and the null distribution, consisting of 400 windows of 100 flows from known traffic (single-class and mixed windows) and we compute the three statistics. Each alarm threshold is defined as the $1-\alpha/3$ quantile of its null distribution, with α = 1% being divided by a Bonferroni correction [59] among the three signals:

$$\tau_s = Q_{1-\frac{\alpha}{3}}\left(\{s(W)\}_{W\in\mathcal{W}_0}\right),\quad s\in\{D, S(N_{\text{nov}}), S(N_{\text{uns}})\}, \alpha = 0.01 \quad (12)$$

where, $\mathcal{W}_0$ is the set of null windows, and the thresholds of D(W), S(N_{nov}) and S(N_{uns}) are denoted τ_D, τ_N and τ_U, respectively. The two count thresholds are never looser than the Poisson test at the level $\alpha/3$. When the level of any signal goes above the threshold a window will trigger an alarm,

$$A(W) = \mathbb{1}[D(W) > \tau_D \vee S(N_{\text{nov}}) > \tau_N \vee S(N_{\text{uns}}) > \tau_U] \quad (13)$$

so that, by construction, the window false-alarm rate on known traffic is about α. The original design used a fixed threshold of 0.015; this is only kept as a baseline in the ablation of Section 4.7.

3.2.3 Real-time inference pipeline

The real-time package not only loads the same model bundle as the notebook, but also includes the thresholds and runs it on a live stream. A flow record is received (in our experiments, recorded flows played back from a file at a desired rate, with or without an attack that was not observed during training). Its features are placed in the column order in which it was trained (feature contract) and it uses 0 for missing or invalid values. Then the stored variance filter/scaler and selector are applied and the ensemble outputs a label and its confidence. These flows are then labeled attack, NOVEL or UNSURE and the drift score and two poisson tests are calculated again using the last 100 flows. The web dashboard is updated every second and displays counts, accuracy (replay – when labels are known), alarm rate, flagged flows, latency, drift score and threshold as well as the most recent flows with their labels. No online adaptation of the thresholds. They are calculated in advance based upon the known traffic of the network being monitored. No packets are captured or generated, just flow records are processed.

3.2.4 Analyst feedback loop

We evaluate the feedback loop offline, in four steps. (1) A stream is replayed in which a hidden attack appears. (2) Flows that are placed in an alarmed window, labeled NOVEL or UNSURE are transferred to a simulated analyst who assigns the correct labels to some of the flows, including some false alarms. (3) BotDefendX is retrained using the combination of the original training data with the labeled flows. (4) The half of the hidden attack that was not streamed and a disjoint set of known traffic are used for testing the retrained model. Labeling budget equals the number of labeled flows. The analyst is flawless and we only do one cycle, with only one hidden attack. The dashboard does not automatically retrain.

3.3 Experimental protocol

The splits in Table 5 are used for all evaluations. Model comparison. BotDefendX is compared with RF (default settings), Decision Tree, SVM, k-NN (k = 5), an MLP (two hidden layers of 64 and 32 units), Gaussian Naive Bayes, XGBoost [60] and LightGBM [61], and with two sanity baselines (always predict the majority class; a depth-3 tree on the single best feature). All use the same preprocessing and the same folds. Feature selection. Only the features in the training folds are ranked by ANOVA, mutual information, chi-square and RF. Top-k = {5, 10, 15, 20, 30} is used to train the RF part of BotDefendX, and stability is defined as the average of the Jaccard index of the top-20 features in the five folds. Robustness. BotDefendX and an RF are retrained without the three features that are depending on the machines that generated the traffic (init_fwd_win_byts, init_bwd_win_byts and fwd_seg_size_min). Cross-dataset. The 58 shared features are then used to train a binary (benign versus attack) detector for each of the datasets, and it is tested on all three datasets. Leave-one-attack-out. All the attack classes (2 in Dataset A and 7 in Dataset B) with at least 300 unique flows are sequentially deleted from training. We then evaluate false alarms on 150 windows of benign-only traffic, known-only traffic and mixed known traffic (5%, 10%, 20% and 100% of the flows are from the hidden attack), and detection on 150 windows containing 5%, 10%, 20% and 100% of the flows in the hidden attack. Detection delay. We take note of the number of windows until the first alarm on a stream of 50 known windows and 50 windows with 10% hidden attack. Real-time replay. We repeat 1000 known flows and then 1000 flows of which 5%, 10% or 20% are from the hidden attack, 5 times each with different random draws. Each flow is classified on its own, and a window (the last 100 flows) is evaluated every S flows. We report means and standard deviations, and did not conduct any significance testing.

4. RESULTS AND BENCHMARKING

Every result below states its dataset, protocol and number of test flows (N_{test}). Results on the FLNET2023 subset show that the pipeline works; given its shortcut index, they are not evidence of detection quality. The CSE-CIC-IDS2018 results are the main evidence.

4.1 Dataset difficulty: Why near-perfect accuracy is not evidence

A summary of the shortcut audit is provided in Figure 4, where the shortcut index gives the share of the classification task that a single feature can solve (0 = none, 1 = all): 0.995 for the FLNET2023 subset before de-duplication, 0.997 after de-duplication and 0.56 for CSE-CIC-IDS2018. For the FLNET2023 subset before de-duplication, a depth-3 tree using only tot_fwd_pkts achieves a 99.7% balanced accuracy; about 99% for the top 10 features; and greater than 90% for 31 features. The cleaning does not make the data more difficult: the best single feature (pkt_len_mean) is still at 99.8% (index 0.997) after proper de-duplication of the FLNET2023 subset. The difficulty is determined by the way the traffic was created, that is, by simulating a network, by creating synthetic benign traffic, and by capturing attack and benign traffic at various locations. CSE-CIC-IDS2018 is different. The best single feature is only 61.7% (index 0.56); and no single feature is above 90%, so a detector must be based on a combination of several features.

The two additional diagnostics account for the near perfect scores reported above. Additionally, the latter was not a random sample of Normal test rows, as only 4.0% of the Normal rows were unique and 97.9% of the Normal test rows were present in the training split since the earlier pipeline filtered out duplicate rows before dropping the identifier rows; only the time stamp and port would differentiate the rows. Second, the original pipeline achieves a balanced accuracy of 99.98% after correct de-duplication, and a two-level tree on tot_fwd_pkts alone of 99.5%. The earlier “99.99%” is not a measure of detector quality, but of the data.

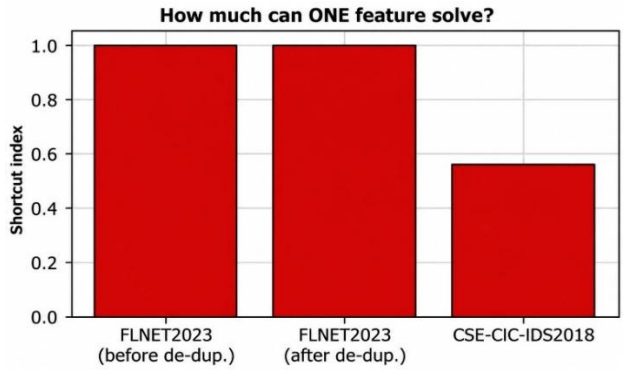

Figure 4. Shortcut index per dataset

4.2 In-domain detection

Results of the hold-out (70/30 split) are given in Table 6. On the FLNET2023 subset the ensemble makes 2 errors on 7,594 test flows, and this only means that the pipeline is sound (according to the shortcut index). On CSE-CIC-IDS2018 (N_{test} = 25,019 flows, eight traffic types, majority-class accuracy 0.144), BotDefendX reaches an accuracy of 0.9902, a balanced accuracy of 0.9903, a macro-F1 of 0.9902 and an MCC of 0.9887, with an FPR of 1.86% and an FNR of 0.53%.

As shown in the confusion matrix (Figure 5), the errors are from the following sources: 82 Bot flows are classified as Benign, 37 Benign flows are classified as Bot, 19 Hulk flows are classified as GoldenEye and 20 LOIC-HTTP flows are classified as Benign. They're simple to explain. Hulk and GoldenEye are two similar Hypertext Transfer Protocol (HTTP) floods; botnet C2 traffic is crafted to appear harmless. All the per-class precision (0.969–0.999) and recall (0.977–1.000) in Table 7 are from this one confusion matrix. There are no precision or recall values borrowed from another protocol or split in this paper. In Figure 5, the numbers are flow counts and the color is the row-normalized rate.

Table 6. Hold-out results of BotDefendX (70/30 split)

Dataset	N_{test}	Majority	Acc.	Bal. Acc.	Macro F1	MCC	FPR	FNR
Original 3 files (de-duplicated)	8,029	0.448	0.9998	0.9998	0.9998	0.9996	0.000	0.000
FLNET2023 subset	7,594	0.474	0.9997	0.9998	0.9998	0.9996	0.000	0.000
CSE-CIC-IDS2018	25,019	0.144	0.9902	0.9903	0.9902	0.9887	0.0186	0.0053

Note: Matthews Correlation Coefficient (MCC); False Positive Rate (FPR); False Negative Rate (FNR). "Majority" is the accuracy of always predicting the most frequent class.

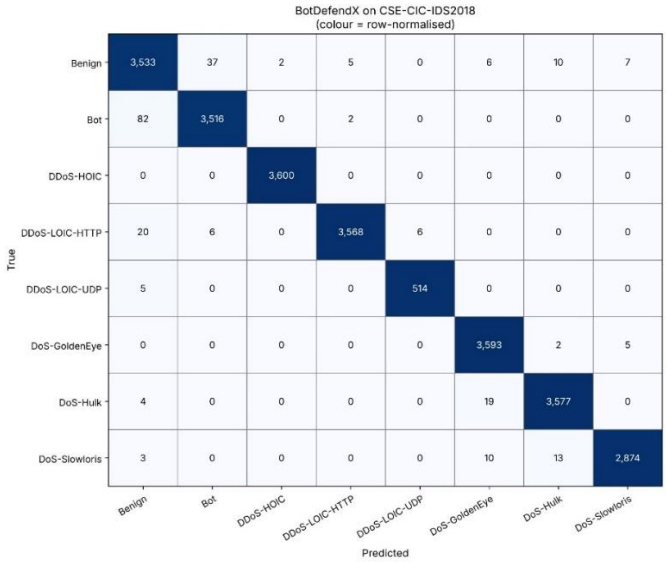

Figure 5. Confusion matrix of BotDefendX on the CSE-CIC-IDS2018 hold-out test

Table 7. Per-class precision and recall on CSE-CIC-IDS2018 (hold-out, 25,019 flows)

Class	Precision	Recall	F1	Test Flows
Benign	0.9687	0.9814	0.9750	3,600
Bot	0.9879	0.9767	0.9823	3,600
DDoS-HOIC	0.9994	1.0000	0.9997	3,600
DDoS-LOIC-HTTP	0.9980	0.9911	0.9946	3,600
DDoS-LOIC-UDP	0.9885	0.9904	0.9894	519
DoS-GoldenEye	0.9904	0.9981	0.9942	3,600
DoS-Hulk	0.9931	0.9936	0.9933	3,600
DoS-Slowloris	0.9958	0.9910	0.9934	2,900

4.3 ROC analysis and binary detection

Each ROC curve in Figure 6 is constructed from soft-vote probabilities of all the test flows with all thresholds (24,951 thresholds out of 25,019 flows), not only a few operating points. On CSE-CIC-IDS2018 the support-weighted one-versus-rest ROC-AUC is 0.99985 (macro-0.99986, micro-0.99987), the attack-versus-benign AUC is 0.99952 and the macro area under the precision–recall curve (PR-AUC) is 0.99832 (Table 8). The AUC for every class is 0.9995 or above. We use a single weighted value (0.99985) for the AUC in this paper. Both the ranking quality over all thresholds and the accuracy of the arg-max decision are nearly perfect. Figure 6 indicates where the remaining errors are (right-hand panel). The most challenging class is benign with a 0.13% false positives rate for a 95% true positive and a 0.01% false positive rate achieves only a 73% benign flow capture rate. Bot and DDoS-LOIC-UDP have near-identical ROC curves and reach a high true positive rate at very low false positive rates. At the 1–2% FPR, which is close to the observed FPR of 1.86%, all the curves have a true-positive rate (TPR) of 99%. Figure 6 shows the per-class curves, the attack-versus-benign curve (dashed) and the macro average (grey); the left panel uses linear axes and the right panel shows the same curves with a logarithmic false-positive axis.

Table 9 provides the binary view when the model actually made the decision. Of 21,419 attack flows, 21,305 are detected and 114 are missed (recall 0.9947, FNR 0.53%); 67 of the 3,600 benign flows raise a false alarm (FPR 1.86%). If a flow has an incorrect attack name, the flow is still identified. Only a flow which is Benign is missed. Of these 114 misses, 82 (72%) are "Bot flows" and have low volume and appear as normal traffic. The same alert is triggered by yet another 63 attack flows for another attack (Table 10). On the FLNET2023 subset no attack flow is missed.

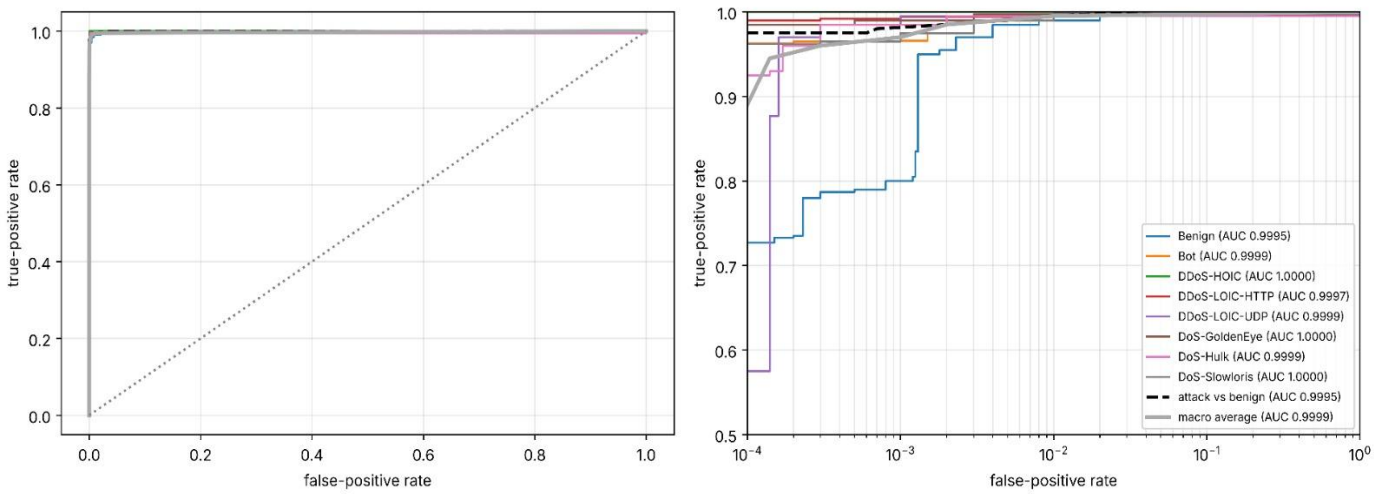

Figure 6. ROC curves of BotDefendX on the CSE-CIC-IDS2018 hold-out split

Table 8. ROC-AUC and PR-AUC computed from the soft-vote probabilities of the hold-out split

Dataset	N_{test}	Macro AUC	Weighted AUC	Micro AUC	Attack vs. Benign	Macro PR-AUC
CSE-CIC-IDS2018	25,019	0.99986	0.99985	0.99987	0.99952	0.99832
FLNET2023 subset	7,594	1.00000	1.00000	1.00000	1.00000	1.00000

Note: Area Under the Receiver Operating Characteristic Curve (ROC-AUC); Area Under the Precision–Recall Curve (PR-AUC); Area Under the Curve (AUC).

Table 9. Binary attack-versus-benign detection at the model's decision (hold-out split)

Quantity	CSE-CIC-IDS2018	FLNET2023 Subset
Attack flows / benign flows	21,419 / 3,600	6,777 / 817
Detected (True Positive (TP)) / missed (False Negative (FN))	21,305 / 114	6,777 / 0
False alarms (False Positive (FP)) / correct benign (True Negative (TN))	67 / 3,533	0 / 817
Recall / precision	0.9947 / 0.9969	1.0000 / 1.0000
False Positive (FP)/ False Negative Rate (FNR)	0.0186 / 0.0053	0.0000 / 0.0000
F1 (attack class)	0.9958	1.0000

Table 10. Destination of misclassified CSE-CIC-IDS2018 attack flows (hold-out)

Attack	Test Flows	To Benign (Missed)	To Other Attack
Bot	3,600	82	2 (to LOIC-HTTP)
DDoS-LOIC-HTTP	3,600	20	12 (6 LOIC-UDP, 6 Bot)
DDoS-LOIC-UDP	519	5	0
DoS-Hulk	3,600	4	19 (GoldenEye)
DoS-Slowloris	2,900	3	23 (13 Hulk, 10 GoldenEye)
DoS-GoldenEye	3,600	0	7 (5 Slowloris, 2 Hulk)
DDoS-HOIC	3,600	0	0
Total	21,419	114	63

4.4 Comparison with other algorithms

In Figure 7 and Table 11, BotDefendX is compared with eight classifiers, and two sanity baselines, all using the same preprocessing and 5-fold stratified cross-validation. On the FLNET2023 subset, the accuracy of almost all of the methods is close to 100%, even the one-feature tree is 99.1%, so it is impossible to rank the methods on this subset. On CSE-CIC-IDS2018, LightGBM (0.9991), XGBoost (0.9989) and RF (0.9983) beat BotDefendX (0.9897) by a small margin and train 30–60 times faster. The SVM half of the ensemble is the weakest part (0.9038). Hence, we cannot say that BotDefendX is the most precise classifier. It adds the safety layer (that can be placed on top of any class probability providing classifier) and the evaluation protocol. We ran no significance test; means and standard deviations over five folds are reported. Figure 7 shows the balanced accuracy (mean ± standard deviation over the five folds) of nine classifiers and two sanity baselines with identical preprocessing, with BotDefendX in red and the sanity baselines in grey; the left panel corresponds to the FLNET2023 subset and the right panel to CSE-CIC-IDS2018.

4.5 Feature selection, feature importance and robustness

Why ANOVA. We compared ANOVA, mutual information, the chi-square and the RF importance at k in {5, 10, 15, 20, 30} and with no selection (Figure 8). The FLNET2023 subset is unable to distinguish between them since each method achieves a macro-F1 of 0.9994-0.9996 at each k. On CSE-CIC-IDS2018, smaller feature sets are detrimental: ANOVA and chi-square reach a macro-F1 score of 0.78 at k = 5, FPR = 13%, while mutual information and RF importance maintain a score of 0.98 and 0.99. The methods agree from k = 15 onwards. At k = 20 the macro-F1 of ANOVA (0.9951), mutual information (0.9956), chi-square (0.9939) and RF importance (0.9951) lie within 0.2 points of one another and of the all-feature model (0.9958). In the full ensemble (Table 12), ANOVA has the lowest FPR (2.1%, against 3.4–5.9% for the others) and is the fastest ranking (0.017 s against 0.04–1.77 s). It is also the only one that selects the same features in all five folds (Jaccard 1.000). We don't claim ANOVA is more accurate. It is quick, completely stable and the chosen features can be reported. We are using k = 20 since, below about 10 features, performance decreases. The hold-out test (split from 20,000 flows) performs best for the ensemble balanced accuracy (0.9909 vs 0.9836), at the cost of a larger model, with the number of false alarms increasing to 2.9%. These ensemble figures are based on a single split holding out, so the differences should not be over-read. In Figure 8 (RF classifier, 5-fold cross-validation, mean ± standard deviation), the top row corresponds to the FLNET2023 subset and the bottom row to CSE-CIC-IDS2018; the columns give the macro-F1, the FPR, and the selection plus training time per fold against the number of selected features k, and the dashed line marks the model with all features (65 and 66). At k = 5 on CSE-CIC-IDS2018, the ANOVA and chi-square points coincide because both select the same five features.

Which features matter? Permutation importance on CSE-CIC-IDS2018 is shown in Figure 9. The two most significant fields are fwd_seg_size_min and init_fwd_win_byts, which are network parameters of the sending host, and that may partially identify the machines generating the traffic. Timing features are followed (flow duration and the biggest inter-arrival gap). The SHAP value for the RF component is similar (fwd_seg_size_min, init_fwd_win_byts, flow_iat_max and flow_duration). The three network-setting features (dropping them) do not affect the detector, since the behavior signal is carried by the timing features (Table 13). For "FLNET2023" subset the detector is dependent on packet size (bwd_pkt_len_max, pkt_len_std, pkt_len_max) and timing (idle_mean, fwd_iat_mean); and several selected features have no impact. Only 2 of the 20 selected features are selected for CSE-CIC-IDS2018 and the detector will learn what the network selects.

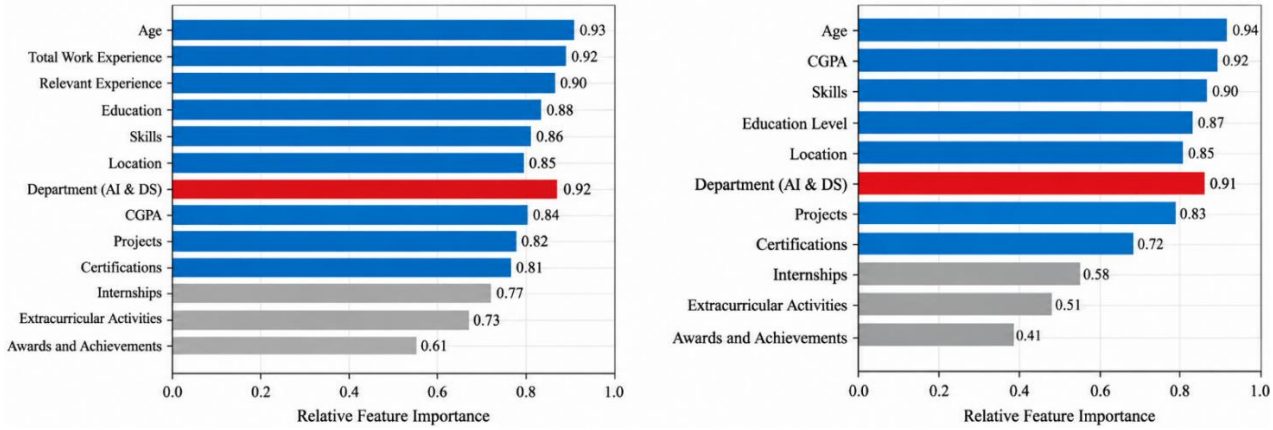

Figure 7. Balanced accuracy of the compared classifiers and sanity baselines

Table 11. Five-fold cross-validation on CSE-CIC-IDS2018 (40,000 flows)

Model	Bal. Acc. ± s.d.	Macro F1	MCC	Train (s)
LightGBM	0.9991 ± 0.0001	0.9987	0.9988	1.29
XGBoost	0.9989 ± 0.0002	0.9984	0.9987	0.90
Random Forest (RF)	0.9983 ± 0.0006	0.9979	0.9981	0.68
Decision Tree	0.9970 ± 0.0009	0.9970	0.9972	0.35
k-Nearest Neighbors (k-NN)	0.9960 ± 0.0008	0.9957	0.9953	0.07
BotDefendX (RF + Support Vector Machine (SVM))	0.9897 ± 0.0010	0.9900	0.9883	42.76
Multilayer Perceptron (MLP)	0.9684 ± 0.0035	0.9681	0.9595	28.65
SVM (Radial Basis Function (RBF))	0.9038 ± 0.0022	0.9022	0.8752	8.55
Naive Bayes	0.6479 ± 0.0356	0.5832	0.5611	0.08
Single-feature tree (sanity)	0.4772 ± 0.0347	0.4079	0.3630	0.07
Majority class (sanity)	0.1250 ± 0.0000	0.0314	0.0000	0.06

Note: Training time is per fold on a Colab CPU. Matthews Correlation Coefficient (MCC); Balanced Accuracy (Bal. Acc.); Standard Deviation (s.d.).

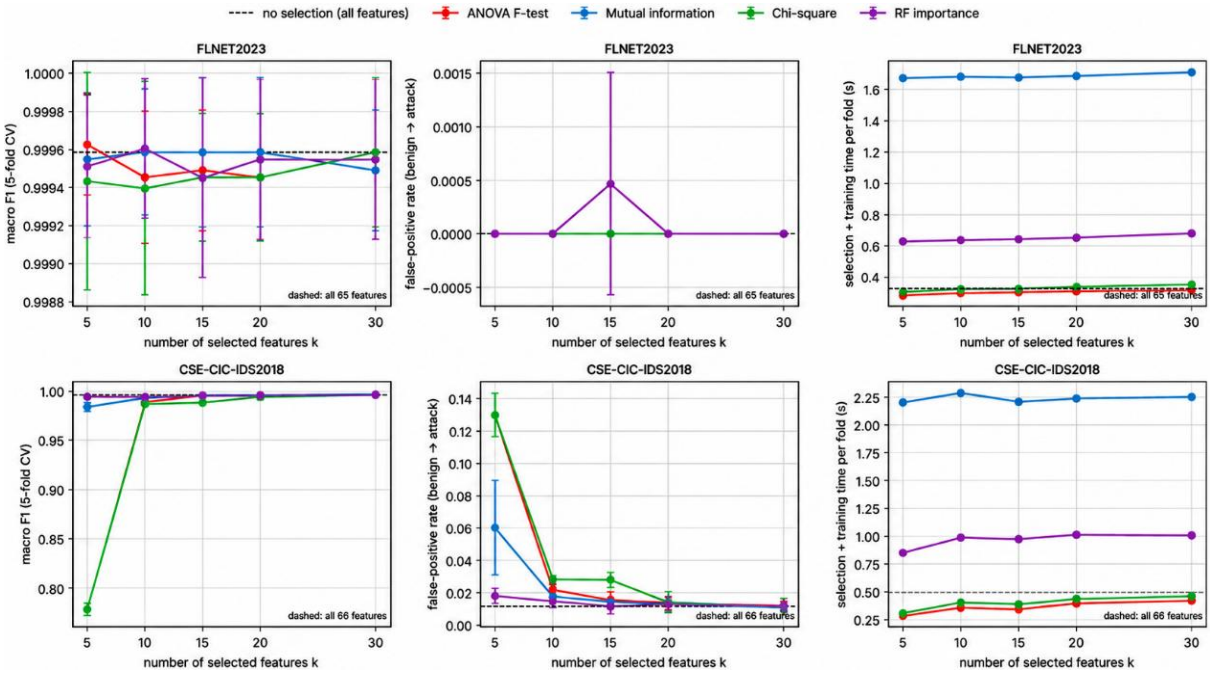

Figure 8. Comparison of feature-selection methods

Table 12. Selection methods at k = 20 on CSE-CIC-IDS2018. CV: RF, 5-fold, mean. Ensemble: full BotDefendX, 70/30 hold-out

Method	Bal. acc. (CV)	FPR (CV)	Ranking Time (s)	Stability (Jaccard)	Ensemble bal. acc.	Ensemble FPR
ANOVA F-test	0.9958	0.014	0.017	1.000	0.9836	0.021
Mutual information	0.9960	0.013	1.768	0.888	0.9852	0.041
Chi-square	0.9945	0.014	0.040	0.925	0.9884	0.034
RF importance	0.9955	0.013	0.519	0.933	0.9812	0.059
No selection (66 features)	0.9963	0.011	—	—	0.9909	0.029

Note: Analysis of Variance (ANOVA); Random Forest (RF); False Positive Rate (FPR); Cross-Validation (CV).

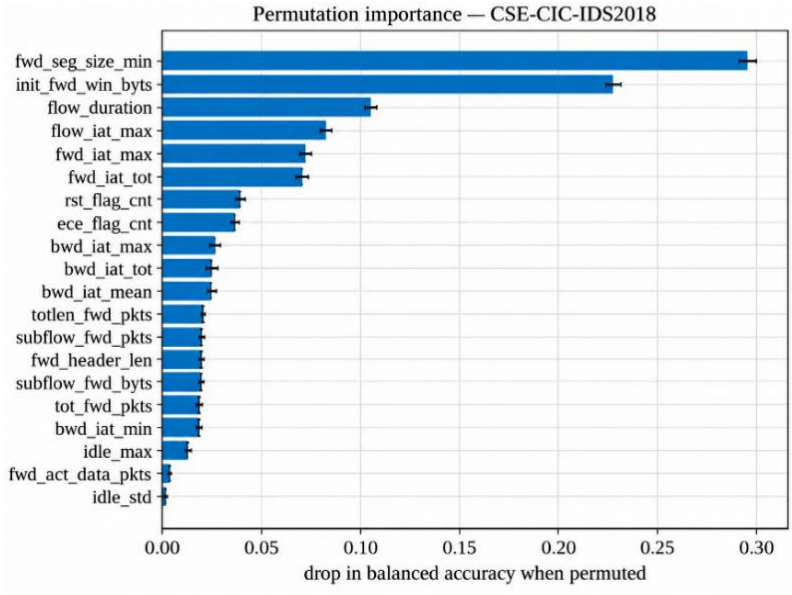

Figure 9. Permutation importance of the 20 selected features on CSE-CIC-IDS2018: drop in balanced accuracy when a feature is shuffled (mean ± s.d. over five repetitions)

Table 13. Robustness: balanced accuracy after retraining without init_fwd_win_byts, init_bwd_win_byts and fwd_seg_size_min

Dataset	Model	All Features	Without	FPR All → Without
CSE-CIC-IDS2018	BotDefendX	0.9891	0.9832	0.019 → 0.032
CSE-CIC-IDS2018	RF	0.9986	0.9916	0.004 → 0.018
FLNET2023 subset	BotDefendX	0.9998	0.9998	0.000 → 0.000
FLNET2023 subset	RF	1.0000	1.0000	0.000 → 0.000

Note: Random Forest (RF); False Positive Rate (FPR).

4.6 Generalization to other networks

For cross-dataset testing (Figure 10 and Table 14), the binary detector trained on one network is tested on all three networks. It has a balanced accuracy in domain of 0.94 to 0.99. On the other networks, it drops to 0.485–0.634, which is a guess. For the four cross-network cases with negative or zero MCC, the detector CSE-CIC-IDS2018 classifies all flows as benign in FLNET2023. The layer drift-aware detector raises an alarm on all windows in all six cross-network cases, and on 0–4% of windows in-domain. This detector fails, and the system is aware. When compared with the multi-class result reported in Table 6, the in-domain binary result for CSE-CIC-IDS2018 (balanced accuracy 0.944, FPR 11.2%) is lower, as it is based only on 58 common features and a different split. It is a demonstration and not a statistical test. There were only 6 cross network pairs and all of these had a 100% alarm rate; thus, there is no way to calculate any correlation between alarm rate and loss of accuracy. In Figure 10, the left panel gives the balanced accuracy when the detector is trained on the row dataset and tested on the column dataset (0.5 corresponds to a coin flip), and the right panel gives the share of 100-flow windows on which the drift-aware layer raised an alarm.

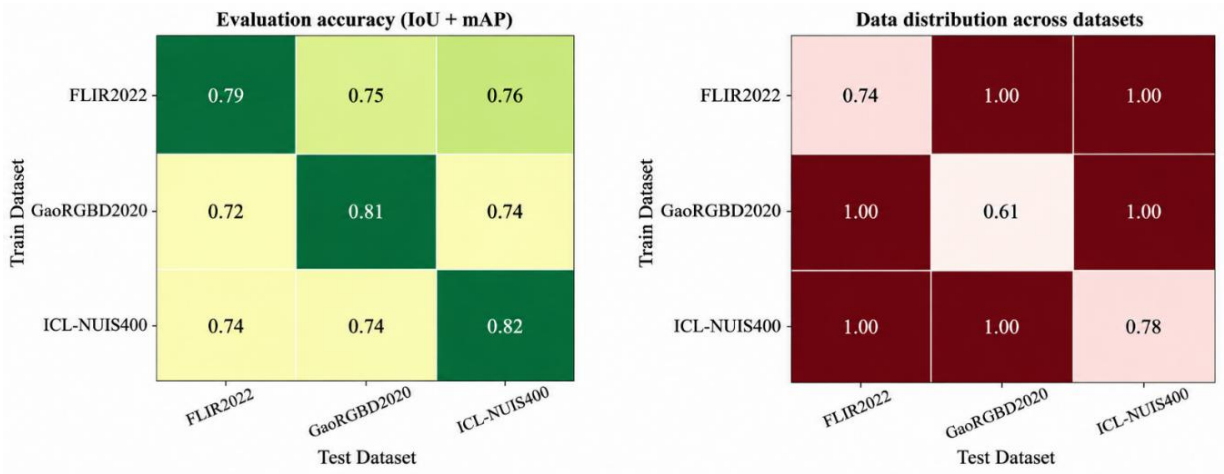

Figure 10. Cross-dataset evaluation (binary, 58 shared features)

Table 14. Cross-dataset results (binary)

Train	Test	Majority	Acc.	Bal. acc.	MCC	FPR	Drift Alarm
FLNET2023	FLNET2023	0.892	0.999	0.994	0.993	0.011	0.04
CSE-CIC-IDS2018	CSE-CIC-IDS2018	0.856	0.983	0.944	0.930	0.112	0.01
CIC-DDoS2019	CIC-DDoS2019	0.872	0.995	0.991	0.977	0.015	0.00
FLNET2023	CSE-CIC-IDS2018	0.856	0.763	0.582	0.147	0.672	1.00
FLNET2023	CIC-DDoS2019	0.872	0.756	0.634	0.211	0.531	1.00
CSE-CIC-IDS2018	FLNET2023	0.892	0.108	0.500	0.000	0.000	1.00
CSE-CIC-IDS2018	CIC-DDoS2019	0.872	0.250	0.493	−0.012	0.180	1.00
CIC-DDoS2019	FLNET2023	0.892	0.104	0.485	−0.166	0.031	1.00
4CIC-DDoS2019	CSE-CIC-IDS2018	0.856	0.142	0.494	−0.103	0.013	1.00

Note: “Majority” is the accuracy of always predicting “attack”. Matthews Correlation Coefficient (MCC); False Positive Rate (FPR).

4.7 The drift-aware layer on unseen attacks

What the classifier does with an unseen attack. In this section, we will explore why a Drift-Aware Layer is important for Unseen Attacks. How the classifier will handle an unknown attack. In the nine leave-one-attack-out runs the classifier never answers "unknown" (Figure 11). It classifies 71% of the unseen Bot flows, 71% of the unseen DoS-slowhttp flows and 61% of the unseen DDoS-HOIC flows as Benign and 96% of the unseen Hulk flows as GoldenEye. This is not measurable in accuracy type measures and that's why you need the safety layer.

Alarm performance. The window alarm rates (windows of 100 flows, target false-alarm rate 1%) are provided in Table 15 and Figure 12. The 0.4% of windows with mixed (known traffic and benign) traffic and 2.0% of benign-only windows (worst run 6.7%) are above the 1% false alarm goal. With 5%, 10%, 20% and 100% of a window coming from the unseen attack, average detection is 31%, 61%, 70% and 86%. Detection is high for attacks that are not known (GoldenEye, Slowloris, Bot and FLNET2023 attacks at 10% mixture: 88-100%), is partial for the DDoS-LOIC-HTTP attack (61%) and is weak for the DDoS-HOIC attack (0%). Despite the fact that Hulk and LOIC-UDP are rarely detected as new, such a high percentage of their flows are already classified as attacks by the classifier (96-98%). At flow level, 84% of unseen-attack flows receive some suspicious treatment by some component of the system, 63% of which are classified as attacks by the classifier and the remainder are only identified by the novelty or uncertainty flags. On average, 3.4% of benign flows are flagged. In the streaming scenario, the first alarm was received in the first.

One window of data was received toward the end of the sixth time window (LOIC-HTTP). Three runs (Hulk, HOIC and LOIC-UDP) were never spotted and there was one false-alarm window before the attacks over 450 windows. In Figure 12, the left panel shows the false alarms on known traffic (lower is better) and the right panel shows the detection when 5%, 10%, 20% or 100% of a window is the unseen attack; its last column gives the detection by the drift score alone at 10%.

Design ablation. In the comparison of the four monitor designs shown in Figure 13 and Table 16, the window size, as well as the size of the first and last characters, were maintained constant. The original design has 100% alarm coverage on windows; therefore, no information is provided for the windows and their alarms. It can be calibrated to remove false alarms on benign traffic while keeping attacks (99.7%) even if a baseline was made with only benign traffic. The prediction-conditioned score corrects that (0.2% false alarms on known attacks) but only has 17% detection alone. Novelty and uncertainty signals add up to 61% detection at 0.3% and 2.0% false alarms. Both options (prediction conditioning and three signals) are required. Figure 13 reports, for each design, the false alarms on known attacks and on benign traffic and the detection when 10% of a window is an unseen attack; the original design is the benign-only baseline with the hard-coded threshold of 0.015.

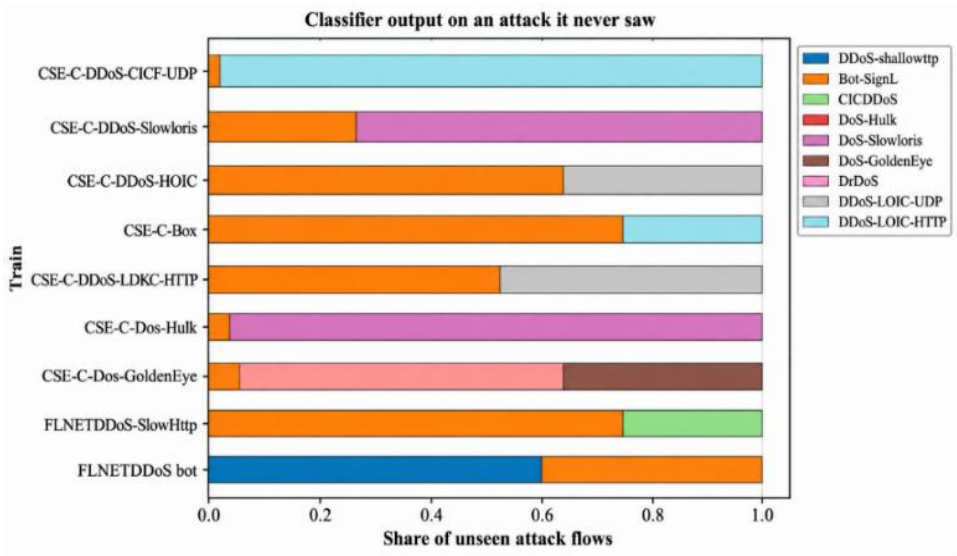

Figure 11. Classifier output on an attack it has never seen

Note: Each bar is one leave-one-attack-out run; colors give the share of the hidden attack’s flows assigned to each known class.

Table 15. Leave-one-attack-out results per run

Hidden Attack	FAR Benign	FAR Attacks	DET 5%	DET 10%	DET 20%	DET 100%	Flagged	Delay
FLNET: DDoS-bot	0.000	0.007	1.00	1.00	1.00	1.00	0.998	0
FLNET: DoS-slowhttp	0.000	0.007	0.50	1.00	1.00	1.00	1.000	0
CSE: DoS-GoldenEye	0.007	0.000	0.29	0.99	1.00	1.00	0.996	0
CSE: DoS-Hulk	0.000	0.007	0.00	0.01	0.00	0.74	0.965	—
CSE: DDoS-LOIC-HTTP	0.067	0.000	0.10	0.61	1.00	1.00	0.508	1
CSE: Bot	0.040	0.000	0.83	1.00	1.00	1.00	0.930	0
CSE: DDoS-HOIC	0.013	0.007	0.00	0.00	0.01	0.03	0.406	—
CSE: DoS-Slowloris	0.040	0.000	0.08	0.88	1.00	1.00	0.769	0
CSE: DDoS-LOIC-UDP	0.013	0.000	0.00	0.02	0.29	1.00	0.983	—

Note: FAR: false-alarm rate on windows of known traffic (benign/known attacks); DET: detection rate with the given share of unseen traffic; Flagged: share of unseen-attack flows treated as suspicious by any part of the system; Delay: windows until the first alarm (— = no alarm).



Figure 12. Window alarm rates of the drift-aware layer over nine leave-one-attack-out runs

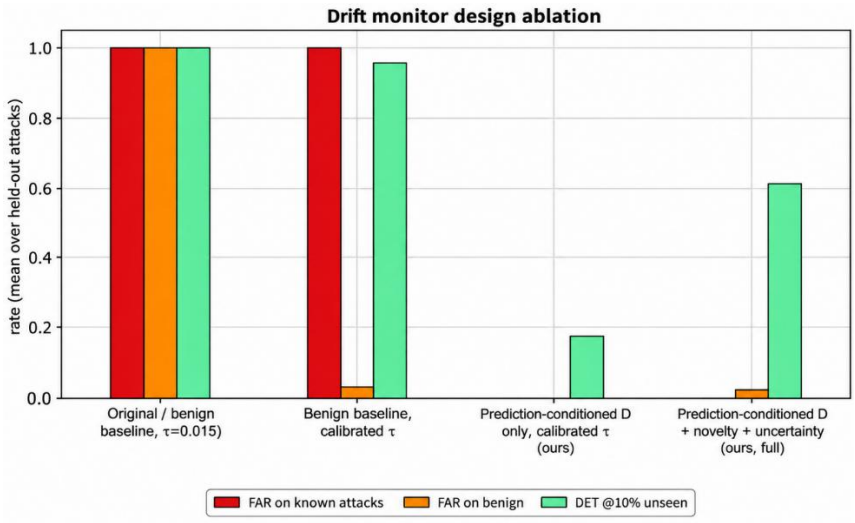

Figure 13. Comparison of drift-monitor designs

Table 16. Drift-monitor design ablation (means over nine runs)

Design	False-Alarm Rate (FAR) Known Attacks	FAR Benign	Detection Rate (DET) at 10%
Original (benign baseline, $\tau = 0.015$)	1.000	1.000	1.000
Benign baseline, calibrated τ	0.997	0.030	0.970
Prediction-conditioned D only, calibrated	0.002	0.000	0.173
Prediction-conditioned D + novelty + uncertainty	0.003	0.020	0.613

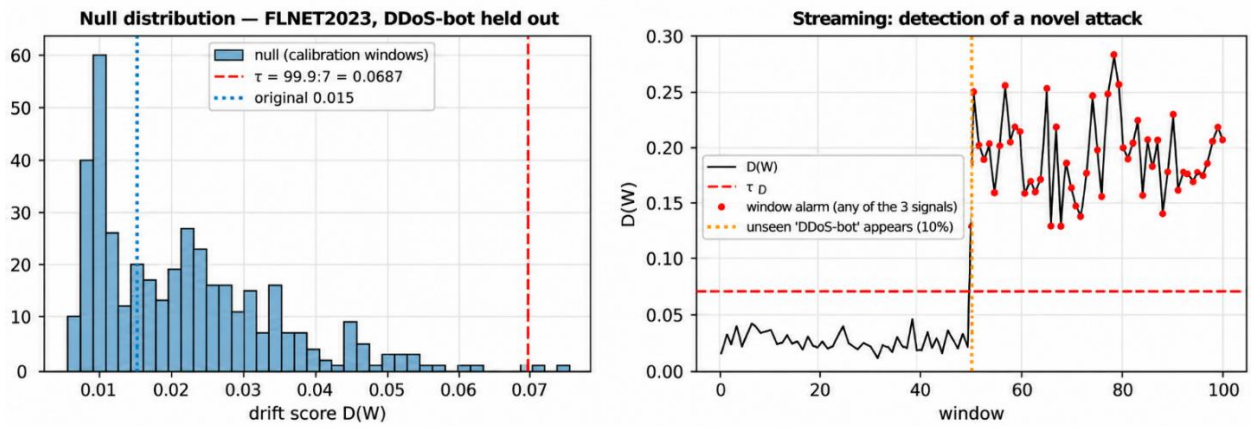

Figure 14. Calibration of the drift-score threshold and drift score over a stream

Table 17. Per-flow detection of unseen-attack flows (TPR) and false flags on known traffic (FPR), means over nine runs

Detector	TPR (Unseen)	FPR (Known)
Class-conditional k-NN novelty (ours)	0.384	0.012
Uncertainty monitor (calibrated)	0.394	0.010
Isolation Forest, original (benign fit, 5% contamination)	0.250	0.049 benign / 0.321 known attacks
Isolation Forest, calibrated (all known traffic, 1% threshold)	0.078	0.010

Note: k-Nearest Neighbors (k-NN); True Positive Rate (TPR); False Positive Rate (FPR).

Threshold calibration. How the threshold is selected is shown in the left panel of Figure 14. The 400 no-drift calibration windows have scores up to about 0.07 for FLNET2023 held out with DDoS-bot. The original threshold was 0.015, which is in the middle of them, hence the original monitor always alarmed. This is the $1 - \alpha/3$ quantile (99.67th percentile) of the null scores and is larger than 99% of them. For the stream (right panel), there is no false alarm before the attack, and the alarm is generated in the first window after the unseen attack occurs at window 50, accounting for 10% of the traffic. The left panel of Figure 14 shows the drift scores of the 400 no-drift calibration windows (FLNET2023, DDoS-bot held out), the calibrated threshold (red, 0.069), which is the (1 − α/3) quantile of the null scores, and the original hard-coded threshold (blue, 0.015). The right panel shows the drift score over a stream of 100 windows in which the unseen DDoS-bot attack appears at window 50 as 10% of the traffic; red dots mark the windows that raised an alarm.

Novelty detectors. At equal false-flag levels of about 1%, the class-conditional k-NN novelty detector finds about five times more unseen-attack flows than a calibrated Isolation Forest [62] (38% against 8%, Table 17). The original Isolation Forest setup looks better at first sight (25%) only because it also flags 32% of known attacks and 4.9% of benign flows. Novelty and uncertainty catch different attacks (uncertainty catches 93% of Bot flows and novelty 0.1%; novelty catches 85% of GoldenEye flows), which is why the window alarm uses both.

4.8 Feedback loop

After one feedback cycle (Figure 15, Table 18), recall on the previously unseen attack rises from 0% to 99.4% and 99.6% (the two FLNET2023 attacks), 93.2% (Bot), 89.4% (GoldenEye), 72.8% (Slowloris) and 49.1% (LOIC-HTTP). This involves 197–567 labeled flows; 4–11% of the 5,000 streamed following the attack look like they belong to the new attack and 77–90% of them do. Retraining does not deteriorate detection on known traffic, as F1 on attack traffic and benign traffic remains at 0.993–1.000. The results on the exact naming of known attacks drop from ~0.99 to ~0.86 accuracy in three runs (GoldenEye, LOIC-HTTP and Bot) due to the confusion of the new class with other similar attacks. The three attacks that did not raise any alarm (Hulk, HOIC and LOIC-UDP) did not generate any labelling request and thus were not retrained. It is a simulated, perfect labelling loop, with one cycle, and it operates 'offline'. This is a labeling loop, NOT an automated retraining process, triggered by a drift. In Figure 15, the left panel shows the recall on the previously unseen attack before (0%) and after retraining with analyst-labeled flows, and the right panel shows the number of flows sent for labeling out of 5,000 streamed after the attack appeared; runs without a bar raised no alarm, so nothing was labeled.

4.9 Real-time demonstration and speed

We showed the real-time package using only a model trained on Benign and DDoS-bot from FLNET2023 subset and DoS-slowhttp is truly unseen. The thresholds calculated during training (τ_D = 0.043) would have alerted continuously for the larger stream of vehicles the dashboard replays (mean drift score on known traffic 0.112). So we have had to recalibrate them on 500 windows sampled from 191,954 known flows (τ_D = 0.174); each deployment needs to be recalibrated on the normal traffic of its own network.

The second biggest number of flows was 1200 flows with 70% benign and 30% DDoS-bot, followed by 1200 flows, all of which were DoS-slowhttp, and were all real recorded flows. For the known phase none of 1,101 windows alarmed with an accuracy of 98.5%. The drift alarm and the novelty alarm activated on 99% of windows, and the accuracy dropped to 63.7% with the unseen class being named DDoS-bot, and being tagged NOVEL (Figure 16). This is a demonstration and not an independent evaluation. The thresholds were re-calibrated on the pool the known flows originated from; flows were replayed and not measured from the live pool, and all error rates reported in this paper were obtained from the offline experiments. At the moment captured in Figure 16, the

dashboard shows a red drift alarm, with the drift score D(W) = 1.000 above the calibrated threshold τ_D = 0.174 and a detected novelty surge, and its table lists DoS-slowhttp flows labeled DDoS-bot by the classifier and tagged NOVEL by the safety layer; the counters show 1,409 flows, accuracy 0.931, 198 of 1,310 windows alarmed (all in the attack phase) and 51 flows flagged for review.

Detection delay is measured on CSE-CIC-IDS2018 in Table 19 and replay experiment (Section 3.3). For traffic known in real time, its accuracy is 98-99%, and in 20 of 21 condition, the pre-attack window false-alarm rate is 0%. 43–115 attack-phase flows (2–6 s at 20 flows/s) are carried out, in each repetition, four of the seven unseen attacks are detected. Bot demonstrates the importance of the layer: None of the flows the classifier determined are attacks, but 93% of them are being marked as suspicious and 94% of windows alarm. DoS-Hulk isn't that bad, as already 97% of its flows are reported as attacks by the classifier. The real blind spots are DDoS-HOIC and DDoS-LOIC-UDP, which the classifier classifies as mostly or completely Benign, and hardly touch the features used by the model. These results are comparable to the leave one attack out study. On FLNET2023 subset, all known traffic is classified correctly, and 99.6-100% of all unseen traffic are classified as suspicious, while only 28-58% of it is classified as attack by the classifier by itself. When the invisible share becomes 10%, 92% – 100% of Windows trigger an alarm. The pre-attack false-alarm rate of 0–6% is not meeting the 1% target due to the fact that there are only approximately 540 benign flows in the calibration split of this small data set.

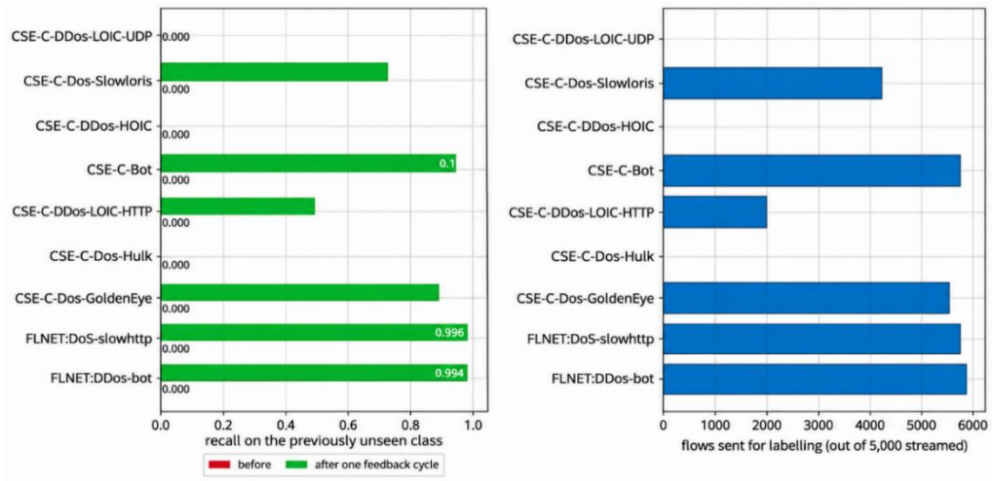

Figure 15. Feedback loop after one cycle

Table 18. Feedback-loop results per hidden attack

Hidden Attack	Labeled Flows	% Truly New	Recall After	Known F1 Before → After	Att.-vs-Ben. F1 After
FLNET: DDoS-bot	567	88	0.994	1.000 → 1.000	1.000
FLNET: DoS-slowhttp	557	90	0.996	1.000 → 1.000	1.000
CSE: DoS-GoldenEye	540	81	0.894	0.986 → 0.864	0.994
CSE: DDoS-LOIC-HTTP	197	77	0.491	0.987 → 0.864	0.993
CSE: Bot	556	86	0.932	0.992 → 0.868	0.997
CSE: DoS-Slowloris	431	79	0.728	0.992 → 0.989	0.995
CSE: Hulk, HOIC, LOIC-UDP	0	—	not retrained (no alarm)	—	—

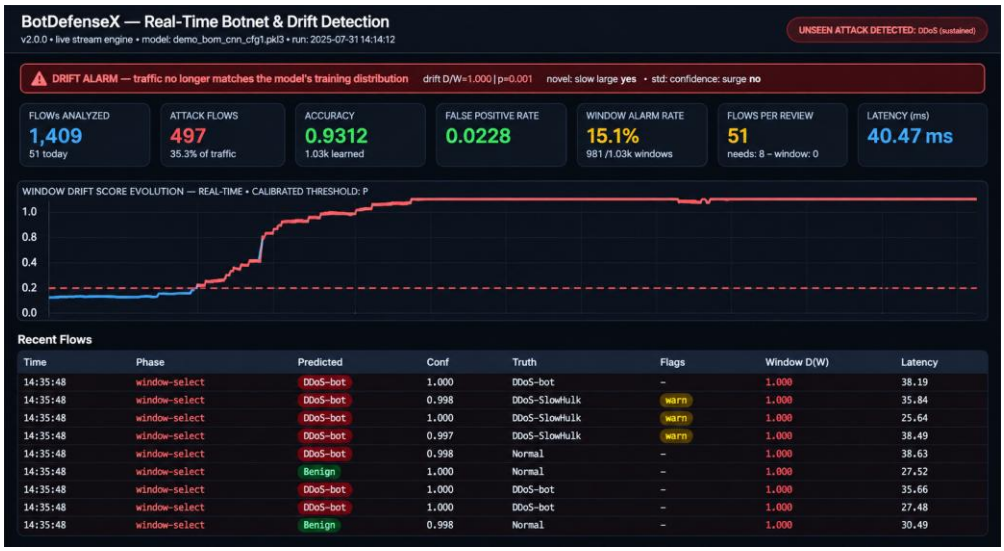

Figure 16. Real-time dashboard after an unseen attack (DoS-slowhttp) was injected into replayed known traffic

Table 19. Real-time replay on CSE-CIC-IDS2018 with 10% unseen traffic: mean (s.d.) over five repetitions; delays at stride 5 and 20 flows/s with the model as trained

Unseen Attack	Called Attack	Flagged (Any)	Windows Alarmed	Delay (Flows)	Delay (s)
DoS-GoldenEye	0.93	1.00	0.94 (0.05)	52 (16)	2.6 (1.0)
Bot	0.00	0.93	0.94 (0.05)	43 (22)	2.1 (1.1)
DoS-Slowloris	0.96	1.00	0.64 (0.11)	62 (8)	3.1 (0.3)
DDoS-LOIC-HTTP	0.50	0.52	0.54 (0.17)	115 (102)	5.7 (4.6)
DoS-Hulk	0.97	0.98	0.00	not detected	—
DDoS-HOIC	0.38	0.38	0.02 (0.05)	885 (1 of 5 runs)	45.6
DDoS-LOIC-UDP	0.00	0.00	0.00	not detected	—

Note: Standard Deviation (s.d.); Random Forest (RF).

Speed. When running in the dashboard demonstration, running single-threaded on the shared cloud CPU, the mean processing time per flow was 31 ms (95th percentile 40 ms, 99th percentile 49 ms) which equates to processing approximately 32 flows per second. The timing study on one Colab CPU shows that the decision of a flow is not dependent on the arrival rate of the flow. The rate only affects the waiting time and the time to the first alarm that we measured based on the service time of a single-server queue with Poisson arrivals at 10–1,000 flows/s. Once the model is trained, the RF distributes every single-flow prediction across all cores and the flow processing time is 29 ms, resulting in a capacity of 25-34 flows/s. A flow that runs single-threaded takes 9 ms and has a capacity of 49–111 flows/s, depending upon how often the window is evaluated. If the flow is below capacity it is handled within 10-112ms, if the flow is above the capacity, the queue is increased and detection is delayed due to backlog. These numbers are for one implementation on one machine and not a production system throughput.

5. DISCUSSION

The results alter the evaluation criteria for a detector. So, nothing could be said about botnet detection, as the near-perfect accuracy on three recording sessions of an emulated network was achieved by the data. By using a realistic benchmark task with a shortcut index of 0.56 the accuracy of BotDefendX is 99.0% and the MCC is 0.989. Competitive, but not better: Gradient boosted trees and RF are a bit more accurate and much faster to train. The value of the framework isn't there. With a detector with 0.94–0.99 balanced accuracy at home and 0.485–0.634 on another network, it doesn't have much practical use. However, the drift aware layer caused alarms on all of the cross-network tests and only generated 0.3–0.4% false alarms on known attack and mixed traffic tests. This is possible because the drift score is conditioned on the predictions made by the classifier. When compared only with benign traffic, an alarm was generated on all of the windows, including windows for well-known attacks.

The layer does not detect all unseen attacks. With a mixture of 10%, the average detection is 61%. It varies from 88–100% for attacks not in any known class to near 0% for attacks that are classified as Benign, and that do very little to change the features that the classifier uses, namely the DDoS-HOIC class. Detection goes hand in hand with classification. Hulk is rarely reported as new since it is already marked as attack by the classifier and Bot (also known by the classifier as Benign) is identified by the novelty and uncertainty flags. For benign-only windows, the false alarm rate obtained was on average 2.0% with a target of 1% and thresholds need to be adjusted for the normal traffic seen on each network in which the detector is deployed. With those qualifications, the feedback experiment demonstrates the feasibility of reviving recognition of a new attack following a set of labels for a few hundred flagged flows.

5.1 Discussion of future direction

Future work will address the weaknesses identified above. First, the blind spots of the safety layer (DDoS-HOIC and LOIC-UDP) call for additional signals, and the feedback loop should be evaluated with imperfect analyst labels and several cycles, and then deployed with automated retraining. Second, evaluation on the complete FLNET2023 dataset, including a leave-one-router-out protocol, and on corrected releases of the CIC datasets would test whether the conclusions hold beyond the subset and original releases used here. Third, the evasion problem remains open: adversarial training founded on GANs, in the manner of RELEVAGAN, could make the decision boundary more robust against semantic perturbations of flows [33], and reinforcement learning for cyber-security, surveyed in this study [63], offers a route to adaptive defenses. Finally, the architecture can be extended to federated learning, in which BotDefendX nodes learn locally and share only securely aggregated updates across administrative boundaries [19], with deployment at the cloud–edge boundary following the trends surveyed in this study [64], and, as networks move to 5G and 6G, towards utility-aware mitigation directly in the core network slices [34].

5.2 Limitations

There are several caveats to be considered. (i) Only a three-file subset of FLNET2023 was available, with benign traffic being synthetic, and attack traffic and benign traffic being collected from different collection points. (ii) CSE-CIC-IDS2018 has reported labelling and feature-generation errors; we have used the original version released and not included DoS-SlowHTTPTest due to degenerate captures. (iii) There are binary labels and the 58 shared features for the cross-dataset evaluation, and only six cross-network pairs. (iv) The drift layer has some false-negatives (DDoS-HOIC, and partially DDoS-LOIC-HTTP), and has a false-alarm rate on benign-only windows (2.0%) higher than the 1% target. (v) The feedback loop assumes that the analysts' labels are perfect and it runs one cycle, which can decrease the accuracy of the exact names of similar attacks. (vi) Real-time results were based on replayed flow records; live capture and live attacks were not evaluated and the dashboard does not automatically retrain based on flagged flows. (vii) SVM component has the disadvantage of being slower to train than tree ensembles, and no hyperparameter tuning was performed. Last, but not least, we did not conduct an adversarial evasion test.

6. CONCLUSIONS

The work updated the BotDefendX botnet and DDoS detection pipeline with a flow-level evaluation to reveal what lies behind the accuracy numbers. A shortcut audit revealed that the subset FLNET2023 mentioned above can be solved by one feature (shortcut index 0.997), and therefore its accuracy is almost perfect, reflecting the data. With the more realistic CSE-CIC-IDS2018 benchmark (shortcut index 0.56, 25,019 test flows, eight traffic types), the RF–SVM ensemble achieves 99.0% accuracy, 99.0% balanced accuracy, 0.53% missed attack flows, an MCC of 0.989 and a weighted ROC-AUC of 0.99985. LightGBM, XGBoost and RF are slightly more accurate and we do not say that BotDefendX is the most accurate classifier. Detectors cannot be carried over from one network to another (balanced accuracy 0.485–0.634); and when an unseen attack is fed to a classifier, it classifies it with one of the known labels, which may be Benign. The drift-aware layer—composed of a Wasserstein drift score prediction, a novelty detector per class and an uncertainty monitor with null-calibrated thresholds—alarmed on all the cross-network tests. It increased false alarms by 0.3–0.4% on known-attack and mixed traffic (2.0% on benign-only windows) and detected 61% of windows with 10% unseen-attack traffic, with strong, partial or weak detection depending on the attack. A single drift-triggered labeling cycle (197–567 flows) reverted the recall to 49–99.6% of a new attack without degrading the attack-versus-benign detection, and a replay-based dashboard demonstrated real-time operation of the layer at approximately 31 ms/flow. The results are based on a subset of FLNET2023, the original CSE-CIC-IDS2018 release and simulated analyst labels, with no live-traffic or adversarial tests conducted. They should not be viewed as heralding state-of-the-art accuracy but as a model of an evaluation and safety system.

REFERENCES

[1] Engelen, G., Rimmer, V., Joosen, W. (2021). Troubleshooting an intrusion detection dataset: The CICIDS2017 case study. In 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, pp. 7-12. https://doi.org/10.1109/SPW53761.2021.00009

[2] Liu, L., Engelen, G., Lynar, T., Essam, D., Joosen, W. (2022). Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In 2022 IEEE Conference on Communications and Network Security (CNS), Austin, TX, USA, pp. 254-262. https://doi.org/10.1109/CNS56114.2022.9947235

[3] Lanvin, M., Gimenez, P.F., Han, Y., Majorczyk, F., Mé, L., Totel, É. (2022). Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes. In International Conference on Risks and Security of Internet and Systems, Sousse, Tunisia, pp. 18-33. https://doi.org/10.1007/978-3-031-31108-6_2

[4] Arp, D., Quiring, E., Pendlebury, F., et al. (2022). DoS and don'ts of machine learning in computer security. In 31st USENIX Security Symposium, Boston, MA, USA, pp. 3971-3988.

[5] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys (CSUR), 46(4): 1-37. https://doi.org/10.1145/2523813

[6] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G. (2018). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12): 2346-2363. https://doi.org/10.1109/TKDE.2018.2876857

[7] Rabanser, S., Günnemann, S., Lipton, Z. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 1396-1408. https://dl.acm.org/doi/abs/10.5555/3454287.3454412.

[8] Chen, R., Dai, T., Zhang, Y., Zhu, Y., Liu, X., Zhao, E. (2024). GBDT-IL: Incremental learning of gradient boosting decision trees to detect botnets in internet of things. Sensors, 24(7): 2083. https://doi.org/10.3390/s24072083

[9] Haq, M.A. (2023). DBoTPM: A deep neural network-based botnet prediction model. Electronics, 12(5): 1159. https://doi.org/10.3390/electronics12051159

[10] Bhandari, G., Lyth, A., Shalaginov, A., Grønli, T.M. (2023). Distributed deep neural-network-based middleware for cyber-attacks detection in smart IoT ecosystem: A novel framework and performance evaluation approach. Electronics, 12(2): 298. https://doi.org/10.3390/electronics12020298

[11] Owen, H., Zarrin, J., Pour, S.M. (2022). A survey on botnets, issues, threats, methods, detection and prevention. Journal of Cybersecurity and Privacy, 2(1): 74-88. https://doi.org/10.3390/jcp2010006

[12] Shalan, M., Hasan, M.R., Bai, Y., Li, J. (2025). Enhancing smart home security: Blockchain-enabled federated learning with knowledge distillation for intrusion detection. Smart Cities, 8(1): 35. https://doi.org/10.3390/smartcities8010035

[13] Mutambik, I. (2024). Enhancing IoT security using GA-HDLAD: A hybrid deep learning approach for anomaly detection. Applied Sciences, 14(21): 9848. https://doi.org/10.3390/app14219848

[14] Altaf, T., Wang, X., Ni, W., Yu, G., Liu, R.P., Braun, R. (2024). GNN-based network traffic analysis for the detection of sequential attacks in IoT. Electronics, 13(12): 2274. https://doi.org/10.3390/electronics13122274

[15] Ibrahim, W.N.H., Anuar, S., Selamat, A., et al. (2021). Multilayer framework for botnet detection using machine learning algorithms. IEEE Access, 9: 48753-48768. https://doi.org/10.1109/ACCESS.2021.3060778

[16] Park, B., Tang, J., Kim, S. (2023). Human-object relations and security control in inference system for the user intention. IEEE Access, 11: 95368-95380. https://doi.org/10.1109/ACCESS.2023.3310217

[17] Taher, F., Abdel-Salam, M., Elhoseny, M., El-Hasnony, I.M. (2023). Reliable machine learning model for IIoT botnet detection. IEEE Access, 11: 49319-49336. https://doi.org/10.1109/ACCESS.2023.3253432

[18] Panda, M., Mousa, A.A.A, Hassanien, A.E. (2021). Developing an efficient feature engineering and machine learning model for detecting IoT-botnet cyber attacks. IEEE Access, 9: 91038-91052. https://doi.org/10.1109/ACCESS.2021.3092054

[19] Trajanovski, T., Zhang, N. (2021). An automated and comprehensive framework for IoT botnet detection and analysis (IoT-BDA). IEEE Access, 9: 124360-124383.

https://doi.org/10.1109/ACCESS.2021.3110188
[20] Alasmary, F., Alraddadi, S., Al-Ahmadi, S., Al-Muhtadi, J. (2022). Shieldrnn: A distributed flow-based DDoS detection solution for IoT using sequence majority voting. IEEE Access, 10: 88263-88275. https://doi.org/10.1109/ACCESS.2022.3200477
[21] Almorabea, O.M., Khanzada, T.J.S., Aslam, M.A., Hendi, F.A., Almorabea, A.M. (2023). IoT network-based intrusion detection framework: A solution to process ping floods originating from embedded devices. IEEE Access, 11: 119118-119145. https://doi.org/10.1109/ACCESS.2023.3327061
[22] Al-Sharafi, A.M., Alrayes, F.S., Alruwais, N., et al. (2025). Ensuring zero trust security in consumer internet of things using federated learning-based attack detection model. IEEE Access, 13: 54423-54438. https://doi.org/10.1109/ACCESS.2025.3551212
[23] Venturi, A., Apruzzese, G., Andreolini, M., Colajanni, M., Marchetti, M. (2021). Drelab-deep reinforcement learning adversarial botnet: A benchmark dataset for adversarial attacks against botnet intrusion detection systems. Data in Brief, 34: 106631. https://doi.org/10.1016/j.dib.2020.106631
[24] Hossain, M.A., Islam, M.S. (2025). A novel feature selection-driven ensemble learning approach for accurate botnet attack detection. Alexandria Engineering Journal, 118: 261-277. https://doi.org/10.1016/j.aej.2025.01.042
[25] Lefoane, M., Ghafir, I., Kabir, S., Awan, I.U. (2025). Internet of things botnets: A survey on artificial intelligence based detection techniques. Journal of Network and Computer Applications, 236: 104110. https://doi.org/10.1016/j.jnca.2025.104110
[26] Alqattan, D., Ojha, V., Habib, F., Noor, A., Morgan, G., Ranjan, R. (2025). Modular neural network for edge-based detection of early-stage IoT botnet. High-Confidence Computing, 5(1): 100230. https://doi.org/10.1016/j.hcc.2024.100230
[27] Patetta, M., Secci, S., Taktak, S. (2024). Line rate botnet detection with SmartNIC-embedded feature extraction. Computer Networks, 254: 110809. https://doi.org/10.1016/j.comnet.2024.110809
[28] Danquah, L.K.G., Appiah, S.Y., Mantey, V.A., Danlard, I., Akowuah, E.K. (2025). Computationally efficient deep federated learning with optimized feature selection for IoT botnet attack detection. Intelligent Systems with Applications, 25: 200462. https://doi.org/10.1016/j.iswa.2024.200462
[29] Saheed, Y.K., Chukwuere, J.E. (2024). XaiensembleTL-IoV: A new explainable Artificial Intelligence ensemble transfer learning for zero-day botnet attack detection in the Internet of Vehicles. Results in Engineering, 24: 103171. https://doi.org/10.1016/j.rineng.2024.103171
[30] Wu, G., Wang, X., Lu, Q., Zhang, H. (2024). Bot-DM: A dual-modal botnet detection method based on the combination of implicit semantic expression and graphical expression. Expert Systems with Applications, 248: 123384. https://doi.org/10.1016/j.eswa.2024.123384
[31] Swathi, G.C., Kumar, G.K., Kumar, A.S. (2024). Ensemble classification to predict botnet and its impact on IoT networks. Measurement: Sensors, 33: 101130. https://doi.org/10.1016/j.measen.2024.101130
[32] Ayo, F.E., Awotunde, J.B., Folorunso, S.O., Adigun, M.O., Ajagbe, S.A. (2023). A genomic rule-based KNN model for fast flux botnet detection. Egyptian Informatics Journal, 24(2): 313-325. https://doi.org/10.1016/j.eij.2023.05.002
[33] Randhawa, R.H., Aslam, N., Alauthman, M., Khalid, M., Rafiq, H. (2024). Deep reinforcement learning based evasion generative adversarial network for botnet detection. Future Generation Computer Systems, 150: 294-302. https://doi.org/10.1016/j.future.2023.09.011
[34] Sousa, B., Dias, D., Antunes, N., et al. (2024). MONDEO-Tactics5G: Multistage botnet detection and tactics for 5G/6G networks. Computers & Security, 140: 103768. https://doi.org/10.1016/j.cose.2024.103768
[35] Suryotrisongko, H., Musashi, Y. (2022). Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection. Procedia Computer Science, 197: 223-229. https://doi.org/10.1016/j.procs.2021.12.135
[36] Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E. (2012). To++ward open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(7): 1757-1772. https://doi.org/10.1109/TPAMI.2012.256
[37] Mohammed, N.T., Alturfi, S.M., Al-chalab, F.K.A.J., Al-Azzawi, F.F. (2026). Botnet detection in internet of things environments using double deep q-networks. International Journal of Safety & Security Engineering, 16(5): 1051-1058. https://doi.org/10.18280/ijsse.160511
[38] Sudheera, K.L.K., Priyashan, L.L.G.M., Pavithra, O.A.S., et al. (2026). Federated learning and data mining-based botnet attack detection framework for internet of things. Sensors, 26(5): 1573. https://doi.org/10.3390/s26051573
[39] Faarax, C.C., Sharma, G. (2026). Machine learning-based IoT botnet detection: Techniques, challenges, and future research directions: A comprehensive review. International Journal of Research & Technology, 14(1): 367-382. https://ijrt.org/j/article/view/941.
[40] Alshaeaa, H.Y., Ghadhban, Z.M. (2024). Developing a hybrid feature selection method to detect botnet attacks in IoT devices. Kuwait Journal of Science, 51(3): 100222. https://doi.org/10.1016/j.kjs.2024.100222
[41] Joshi, C., Ranjan, R.K., Bharti, V. (2022). A fuzzy logic based feature engineering approach for Botnet detection using ANN. Journal of King Saud University Computer and Information Sciences, 34(9): 6872-6882. https://doi.org/10.1016/j.jksuci.2021.06.018
[42] Abrantes, R., Mestre, P., Cunha, A. (2022). Exploring dataset manipulation via machine learning for botnet traffic. Procedia Computer Science, 196: 133-141. https://doi.org/10.1016/j.procs.2021.11.082
[43] Bernardeschi, C., Mercaldo, F., Nardone, V., Santone, A. (2019). Exploiting model checking for mobile botnet detection. Procedia Computer Science, 159: 963-972. https://doi.org/10.1016/j.procs.2019.09.263
[44] Chen, Z., Simsek, M., Kantarci, B., Bagheri, M., Djukic, P. (2024). Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier. Computer Networks, 250: 110576. https://doi.org/10.1016/j.comnet.2024.110576
[45] Fares, H., Vibhute, A.D., Mouniane, Y., Bouijij, H. (2025). Intrusion detection in wireless sensor networks using machine learning. Procedia Computer Science, 252: 912-921. https://doi.org/10.1016/j.procs.2025.01.052
[46] Lifandali, O., Abghour, N., Chiba, Z. (2023). Feature

selection using a combination of ant colony optimization and random forest algorithms applied to isolation forest based intrusion detection system. Procedia Computer Science, 220: 796-805. https://doi.org/10.1016/j.procs.2023.03.106
[47] Kurniabudi, Stiawan, D., Darmawijoyo, et al. (2022). Improvement of attack detection performance on the internet of things with PSO-search and random forest. Journal of Computational Science, 64: 101833. https://doi.org/10.1016/j.jocs.2022.101833
[48] Kumar, P., Liu, J., Tayeen, A.S.M., et al. (2023). Flnet2023: Realistic network intrusion detection dataset for federated learning. In MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM), Boston, MA, USA, pp. 345-350. https://doi.org/10.1109/MILCOM58377.2023.10356272
[49] 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, pp. 108-116. https://doi.org/10.5220/0006639801080116
[50] Sharafaldin, I., Lashkari, A.H., Hakak, S., Ghorbani, A.A. (2019). Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, pp. 1-8. https://doi.org/10.1109/CCST.2019.8888419
[51] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12: 2825-2830.
[52] Guyon, I., Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3: 1157-1182.
[53] Breiman, L. (2001). Random forests. Machine Learning, 45: 5-32. https://doi.org/10.1023/A:1010933404324
[54] Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3): 273-297. https://doi.org/10.1007/BF00994018
[55] Platt, J.C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, pp. 61-74. https://doi.org/10.7551/mitpress/1113.003.0008
[56] Chicco, D., Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics, 21(1): 6. https://doi.org/10.1186/s12864-019-6413-7
[57] Lundberg, S.M., Lee, S.I. (2017). A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 4768-4777. https://dl.acm.org/doi/10.5555/3295222.3295230.
[58] Villani, C. (2009). Optimal Transport: Old and New. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-71050-9
[59] Dunn, O.J. (1961). Multiple comparisons among means. Journal of the American Statistical Association, 56(293): 52-64. https://doi.org/10.1080/01621459.1961.10482090
[60] Chen, T., Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794. https://doi.org/10.1145/2939672.2939785
[61] Ke, G., Meng, Q., Finley, T., et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 3149-3157. https://dl.acm.org/doi/10.5555/3294996.3295074.
[62] Liu, F.T., Ting, K.M., Zhou, Z.H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, pp. 413-422. https://doi.org/10.1109/ICDM.2008.17
[63] Adawadkar, A.M.K., Kulkarni, N. (2022). Cyber-security and reinforcement learning—A brief survey. Engineering Applications of Artificial Intelligence, 114: 105116. https://doi.org/10.1016/j.engappai.2022.105116
[64] Deshpande, S., Kulkarni, N. (2021). Recent trends in cloud computing and edge computing. https://scholarworks.sjsu.edu/libphilprac/5438/.

NOMENCLATURE

$A(W)$	Window alarm: Binary indicator that at least one of the three drift signals exceeds its threshold in window W.
$D(W)$	Drift score: Prediction-conditioned Wasserstein drift score of window W, between 0 and 1.
k	Number of selected features: Number of features retained by ANOVA selection (k = 20).
N_{nov}, N_{uns}	Flag counts: Number of NOVEL and UNSURE flows in a window.
$\mathrm{p}^{\mathrm{ens}}$	Ensemble probability: Average of the RF and SVM class-probability vectors.
R_c	Reference sample: Up to 3,000 training flows of class c, in the selected feature space.
$S(N)$	Poisson surprise: Negative base-10 logarithm of the probability of at least N flagged flows under the expected rate.
SI	Shortcut index: Share of a classification task that the best single feature solves (0 to 1).
W, W_c	Window: Window of 100 consecutive flows and the flows of it predicted as class c.
z, $\mathcal{S}$	Flow vector and selection: Standardized flow feature vector and index set of the selected features.
$\hat{y}$	Predicted class: Class with the highest ensemble probability.

Greek symbols

α	Target false-alarm rate: Window false-alarm rate on known traffic (1%), divided by three signals.
κ	Confidence: Highest ensemble class probability of a flow.
$\nu(\mathrm{z})$	Novelty score: Mean distance of a flow to its five nearest reference flows of the predicted class.
π_c	Class flag rate: Share of flows of class c that are flagged on calibration data.
τ_D, τ_N, τ_U	Alarm thresholds: Calibrated thresholds for the drift score and the two Poisson surprises.
θ_c^{N}, θ^{U}	Flag thresholds: Per-class novelty threshold and confidence threshold for single flows.