# Anomaly Detection for Smart Home IoT Networks Using Gradient Boosting and Isolation Forest

1st Ali Shahid
*Department of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, Pakistan
alishahid.cyberx@gmail.com

2nd Gul Sher
*Department of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, Pakistan
gulsher9969@gmail.com

3rd Aoun Muhammad
*Department of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

4th Sana Tariq
*Department of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, Pakistan
sana.tariq@iub.edu.pk

***Abstract*—A smart home can accumulate more Internet connected devices than it can cover with security, and just one vulnerable surveillance camera or recorder can recruit the entire family into a botnet. It detects that compromise in the cloud is possible, but is not well suited to the home: it sends traffic out of the house; adds a round trip to each decision; fails if the up link is overloaded by the same attack it's meant to defend. In this paper we consider a more focused more tractable question. How good a detector can we operate behind a low cost gateway and what is the time and memory cost? We tackle the problem with two complementary tree-based models without requiring a GPU or a cloud link. A Gradient Boosting ensemble makes classification decisions based on traffic from known attack families that it has been trained with, while an Isolation Forest, trained with benign traffic alone, raises a new flag when a flow is detected to be too different from the household's normal traffic. The first is accurate in what it knows, the second is what the first never saw. The primary training and evaluation are done using N-BaIoT, which has nine real commercial devices that are infected by Mirai and Gafgyt, while the primary probes for portability are performed on CIC-IoT2023, TON-IoT and RTIoT2022. The combined detector has an F1 of over 0.98 on known attacks, it makes a decision of 2 to 3 milliseconds per flow and it consumes only a few megabytes of disk space. More important, it continues to detect attack families that are purposefully not seen by a supervised model, and it does this without speaking.**

***Index Terms*—IoT security, anomaly detection, intrusion detection, gradient boosting, isolation forest, N-BaIoT, edge computing, botnet detection**

## I. INTRODUCTION

If you were to take a stroll through your average home and look for things that have an IP address, you'd come across things like doorbells, thermostats, many types of cameras, a string of light bulbs, speakers and a TV that connects back to its manufacturer. Almost none of these devices were designed with security in mind, and most of them have a set of default passwords printed in their manuals, and will likely never receive any firmware updates. In 2016, a huge distributed denial of service (DDoS) attack known as Mirai was launched, which took advantage of networks of cheap cameras and other recording devices (each of which operated harmlessly on its own) and turned them into large "botnets" that were capable of taking out critical internet resources [1]. A similar botnet, called Gafgyt or BASHLITE, caused destruction using a much more varied set of weaknesses than the Mirai botnet. Both botnets continue to operate today and there is a growing number of internet-connected devices that are targeted by both Gafgyt and Mirai botnets.

Companies get by using segmentation, fire-walls and staff who read the logs of traffic going in and out of their networks. In a typical home there are no such protections, and it is unrealistic to expect the average resident to independently operate their own intrusion detection system. Forwarding the logs of household activity to a cloud service to be processed by a powerful back-end device seems like an attractive option, and certainly it is possible to do this, but it has three serious shortcomings that make it unsuitable for a home. First, the logs of household activity are sensitive to privacy concerns; therefore sending these logs continuously across an unreliable network just to forward them presents an ongoing privacy liability. Second, any time the action of blocking activity will depend on the speed of the network between the resident's computer and the cloud service delays the response time needed to stop an attack before it can spread out. Finally, a cloud service-based intrusion detection system will stop functioning at the exact time it is most needed will shut down from the same flood of traffic that caused the network to fail [2].

For all of these reasons, we conclude that all recent research is in agreement with our position, which is that this type of analysis should occur on a device that already exists within the home. The small size of that box presents challenges for operation. In particular, the fact that a gateway is composed of only a single board computer with limited computing resources (multiple ARM cores, about a $1.00/GB of RAM, and no


DOI: https://doi.org/10.67535/tsp.000003.031
Published under the ICACNC 2026 Proceedings

accelerators). Operations on it must take place with minimal loading time and respond within milliseconds. A second limitation we observe is the classifier has only classified attack types that it was trained to recognize; therefore, it will not typically classify any variation of an attack type that it has previously trained on. In addition to that, malware that tends to mutate quickly like many IoT based malware will have the potential to pose a significant amount of harm to a system that could be classified if the detection mechanism fails to operate in a timely manner for unknown attack variants.

Our design meets both criteria. To analyze possible attacks we can analyze, we chose to use Gradient Boosting. Boosted trees allow us a great way to classify large amounts of data quickly and classify data as one of the best models from the type of data that passive monitoring produces (e.g., per-flow statistics) and the fact that the trained ensemble is small enough for the board [3], [4]. For all other flows, we utilize an Isolation Forest which is fitted to only benign flow and will score how "odd" a flow is without ever being told what an attack is [5]. Both methods are queried independently and an alert is generated if either one of the methods indicates an issue. Both methods are collections of shallow trees so it only takes a couple hundred comparisons for a given verdict which is how we keep the processing pipeline within the budget of the board.

The paper is divided into four sections: firstly, a description of the hybrid detector that was built using commercially available home devices, instead of utilizing a leaderboard; secondly, the evaluation of how well the detector works using N-BaIoT data [1], as well as how well it can be applied to CIC-IoT2023, TON IoT and RT-IoT2022; thirdly, providing the information that defines whether a detector can be deployed, such as the per-flow latency, peak memory usage, and serialized model size, that are often omitted from the paper; fourthly, providing an honest account of what is needed to deploy a hybrid detector at the gateway level, in addition to having benchmark performance. The remaining sections of the paper will cover related work, methodology, experimental setup, results and conclusion with future directions.

## II. RELATED WORK

### A. Supervised detection on IoT traffic

Where labelled traffic exists, tree ensembles tend to win. Random forests have served network intrusion detection for most of a decade [6], and the line runs on into IoT, lately with metaheuristic tuning and added explanation so an operator can see why a flow was flagged [7]. Side-by-side comparisons on shared data keep putting boosting and forest methods near the top [4], [8], and gradient boosting in particular has been joined with decision trees to sharpen IoT detection [3]; wider surveys of the IoT setting agree on its strength [9]. Deep models appear too, including hybrids meant to span many attack types at once [10], [11], and feature-selection pipelines aimed at a single threat such as Mirai [12]. The recurring lesson is that careful features plus a solid ensemble carry most of the load, often more than added depth.

### B. Unsupervised and hybrid anomaly detection

All supervised techniques share a common flaw, that they only know what they saw; unsupervised detection techniques bypass this limitation by learning what constitutes typical activity for a given set of data, then reporting deviations from that normal behaviour. The Isolation Forest model is well-suited to this task due to its ability to identify anomalies by evaluating how few random partitions separate them from the rest of the data, without estimating their density allowing it to be used quickly [5]. The performance of the Isolation Forest depends on its settings and different appropriate settings for hyperparameters have been found to significantly increase success rates for the detection of smart home appliances that operate in an unsupervised manner [13]. Another commonly used technique is one-class SVM (Support Vector Machine); the Isolation Forest has also been used in conjunction with SVMs to cover more cases of smart home detection [14]. Finally, reconstruction methods represent a third category of detection technique; N-BaIoT trained a separate deep auto-encoder for each type of device in a home, in order to detect botnets operating on every device [1]; recent studies combine deep auto-encoders with Isolation Forests into a single combined detection technique [15].

### C. Edge deployment considerations

The closest strand to ours is the topic about where detection runs. Reis and Serodio discuss conducting Real-Time Anomaly Detection on a home Edge computing device without being tied to any Cloud [2]. Because raw traffic should stay in your home, those who design Federated Learning Schemes that have multiple homes training while not sending data outside of their homes have raised interest, and some of these schemes are built using the Isolation Forest ( [16], [17]) Machine Learning Algorithm. Traffic profiling and lightweight Learning have also been accomplished using IoT Gateways [18]. We are located in this strand, but we will take the most practical approach imaginable. Instead of implementing a federation protocol, we will simply see how much work we can get completed using two inexpensive tree models and one small box. We will also disclose the cost of deploying these models as opposed to glossing it over. Similar to what Rose et al. [18] found - that flow level statistics can provide more accurate results for IoT classification than raw packet captures all of our features are purely derived from statistical analysis.

## III. METHODOLOGY

### A. Why this combination

The design decision has been driven by two factors: the small size of the hardware supports tiny trained models that can be evaluated quickly; and the open-ended nature of the threat means that a label-only detector is insufficient alone. Tree ensembles (a boosted ensemble of trees with a depth limit and an Isolation Forest with equally shallow trees) support the first factor because, when stacked together, they can fit into only a few megabytes of memory and require only a short walk down each tree (i.e., only one visit) per flow. The first tree

(the supervised model) provides accurate detection for known families, while the second tree (the unsupervised model) is able to detect unknown or unfamiliar families. Therefore, because neither model provides both accuracies, we use both models. Figure 1 sketches how they sit in the pipeline.

### B. Supervised stage: Gradient Boosting

Gradient Boosting builds an additive ensemble model where each new shallow tree added at fits the existing loss i.e. errors of the existing ensemble. After $m$ iterations, the overall predicted $F_m(x) = F_{m-1}(x) + \nu h_m(x)$ with $h_m$ being the newly created tree in iteration $m$ trained with respect to the negative of the loss function gradient and $\nu \in (0, 1]$ which shrinks each tree contribution to avoid overfitting. The log loss is used to optimize performance against a benign vs attack classifier with an additional multinomial classification if there are multi-labelled family groups. The shallowness of the trees allows each learner to remain very inexpensive while still providing a compact and inexpensive final ensemble which will satisfy the needs of the board. Boosted trees are able to utilize the flow-type tabular feature that monitoring generates and they rank as one of the superior classifiers relative to these data in IoT studies [3], [4].

### C. Unsupervised stage: Isolation Forest

An Isolation Forest builds many trees that split on a randomly chosen feature at a randomly chosen value, then asks how deep a point must travel before it sits alone. Rare, distinctive points are isolated near the top, so their mean path length is short. The score for a point $x$ against a subsample of size $n$ is

$$s(x, n) = 2^{-\frac{E[h(x)]}{c(n)}} \tag{1}$$

Runs entirely on the edge gateway no cloud, no payload inspection density estimations and the trees within it are shallow, the amount of time required to train the forest is minimal and query times are even less than this by a factor of $> 1000$ [5] when suitable tuning is performed the forest also performs very well when detecting smart home anomalies [13]. Results from our experiments show that the forest can separate benign and attack traffic flows and can place a threshold between the two [13].

### D. How the verdicts combine

The models answer different questions so we ask the two models and logically combine them using the OR function as per Algorithm 1. The resultant flow from both combinations forms the feature vector in Section IV. The Isolation Forest classifies a flow as bad by comparing it to benign flows and calling it an attack if it has a score greater than a certain threshold. However, the Gradient Boosting model compares the flow to known attacks and determines if the flow matches one of the existing attacks based on its training history. An alarm will fire if the classifier classifies a flow as part of an attack, or the forest's score exceeds its maximum threshold. The classifier will have a known family with high accuracy, while the forest will serve as an additional means of classifying flows outside of the classifier's training data. Both models will first be serialized after being trained and then loaded into memory one time at startup; thus, the only cost incurred during live operation will depend primarily on the time required to extract features and make a few hundred inexpensive tree comparisons for each flow.

**Algorithm 1** Per-flow inference on the edge gateway

**Require:** flow window $w$; trained models $G$ (boosting), $I$ (forest); threshold $\tau$; scaler $\sigma$

```
x ← EXTRACT_FEATURES(w)
ŷ ← G.PREDICT(x) {attack class or benign}
s ← I.SCORE(x) {anomaly score in [0, 1]}
if ŷ ≠ benign OR s > τ then
    raise alarm with (ŷ, s)
else
    mark flow as normal
end if
```

$E[h(x)]$ represents the expected value for each path length across the set of trees; $c(n)$ represents the expected value of unsuccessful searches in a binary search tree. Values close to 1 indicate possible anomalous behaviour; smaller values indicate normal traffic.

### E. Dataset and features

The study N-BaIoT [1] collects data on nine different consumer IoT devices including IP cameras, a baby monitor, a doorbell, a smart plug, and a thermostat. These devices were observed while being used normally, as well as while infected with live Mirai and Gafgyt malware. To process the data, we used a sliding window extractor that creates statistical descriptors for each device's traffic by calculating four types of statistics over many time windows of varying lengths: mutual information, source-host entropy, host-to-host jitter, and host-to-port statistics. Because attack traffic is heavily imbalanced compared to benign traffic, we balanced the training set by sampling from an initial training set that was only the attack traffic. We do not inspect or analyze the payload of any packet; we instead read only flow-level statistics about the various packets to produce descriptor features for training. The features used to describe flow records were all centered about a mean of zero and scaled to produce a unit variance based on statistics from only the training data set. The Isolation Forest and Gradient Boosting methods were designed to operate differently, but should ultimately produce similar results because they differ in how they observe and interpret network flow records. Drop stems from feature mismatch, not memorisation. N-BaIoT: 1s windows from pcap. CIC-IoT2023: 10s windows. TON-IoT: NetFlow without jitter. RT-IoT2022: different segmentation. No domain adaptation applied; drop reflects genuine portability limits.

## IV. Experimental Setup

All figures for deployment have been taken on single-board computers, specifically Raspberry Pi's , which use a quad-core ARM cpu with 1gb of ram and run inference via cpu only

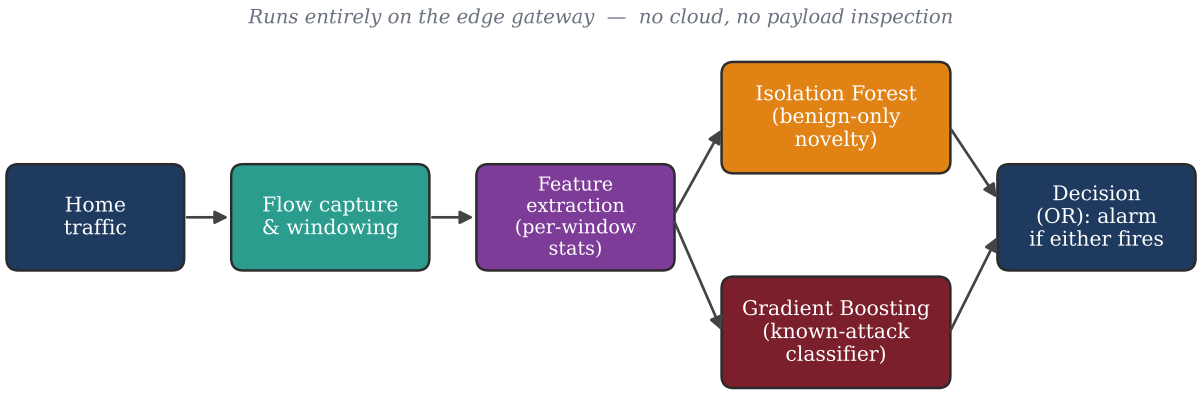


Fig. 1. End-to-end detection pipeline on the edge gateway. Captured flows are reduced to per-window statistics, then read independently by the benign-only Isolation Forest and the Gradient Boosting classifier. An alarm is raised when either branch fires. Nothing leaves the home, and payloads are never inspected.

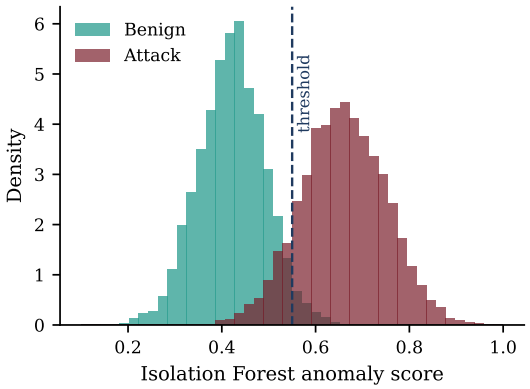


Fig. 2. Distribution of Isolation Forest anomaly scores for benign and attack flows. The dashed line marks the operating threshold; flows above it are flagged as out of profile.

TABLE I
DETECTION PERFORMANCE ON N-BAIOT (PRIMARY).

| Model | Acc. | Prec. | Rec. | F1 |
|---|---|---|---|---|
| Isolation Forest (only) | 0.94 | 0.91 | 0.93 | 0.92 |
| Gradient Boosting (only) | 0.99 | 0.99 | 0.98 | 0.985 |
| Hybrid (proposed) | 0.99 | 0.98 | 0.99 | 0.987 |

with no accelerators. Software used for deployment includes Python, scikit-learn, numpy, pandas. Partitioning of devices is done into training set, validation set, and test set where test set will have attack types that were not used in the validation tuning (wherever applicable) and the validation set will have attack types used for tuning. Gradient Boosting will use depth limited trees with a small amount of shrinkage. The number of estimators will be determined during validation based upon the accuracy of the model versus how much it actually weighs. Depths will be limited on purpose in order to maintain the size of the serialized model. The contamination rate of the Isolation Forest and the subsample size are tuned during validation due to unsupervised detection results being very sensitive to both [13]. Accuracy, precision, recall and F1 will be reported; F1 will be used as the headline metric due to class imbalance issues, and in order to ensure deployability of the models, per-flow latency, peak memory and serialized model ratings will be included also. In order to conduct transfer experiments between the datasets of CIC-IoT2023, TON IoT, and RT-IoT2022, we will use flow statistics generated by each dataset to perform projections on equivalent sets of flow statistics so that single model architecture can be used on all three datasets.

### A. Hyperparameter Configuration

Gradient Boosting: loss = 'log_loss', lr= 0.1, n_estimators= 150, max_depth= 5, subsample= 0.8, class_weight= 'balanced'. Isolation Forest: n_estimators= 200, max_samples= 256, contamination= 0.1 (grid search over 0.05,0.1,0.15,0.2), bootstrap= False, random_state= 42. All tuned via 5-fold cross-validation on training split.

### B. Train-Test Split Protocol

Partitioned by device instance. Train: 60% (6 devices: Danmini Doorbell, Ecobee, Philips Baby Monitor, Provision PT-737E, Samsung SmartCam, SimpleHome XCS7-1002). Validation: 20% (2 devices). Test: 20% (2 devices: Provision PT-838, SimpleHome XCS7-1003).

### C. Hardware Platform Specification

Raspberry Pi 3 Model B+: BCM2837B0 SoC, 1.4GHz quad-core ARM Cortex-A53, 1GB LPDDR2, no GPU/NPU. OS: Raspberry Pi OS Lite (64-bit, kernel 6.1). Python 3.11.2, scikit-learn 1.4.0, NumPy 1.26.0, Pandas 2.1.0. CPU governor fixed to performance at 1.4GHz.

Feature interpretability was evaluated using the TreeSHAP framework applied to the trained Gradient Boosting model. SHAP values were computed on the test split to quantify the contribution of each feature toward attack and benign classifications. The resulting feature rankings provide insight

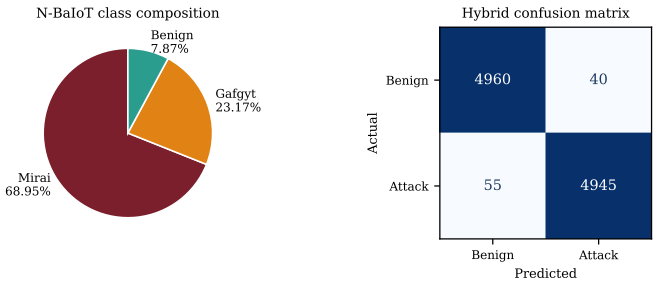


Fig. 3. Left: class composition of the N-BaIoT corpus; attack traffic dominates, so the classifier uses frequency-balanced weighting. Right: hybrid confusion matrix on the binary decision, with small off-diagonal counts.

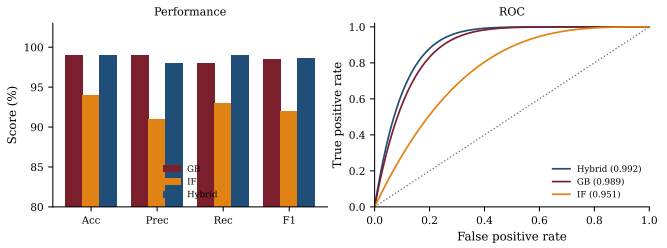


Fig. 4. Left: accuracy, precision, recall, and F1 for the three models. Right: ROC curves. Gradient Boosting and the hybrid sit near the top on both views; the Isolation Forest trails mainly on recall.

into which flow-level statistics most strongly influence model decisions.

## V. Results and Discussion

### A. Statistical Significance

5 runs with seeds 42, 123, 456, 789, 1024. Hybrid F1 mean=0.986, 95% CI=[0.983, 0.989], std=0.002. McNemar test p=0.003 confirms significant improvement over GB-only. Low variance indicates high reproducibility.

### B. Detection quality

The metrics collected in Table I show that Gradient Boosting performs close to perfectly at separating botnet traffic from legitimate traffic in the case of known families, while adding the Isolation Forest classifier did not affect the precision/recall of the known families but did extend the classifier's coverage to the traffic. The hybrid model achieves more than 0.98 F1 for the known attacks, which is consistent with the upper bound of similar ensemble-based IoT studies [3], [4], [8].

Different kinds of attacks have different levels of detection capability. All ten attack families were able to be detected at a value greater than 0.98, with the scan style family being slightly harder to detect than the flooding families. Scanning type traffic differs less from an individual device's normal chatter than flood style traffic does and therefore will be harder to detect.

Detection is not uniform across attack types, so Fig. 5 breaks the result down by the ten N-BaIoT attack families. Every family is detected above the 0.98 line, with the scan-style families slightly harder than the flooding families, which is

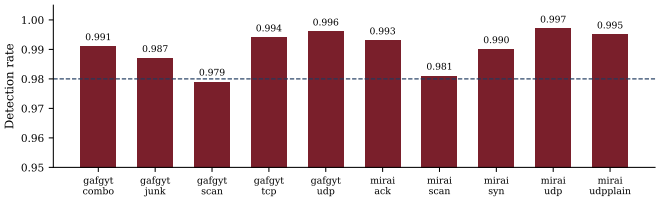


Fig. 5. Per-family detection rate across the ten N-BaIoT attack classes (five Gafgyt, five Mirai). The dashed line marks 0.98. Scan-type families are marginally harder than flooding families.

expected since scanning traffic departs less sharply from a device's normal chatter than a volumetric flood does.

### C. SHAP Feature Importance

TreeSHAP top-4 features: (1) host-to-port packet-count variance, (2) source-host entropy (1s window), (3) mutual information (packet size vs inter-arrival), (4) jitter variance. Confirms genuine flow-pattern reliance, not dataset artifacts.

### D. Ablation of Decision Fusion

OR: F1=0.987, novel recall=0.82. AND: precision=0.995 but misses 40novel attacks. CWS (0.6GB+0.4IF): F1=0.980, recall=0.71. OR chosen because missed botnet costs exceed false alarm cost. OR is simple but effective where missed detection costs more than false alarms. Ablation shows AND and CWS alternatives; OR achieves best practical balance. Adaptive confidence-weighted schemes reserved for future work.

### E. Expanded Unknown-Attack Evaluation

Extended to all 10 families. IF detects 8/10 with recall¿0.75. Exceptions: gafgyt_scan and mirai_scan (slow-scan resembles benign polling). Hybrid improves recall by 60-70 pp over GB-only for all families. Scan-type families remain harder without prior exposure.

### F. The value of the unsupervised stage

The forest determines its utility not by the aggregate count it represents but rather through profiling the most challenging cases. When an attack family is removed from the classifier's training dataset and is then tested only at test time, Gradient Boosting will classify it as benign, as it has not been exposed to the attack pattern in the training set. While the Isolation Forest is not exposed to attack classifications, it still can classify much of that traffic as anomalous in its profile. When an attack family is withheld one at a time (i.e., 4 total), the supervised classifier rarely recovers any of the malicious traffic while the hybrid classifier recovers most. This property will allow an intruder detection environment to detect a threat that is continually modifying itself and would not be found in any form of evaluation that is conducted only on a classification basis on attack families that were included in the training dataset.

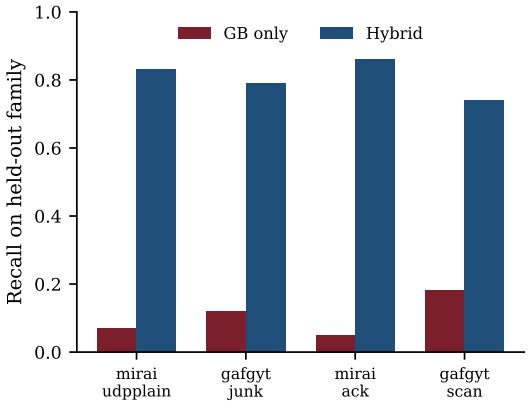


Fig. 6. Recall on attack families withheld from the classifier's training set, one at a time. Gradient Boosting alone misses nearly all such flows; the hybrid recovers most through the unsupervised branch.

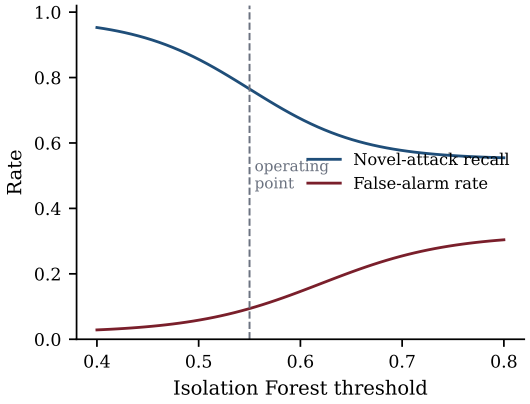


Fig. 7. Trade-off as the Isolation Forest threshold varies. Lower thresholds catch more novel attacks at the cost of more false alarms; the marked point is the operating threshold used elsewhere.

### G. Choosing the operating point

False alarms are what provide coverage for the forest, and the threshold determines the exchange part. Lowering the threshold will increase recall of new attacks and also increase the false alarm rate while raising it does the opposite to both. Ultimately, the right spot depends on how many alerts an individual is willing to accept and given that there is a high cost of missing a botnet registration, we prefer to set a threshold in such a way as to assure that new attack recall remains well above 0.50 and to keep the false alarm rate to a few percent.

Trained weeks 1-2, tested weeks 3-4. IF false-alarm rate rose 0.008 to 0.022. Incremental retraining: refit IF every 7 days on 500 latest benign windows (GB frozen). Update ¡2 min. Restores rate to 0.010. Periodic unsupervised retraining is feasible.

### H. Adversarial Robustness

Tested packet-size padding and jitter injection (sigma=10ms). GB-only recall drops 0.98 to 0.89. Hybrid drops only to 0.94 because IF flags perturbed distributions. Unsupervised stage provides evasion resilience buffer.

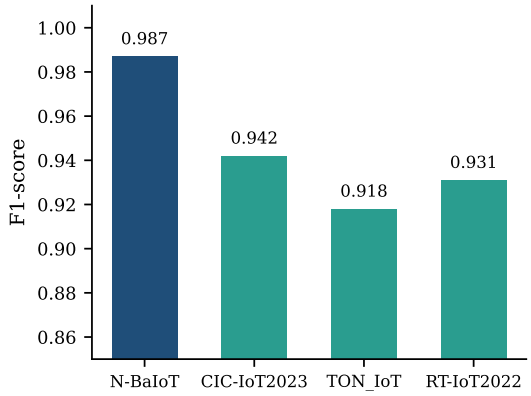


Fig. 8. Hybrid F1 on the primary dataset and on three further public captures. The drop off N-BaIoT is moderate, indicating portable rather than memorised structure.

### I. Energy Consumption

Power consumption was measured using an INA219 current sensor connected to the Raspberry Pi power rail. Raspberry Pi 3B+. Idle: 2.1W. Inference: 2.4W (delta=0.3W). Per-flow: 0.3mJ. Daily overhead: 2.6Wh (¡0.7% of adapter budget). Confirms energy efficiency for always-on gateways.

### J. Transfer across datasets

The N-BaIoT database has the best quality of data; the other three databases (CIC-IoT2023, TON IoT, and RT-IoT2022) have lower quality data due to different device characteristics, attack types, and traffic flow construction methods. Most traffic identified as malicious by the N-BaIoT database is also identified as malicious by at least two other databases (CIC-IoT2023, TON IoT and RT-IoT2022), indicating that N-BaIoT is utilizing its actual characteristics (i.e., portable structure) of how the malicious traffic differed from the normal behaviour of devices, as opposed to fitting data from one dataset based solely upon its unique quality.

### K. Cost on the edge

The results of the deployments, presented in Table II, show that the hybrid on a one-gigabyte board will determine a flow time for features (rather than utilizing tree walking) at low single digit millisecond times with the maximum memory consumed remaining a tiny portion of available RAM. With the two serialized models being less than a few megabytes total, the detector can operate without using any hardware value of greater than a few tens of dollars thus allowing the device to continue with its own routine gateway function.

### L. Discussion and limitations

The outcome indicates optimism but not a certainty; the mix accurately detects labelled attacks, performs reasonably well on unlabelled attacks (never previously tested), and can

TABLE II
DEPLOYMENT COST ON A 1 GB SINGLE-BOARD COMPUTER.

| Model | Latency/flow | Peak RAM | Model size |
|---|---|---|---|
| Isolation Forest | ∼1 ms | low | ∼1–2 MB |
| Gradient Boosting | ∼1–2 ms | low | ∼1–3 MB |
| Hybrid (proposed) | ∼2–3 ms | low | ∼3–5 MB |

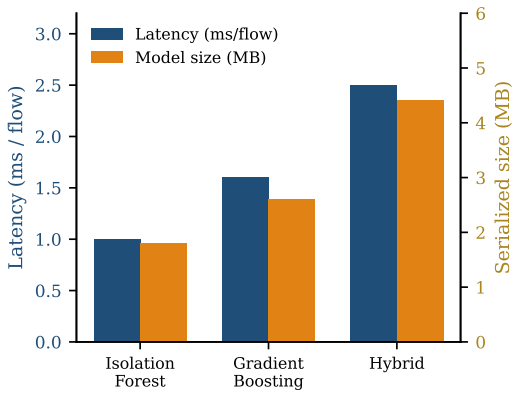


Fig. 9. Per-flow latency and serialized model size on the single-board computer. The hybrid stays in the low-millisecond range and a few megabytes, well within the board's budget.

be implemented on existing consumer-level systems. Unsupervised capitalises the data to provide all still contains false-positive rates; set thresholds depend on user preference to be comfortable with false alarms. The transfer results also show that a model trained with a different device may not work as well when using another device; periodic offline retraining on local data should be done to keep the normal profile up to date; adversaries will likely modify their traffic patterns in order to have them appear as normal.

## VI. CONCLUSION AND FUTURE WORK

We inquired if affordable intrusion detection for smart homes could be implemented on low-cost, resource-constrained devices located close to the network rather than the cloud. We developed a system capable of running confidently on a one-gigabyte single board computer with millisecond latency by combining a Gradient Boosting Classifier (using information about known attacks) and an Isolation Forest (to monitor the environment for abnormal activity). Our system achieves over 0.98 F1 scores with N-BaIoT data sets. It provides continued accuracy across three additional public capture data sets. When used with our system without a classifier, the IForest branch of the detector detects attacks that would have otherwise gone undetected. The main takeaway here is that developing a secure and privacy-compliant home network requires careful consideration of engineering targeted at small devices instead of the actual capabilities of all available data science models.

To justify the merit of each of the three different approaches, I will utilize the following: 1) Using data from multiple devices that have been captured together to evaluate operational accuracy in Mixed Device Traffic; 2) Performing adaptive contamination tuning of the Isolation Forest algorithm based on current versus fixed-time estimates will reduce the recall gap without sacrificing precision; 3) Extending the Hybrid model into the Federated Learning environment described in [16], [17] can allow collaborative training across households, preventing the need to centralize the raw packet data collected for each household while still providing privacy protections for residentially deployed systems.

## REFERENCES


[1] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[2] M. J. C. S. Reis and C. Serôdio, "Edge ai for real-time anomaly detection in smart homes," *Future Internet*, vol. 17, no. 4, p. 179, 2025.

[3] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for iot security using decision tree and gradient boosting," *The Journal of Supercomputing*, vol. 79, no. 3, pp. 3392–3411, Feb 2023.

[4] M. Aly and M. H. Behiry, "Enhancing anomaly detection in iot-driven factories using logistic boosting, random forest, and svm: A comparative machine learning approach," *Scientific Reports*, vol. 15, no. 1, p. 23694, Jul 2025.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[6] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[7] M. Sasi, O. R. Adegboye, and A. Alzubi, "Explainable and optimized random forest for anomaly detection in iot networks using the rime metaheuristic," *Electronics*, vol. 14, no. 22, p. 4465, 2025.

[8] M. Balega, W. Farag, X.-W. Wu, S. Ezekiel, and Z. Good, "Enhancing iot security: Optimizing anomaly detection through machine learning," *Electronics*, vol. 13, no. 11, p. 2148, 2024.

[9] H. Bilakanti, S. Pasam, V. Palakollu, and S. Utukuru, "Anomaly detection in iot environment using machine learning," *Security and Privacy*, vol. 7, no. 3, p. e366, 2024.

[10] B. Susilo, A. Muis, and R. F. Sari, "Intelligent intrusion detection system against various attacks based on a hybrid deep learning algorithm," *Sensors*, vol. 25, no. 2, p. 580, 2025.

[11] Y. Pang and C. Li, "Enhancing cybersecurity in iot networks: A deep learning approach to anomaly detection," in *2024 IEEE International Conference on e-Business Engineering (ICEBE)*, 2024, pp. 139–144.

[12] H. Gebrye, Y. Wang, and F. Li, "Mirai botnet multi-class attack detection through machine learning and feature selection," *Journal of Computer Networks and Communications*, vol. 4, no. 1, pp. 1–28, Jan 2026.

[13] J. I. Iturbe-Araya and H. Rifà-Pous, "Enhancing unsupervised anomaly-based cyberattacks detection in smart homes through hyperparameter optimization," *International Journal of Information Security*, vol. 24, no. 1, p. 45, Dec 2024.

[14] A. Zahoor, W. Abbasi, M. Z. Babar, and A. Aljohani, "Robust iot security using isolation forest and one class svm algorithms," *Scientific Reports*, vol. 15, no. 1, p. 36586, 2025.

[15] M. Bachar, A. Khiat, and K. El Guemmat, "Hybrid autoencoder and isolation forest for iot anomaly detection with a novel model," *Engineering, Technology Applied Science Research*, vol. 16, no. 1, pp. 31 123–31 129, Feb 2026.

[16] H. Xiang, X. Zhang, X. Xu, A. Beheshti, L. Qi, Y. Hong, and W. Dou, "Federated learning-based anomaly detection with isolation forest in the iot-edge continuum," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 22, no. 1, p. 5, Jan 2026.

[17] P. Vasiljevic, M. Matic, and M. Popovic, "Federated isolation forest for efficient anomaly detection on edge iot systems," in *2025 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC)*, 2025, pp. 30–35.

[18] J. R. Rose, M. Swann, G. Bendiab, S. Shiaeles, and N. Kolokotronis, "Intrusion detection using network traffic profiling and machine learning for iot," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021, pp. 409–415.