Design and Evaluation of a Three-Tier Hybrid Cloud-Edge-AI Intrusion Detection and Response Framework for Mobile and IP Networks

Somendra Pratap Singh[1], Rajesh Kumar Singh[1,] Raghav Mehra[2]

[1]Department of Computer Science, Bhagwant University, Ajmer, Rajasthan,

[2] AIT-CSE, Chandigarh University, Mohali, Punjab

DOI: https://doi.org/10.51244/IJRSI.2026.1310000086

Received: 13 October 2026; Accepted: 19 October 2026; Published: 02 November 2026

ABSTRACT

Networks carrying mobile traffic face three demands that pull against each other. Finding new attacks needs statistical models rich enough to describe normal behaviour, yet such models run poorly on battery powered devices. Cutting false alarms needs longer observation, yet fast containment needs a quick decision. Learning from traffic at many sites needs that traffic pooled, yet privacy rules forbid pooling. Most published work meets one demand at a time. This paper presents HIDRF, which spreads detection and response across three tiers so each tier can take a different position on the same trade off. Device agents send only flow metadata, never packet contents. At the edge, a C4.5 signature module and a Naive Bayes anomaly module run side by side, a Meta-AdaBoostM1 combiner merges their outputs, and its confidence score decides whether a flow triggers an immediate response, goes to the cloud, or is only logged.

The same score selects the response action, so detection and response are one mechanism. Federated learning, simulated with ten edge nodes on one machine, spreads what each node learns into a shared model while data stays where it was collected. On UNSW-NB15 the hybrid edge engine reached 90.25 per cent accuracy at a false alarm rate of 1.1 per cent, below both of its own modules at 8.4 and 2.1 per cent, and the full framework reached 95.0 per cent. Edge processing used 34.6 per cent of a Raspberry Pi 3 Model B+ and finished in under 0.9 seconds from flow record arrival to response. Under deliberate evasion, edge detection fell to 57.3 per cent and cloud escalation recovered it to 81.9 per cent. The contribution is the architecture and its measured breakdown, not any single mechanism. All results come from one 2015 benchmark and are an upper bound rather than expected field performance.

Keywords: intrusion detection, intrusion response, edge computing, federated learning, hybrid detection, mobile IP, UNSW-NB15

INTRODUCTION

Background and the Security Problem

Mobile connectivity is now the normal condition under which network traffic is produced. Estimates place the number of connected devices worldwide above fifteen billion, with projections beyond twenty-nine billion by 2030 (Statista, 2024). Mobile IP lets a device keep one permanent address while it moves between networks (Perkins, 2010), and mobility has a cost in exposure.

A wireless network shares an open medium, so an attacker needs no physical access to cables or switches, and registration forces a mobile node to declare its identity to equipment it has never dealt with before. Denial of service, session hijacking, replay and address spoofing all follow, and all four are still common in deployed networks (Bhatti and Crowcroft, 2013).

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

Limits of Existing Methods

Signature based systems match traffic against a stored list of attack patterns, so they cannot find an attack absent from the list. A gap stays open between the first use of a new attack and the release of a matching rule (Hindy et al., 2020). Anomaly based systems close that gap by modelling normal behaviour, and pay for it in false alarms. Chawla et al. (2023) report a system with a 100 per cent detection rate and a 60 per cent false alarm rate; a detector that flags six of every ten legitimate flows cannot be used, whatever its detection rate.

Mobile devices also lack the processing power and battery budget accurate anomaly detection needs. Edge computing sits near the data and cuts delay; cloud computing gives capacity, but only after a round trip. Detection needs both speed and depth, and the literature has chased them separately. Weiss et al. (2021) conclude that designs combining edge and cloud are the most promising route, and note that such designs have been described far more often than built and measured.

Research Gap

Hybrid detection, machine learning for intrusion detection, edge and cloud architecture, and federated learning have each been studied in depth. They have not been joined into one framework that is built and measured as a whole under real resource limits, with response attached to detection rather than left to a separate system. The parts pull against each other, so the difficulty lies in the joining.

Objective and Scope

This paper addresses one objective from a wider doctoral study: to design a three-tier Hybrid Cloud-Edge-AI Intrusion Detection and Response Framework that answers the limits identified in the earlier objectives of that study, and to measure it.

The scope should be stated plainly. The wider study also analyses Mobile IP and IPv6 mechanisms, IPSec, and security in mobile ad hoc networks. This paper draws on that analysis for the threat picture of Section 1.1, but it does not evaluate those protocols and no claim here rests on them. Every experiment below uses the UNSW-NB15 benchmark, and Section 6 states what that limits.

Contributions

C1. A three-tier device, edge and cloud architecture joined by a single confidence threshold rule that routes each flow to immediate response, to deeper cloud study, or to the log.

C2. A response mechanism driven by the same confidence score as the routing, so that a detection and the action taken on it come from one decision rather than two.

C3. A feature selection method that builds a separate feature subset for each detection module, because the two modules need different properties of the data.

C4. A controlled ablation study measuring what each tier adds, together with an adversarial test measuring what the cloud tier recovers when the edge tier is evaded.

No single mechanism used here is new. Decision trees, Naive Bayes, boosting, sandbox analysis and federated averaging all have long literatures. The claim concerns the composition and its measurement.

Related Work

Signature based detection. Signature matching is still most of what is deployed, largely through Snort and its successors (Roesch, 1999).

Anomaly based and machine learning detection. Anomaly detection models normal behaviour and flags departures from it, a pattern set by Denning (1987). New attacks become findable in principle, but behaviour unusual enough to flag also includes rare legitimate traffic. Ou and Ou (2012) showed early that joining the two paradigms beats either alone, and the reason matters here: they fail on different inputs. Among recent work, Park et al. (2024) report about six per cent improvement over a standard baseline using a Siamese convolutional network, Ola et al. (2025) applied a deep convolutional network to an extended version of UNSW-NB15, Jouhari et al. (2024) joined chi-square feature selection with a CNN-BiLSTM model at lower cost, and More et al. (2024) found ensembles beating single models. Abdulhammed et al. (2019) showed that feature selection rather than classifier choice often decides performance, and that smaller well chosen subsets beat larger ones; that shaped the method of Section 4.2.

Edge, cloud and federated detection. Wang et al. (2020) argue that privacy protection is not optional where edge nodes collect traffic from many users. Weiss et al. (2021) identify combined edge and cloud detection as the most promising direction but do not build one. In the method of McMahan et al. (2017) each node trains on its own data and sends only model updates, so raw data never leaves the node. This suits intrusion detection, because traffic is exactly the data organisations are least able to share, but that work does not test the method in a security setting or inside a layered detector.

Table 1 sets out what each strand supplies and what it leaves open.

Table 1: What existing work supplies and what it leaves open

Strand	Representative work	What it supplies	What it leaves open
Signature detection	Roesch (1999)	Low cost matching of known attacks	Cannot find attacks absent from the rule set
Anomaly detection	Denning (1987)	Can flag previously unseen behaviour	False alarm volume; no cost control
Hybrid detection	Ou and Ou (2012)	Two paradigms beat either alone	Host based only; no tier structure
Machine learning IDS	Park et al. (2024); Jouhari et al. (2024); More et al. (2024)	Strong accuracy on benchmarks	Single stage and central; no resource limit tested
Feature selection	Abdulhammed et al. (2019)	Smaller subsets can beat larger ones	One subset serving two different detectors
Edge and cloud IDS	Wang et al. (2020); Weiss et al. (2021)	Identifies the tiered direction	Proposed rather than built and measured
Federated learning	McMahan et al. (2017)	Learning without pooling data	Not tested for security or inside a layered detector

The gap this paper addresses is the composition, and the composition is hard because the parts conflict.

THE PROPOSED HIDRF FRAMEWORK

Design Goals

Six goals shape the design: find both known and new attacks; respond quickly; stay inside the processing, memory and battery limits of mobile devices; work at modern traffic volumes; keep improving as new attacks appear; and protect privacy while still allowing shared model improvement.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

These goals pull against each other, and one mechanism cannot settle the conflict, because it must take a single position on each trade off and apply it everywhere. A layered design has no such limit. The device tier accepts shallow analysis at very low cost and applies it to everything, the cloud tier accepts high cost for depth and applies it to very little, and the edge tier sits in the middle and carries most of the decisions. What makes this work is the routing between tiers rather than the tiers themselves.

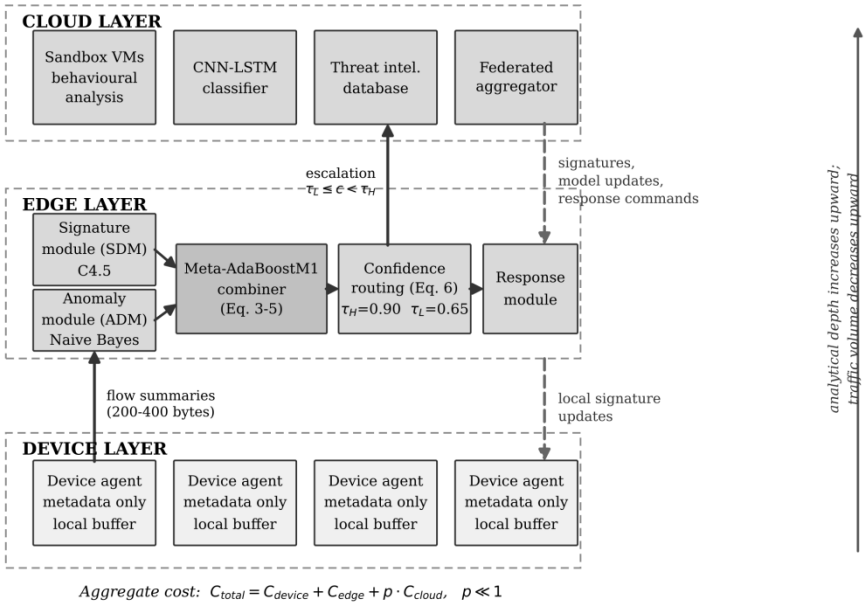


Figure 1: The three-tier HIDRF architecture. Solid arrows carry traffic information upward; dashed arrows carry signatures, model updates and response commands downward.

Because only flows the edge cannot settle reach the cloud, total cost is driven by the cheapest stage rather than the costliest. The downward path matters as much: when the cloud identifies a new attack pattern it sends signatures to every edge node, so a threat found once anywhere becomes findable everywhere at the cheapest tier able to recognise it.

Device Layer

Each agent records connection metadata to a local buffer: source and destination address and port, transport protocol, connection start time, and the sizes and flag values of the first packets. After thirty seconds, or at once if a connection matches the small local signature list with high confidence, the buffer is compressed into a flow summary of about two hundred to four hundred bytes and sent to the nearest edge node over an encrypted MQTT channel.

Raw packet contents never leave the device. This is a property of the architecture rather than a setting, so no edge or cloud component can read message content. The cost is that attacks which only content inspection would catch cannot be found, and Section 6 records this.

Hybrid Edge Detection

Two modules run side by side at the edge, as the middle band of Figure 1 shows.

The **Signature Detection Module (SDM)** is a C4.5 decision tree, that is, a tree of yes or no tests on feature values. It is trained on the fifteen signature features, and rules taken from the tree are held in an indexed database. At run time it walks the tree from root to leaf and returns the leaf label, with the class purity at that leaf as its confidence. The trained tree has an average depth of about twelve. A tree was chosen over a stronger classifier because a person can read its rules, so an alert can carry the condition that raised it.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

The **Anomaly Detection Module (ADM)** is a Naive Bayes classifier, that is, a statistical method treating each feature as independent evidence about the class. It is trained on labelled records using the fifteen anomaly features and predicts one of the ten classes. Equation 1 gives the rule it applies, where C_k is a class, m the number of features and K the number of classes, and Equation 2 the confidence, taken as the posterior of the chosen class normalised across all classes:

$$\ell_A = \arg\max_k P(C_k) \cdot \prod_{j=1}^{m} P(x_A^{(j)} \mid C_k) \quad (1)$$

$$c_A = P(C_{\ell_A} \mid x_A) / \Sigma_{k=1}^{K} P(C_k \mid x_A) \quad (2)$$

The module is therefore a supervised classifier over all ten classes, not a one class model built only from normal traffic. Table 4 reports its precision and recall in each attack category, which a one class model could not produce. It does keep a running buffer of records the framework logged as normal, and that buffer feeds the next local training round, which is how it keeps pace with changes in normal traffic at its own site. That is a separate matter from how the module is built.

The **Meta-AdaBoostM1 combiner** merges the two outputs. AdaBoost builds a strong classifier from many weak ones, weighting each by its training performance. Here it works on the meta feature vector $z = [\ell_S, c_S, \ell_A, c_A]$, holding the two labels and two confidence values. It never sees the original flow features, so its run time cost does not depend on the number of features. Writing T for the number of weak hypotheses, h_t for the t-th and ε_t for its training error (Freund and Schapire, 1997):

$$H(z) = \arg\max_k \Sigma_{t=1}^{T} \alpha_t \cdot \mathbb{1}[h_t(z) = k] \quad (3)$$

$$\alpha_t = \ln((1 - \varepsilon_t) / \varepsilon_t) \quad (4)$$

$$c = (\Sigma_{t : h_t(z) = H(z)} \alpha_t) / (\Sigma_{t=1}^{T} \alpha_t) \quad (5)$$

Confidence Based Routing

The routing rule uses a high threshold $\tau_H = 0.90$ and a low threshold $\tau_L = 0.65$. For a flow record F:

$$D(F) = \text{respond, if } H(z) \neq \text{Normal and } c \geq \tau_H; \text{ escalate, if } H(z) \neq \text{Normal and } \tau_L \leq c < \tau_H; \text{ log, otherwise} \quad (6)$$

The two thresholds cut the confidence range into three parts and so decide how much traffic reaches each tier. Raising τ_L lowers the load on the cloud but lets more uncertain traffic pass unchecked. They were chosen on a validation subset of the training data by weighing false alarm rate, escalation volume and detection rate against each other, then fixed before the test set was used and held fixed across every ablation configuration.

Equation 6 is also the cost control. Writing p for the share of traffic escalated, aggregate cost is $C_{total} = C_{device} + C_{edge} + p \cdot C_{cloud}$ with p much smaller than one, so an operator trades cloud spending against detection depth by moving the two thresholds. Section 5.7 shows the same thresholds govern robustness against an attacker, making them a security control as well.

Cloud Analysis

For medium confidence cases the edge node sends the compressed flow record together with the two module outputs to the cloud over HTTPS with mutual TLS. Packet contents are not sent, in keeping with Section 3.2.

A light pre-filter first checks the record against the full threat intelligence database, which is larger and more current than any single edge rule set, so records escalated only because an edge database was out of date are settled cheaply. Records that remain go to the sandbox, a set of isolated virtual machines running different operating systems and software. The sandbox does not receive packet contents. It builds a replay trace from

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

the flow record, runs that trace in isolation, and records what the trace does: which system calls it makes, which network connections it tries to open, how it uses the file system and the registry where one exists, and how it uses memory.

These behaviour features are the input to a CNN-LSTM model. A CNN, or convolutional neural network, picks out structure inside each observation; an LSTM, or long short term memory network, picks up how a sequence changes over time (Hochreiter and Schmidhuber, 1997). The pairing covers two properties of the data that neither part covers alone.

The difference from edge detection is one of kind, not degree: the edge judges traffic by its statistical properties, while the sandbox watches what the traffic does. Section 5.7 measures the effect. Where the cloud confirms an attack it sends a response command to the originating edge node and adds the confirmed pattern to the signature database and to the federated training data.

Federated Learning

Federated learning lets the edge models keep improving without traffic data being gathered in one place. The model that is federated is the edge anomaly detection model; the signature module is updated instead by the downward signature distribution of Section 3.5. **The experiments reported here simulate ten edge nodes on a single machine. No distributed deployment was carried out.**

Each round runs as follows. The cloud sends the current global model to every node. Each node trains that model on its own local traffic, computes the difference between its trained model and the model it received, encrypts that difference, and sends only the difference. Writing θ for the global model parameters, δ_i for the difference computed at node i, n_i for the number of local training records there and N for the number of nodes, the server merges the updates by weighted average and returns the new global model:

$$\theta' = \theta + \left(\Sigma_{i=1}^{N} w_i \cdot \delta_i \right) / \left(\Sigma_{i=1}^{N} w_i \right), w_i \propto n_i \quad (7)$$

In the deployment model rounds repeat about once a day, with an emergency update pushed at once for a critical new threat; the simulated rounds are not tied to any interval of clock time.

Before merging, each update is compared against the consensus of the others, and an update differing beyond a statistical bound is given less weight or excluded and logged for review. This is a design feature; no poisoning experiment was carried out, so no claim is made about how well it works. Three measures protect privacy: calibrated noise added to updates, following Abadi et al. (2016); secure aggregation, which lets the server compute the average without reading any single update (Bonawitz et al., 2019); and compression of updates. No privacy budget was set and no formal privacy guarantee is claimed.

Intrusion Response

Detection not joined to response tells an operator about a problem without reducing it. Here the response action is selected by the same confidence score that drives routing, at the edge node serving the device, with no round trip to a central authority. Five actions are available: immediate blocking, quarantine of the device into an isolated segment that keeps access to repair services, redirection to a honeypot, dynamic update of edge firewall rules, and notification of security staff. Table 2 maps cases to actions.

Table 2: Response action by detection case

Detection case	Response action	Response time	Automation
Known attack, signature match	Immediate block and alert	under 100 ms	Full
Anomaly, low confidence	Alert and escalate to cloud	under 500 ms	Partial

Anomaly, high confidence	Quarantine and alert	under 300 ms	Full
Cloud confirms new attack	Block, push signature, alert	under 5 s	Full
Distributed attack	Coordinated block across edge nodes	under 2 s	Full

The pattern in Table 2 is that full automation is kept for cases that are either high confidence or cheap to undo. The low confidence anomaly is the one partly automated case, because a wrong automatic action there would cost more than a short wait for human judgement.

EXPERIMENTAL METHODOLOGY

Dataset and Split

The evaluation uses UNSW-NB15 (Moustafa and Slay, 2015), which mixes real background traffic with synthetic attack traffic across nine attack categories and is widely enough used that results can be set beside published work. The standard split was used without change, and each record has 49 features. The split holds 175,341 training and 82,332 test records, 257,673 in all. The balance is very uneven. Normal is 36.09 per cent of records and Generic 22.85 per cent, while Worms is 0.07 per cent, that is 130 training and 44 test records. A classifier labelling every record Normal would score above 36 per cent accuracy while detecting nothing at all, which is why detection rate and false alarm rate are reported separately throughout. The dataset was produced in 2015, so testing on it gives comparability with the literature but does not show behaviour against attack methods developed since.

Preprocessing and Feature Selection

Categorical attributes, mainly protocol, service and state, were converted to numbers. Continuous features were scaled to a common range, which matters for the anomaly module because probabilistic methods are sensitive to differences in magnitude. Records with missing or malformed values were dropped. Feature selection was done twice, once for each module. Rule based classification does well on features that split cleanly at fixed thresholds; statistical anomaly detection does well on features whose distributions differ measurably between normal and attack traffic. One subset serving both must compromise. An information gain filter first removed features with little discriminating value. Recursive feature elimination against the C4.5 classifier then chose the signature subset, and mutual information scored against a Naive Bayes baseline chose the anomaly subset. Fifteen features were kept for each, so the framework uses thirty of the 49 available.

Hardware and Software

The two platforms in Table 3 were chosen to represent the resource conditions of different tiers rather than to maximise measured performance.

Table 3: Hardware and software configuration

Element	Configuration
Edge platform	Raspberry Pi 3 Model B+; 1.4 GHz quad core ARM Cortex-A53; 1 GB RAM
Cloud platform	Intel Core i5-8250U; 8 GB RAM; Ubuntu Linux 22.04
Deep learning framework	TensorFlow 2.12, GPU acceleration for training and sandbox analysis
Language runtime	Python 3.10

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

Device agent	C and Python, for portability across mobile and IoT platforms
Edge classifiers	C4.5 decision tree; Naive Bayes; Meta-AdaBoostM1 combiner
Transport	MQTT device to edge; HTTPS with mutual TLS edge to cloud
Network simulation	NS-3 discrete event simulator with custom detection and aggregation modules

The edge platform was chosen because it is weak. If the engine runs well here it will run on any realistic Multi-access Edge Computing node, so the figures in Section 5.6 are an upper bound.

Metrics

Four measures are reported and must be kept apart, because the same number means different things under each. **Accuracy** is overall accuracy across all ten classes, so a flow correctly identified as an attack but placed in the wrong category counts as an error. **Average Accuracy** is the mean of the per class accuracies, giving each class equal weight whatever its size. **Attack Detection Rate (ADR)** is TP / (TP + FN), computed on the two class split of attack against normal, and is the same quantity as recall.

False Alarm Rate (FAR) is FP / (FP + TN) on the same two class split, the share of normal traffic wrongly flagged. Here TP is an attack correctly flagged, TN normal traffic correctly passed, FP normal traffic wrongly flagged and FN an attack missed. Precision is TP / (TP + FP) and F1 is 2PR / (P + R). Because ADR and FAR are two class measures while accuracy is a ten class measure, neither can be derived from the other, and a figure stated in one cannot be restated in the other. Both are therefore reported in every results table. Reducing false alarms is treated as a first order goal rather than a secondary one, because a detector whose alerts are mostly wrong will stop being read.

Baselines and Protocol

Three baselines are used: a traditional signature based system, a purely machine learning system, and the edge only configuration of HIDRF without cloud escalation or federated learning.

The third is the most informative, because it isolates what the upper tiers add inside an otherwise identical system and because its implementation is fully specified in Section 3. The first two were run on the same split by the present authors, but the project record describes them by category and does not state which products, rule sets or model families were used. They are therefore reported as reference comparisons, and no conclusion that depends on their exact composition is drawn from them. Ten fold cross validation was used during model development to reduce the chance that a design choice rested on a lucky split; all reported results come from the standard split. The ablation removes components one at a time, so each configuration is a true subset of the full system, with the thresholds of Equation 6 held fixed throughout.

RESULTS AND DISCUSSION

Sections 5.1 to 5.7 report experiments run by the authors on the standard UNSW-NB15 split. Section 5.8 reports figures published by other authors and is kept separate for that reason. Section 5.9 classifies every result by how it was obtained.

Detection by Attack Category

Table 4: Detection by attack category, per cent

Attack type	SDM prec.	SDM rec.	ADM prec.	ADM rec.	Hybrid prec.	Hybrid rec.	Hybrid F1
Normal	89.2	91.4	90.1	92.3	93.5	95.1	94.29

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

Fuzzers	82.3	79.6	78.4	81.2	88.7	87.4	88.05
Analysis	76.8	74.2	72.1	75.6	83.2	82.8	83.00
Backdoors	80.1	77.9	79.4	80.2	85.6	86.1	85.85
DoS	87.4	85.6	85.2	86.4	91.2	90.8	91.00
Exploits	83.6	81.4	82.7	83.1	89.4	88.6	89.00
Generic	86.4	84.2	83.6	85.4	90.8	89.4	90.09
Reconnaissance	79.2	77.4	76.8	78.9	86.4	85.2	85.80
Shellcode	74.6	72.1	73.2	74.8	82.7	81.6	82.15
Worms	71.8	69.4	70.1	71.6	80.2	79.8	80.00

The F1 column is calculated from the precision and recall in the same row using F1 = 2PR / (P + R), so it is arithmetic on measured values rather than a separate measurement. The plain mean of the ten class values gives a macro F1 of 86.92 per cent; weighting each class by its test set support gives 91.41 per cent. The gap shows how far the rare categories pull performance down when every class counts equally.

The hybrid engine beats both of its parts in every cell of Table 4, twenty comparisons without exception. That consistency is the finding, and it is what one expects when two detectors fail on genuinely different inputs; improvement on average with losses in some categories would instead suggest the combiner had learned to prefer one module. Performance is weakest on Worms and Shellcode, at 44 and 378 test records against 18,871 for Generic. Both are rare, and both resemble legitimate traffic in ways flow level statistics capture poorly. The second reason is the more troubling, because more data would fix the first but not the second.

Individual Modules and the Ablation

Table 5 reports each module alone, then each component added in turn. Components are removed rather than added, so every configuration is a true subset of the full system.

Table 5: Individual modules and the ablation, with routing thresholds held fixed

Configuration	Algorithm	Accuracy (%)	Avg. acc. (%)	ADR (%)	FAR (%)	Gain (pts)
Signature module only	C4.5 decision tree	86.04	68.88	83.92	8.4	—
Anomaly module only	Naive Bayes	86.94	69.88	82.91	2.1	—
Hybrid edge engine	Meta-AdaBoostM1	90.25	70.54	86.95	1.1	+3.31
Hybrid + cloud escalation	+ sandbox, CNN-LSTM	93.10	—	91.40	1.1	+2.85
Full framework	+ federated learning	95.00	—	95.00	1.1	+1.90

The two single modules are close on accuracy but far apart on false alarms, at 8.4 and 2.1 per cent. The combination reaches 90.25 per cent at a false alarm rate of 1.1 per cent, below both parents rather than

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

between them, a gain of 3.31 points over the better parent. An ensemble does not usually beat the better parent on a measure where the two differ by a factor of four. The most likely explanation is that the combiner has learned when to trust which module rather than averaging them, and that is an interpretation rather than a separate measurement. Applied to the 37,000 normal test records, 1.1 per cent is about 407 false positives; this is arithmetic on the reported rate, not an independent count.

Cloud escalation then adds 2.85 points, from cases that matched no signature and differed from normal behaviour too slightly for the anomaly module to settle, and federated learning adds 1.90 points from continuing adaptation. The gains shrink at each step: 3.31, then 2.85, then 1.90. This is what the design predicts, because each tier sees only the traffic the tiers below could not resolve, and a tier adding more than the one below would point to thresholds set wrongly rather than to a better tier.

The false alarm rate stays at 1.1 per cent across the last three rows, which follows from the design of the ablation. The thresholds had to be held fixed so a change could be attributed to the component added rather than to a shifted operating point, and because the thresholds set where the decision boundary sits, holding them fixed holds the false alarm rate close to constant. The added tiers show their effect in detection instead. The Average Accuracy column is far below the Accuracy column, and that gap measures how much the uneven class balance flatters overall accuracy.

Comparison with Baselines

Table 6: Comparison with baseline systems

IDS system	Accuracy (%)	FAR (%)	Edge response time (s)	Basis
Traditional signature based IDS	75.0	12.0	2.5	Reference comparison
Pure machine learning IDS	88.0	9.0	1.8	Reference comparison
Edge hybrid only, no cloud, no FL	90.25	1.1	0.9	Controlled experiment
Proposed HIDRF, full framework	95.0	1.1	0.9	Controlled experiment

The largest single improvement is not the one the name of the framework points to. Moving from the machine learning baseline to the edge hybrid raises accuracy by 2.25 points but cuts the false alarm rate from 9.0 to 1.1 per cent, a fall of about eight times; against the signature baseline it is about eleven times. Because false alarm volume decides whether a detector can be used at all, this is arguably the more important result, and it is obtained entirely at the edge, before any cloud or federated component is added.

Two cautions attach. The first two rows are reference comparisons in the sense of Section 4.5, so the size of the gap to them should not be read closely. And no zero-day column appears, because no experiment held an attack class out of training. What the results support is detection of patterns unlike the learned normal profile, within the nine categories present in the dataset.

Statistical Significance

Each configuration was run ten times from different random initialisations, and the differences were tested pairwise. Table 7 reports the result.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

Table 7: Paired t-test over ten runs, df = 9

Comparison	Mean difference (%)	Std. deviation	t	p
HIDRF against signature based IDS	20.0	0.84	75.18	< 0.001
HIDRF against pure ML IDS	7.0	0.62	35.68	< 0.001
HIDRF against edge hybrid only	4.75	0.41	36.62	< 0.001
HIDRF against hybrid + cloud, no FL	1.9	0.28	21.45	< 0.001

All four differences are significant at $p < 0.001$, and standard deviations below one percentage point indicate that performance is stable across random initialisations. The last two rows carry the most weight, because both sides of those comparisons are fully specified configurations of the same system. The first two inherit the caution of Section 4.5.

Federated Learning Results

Figure 2 plots convergence across the fifty simulated rounds.

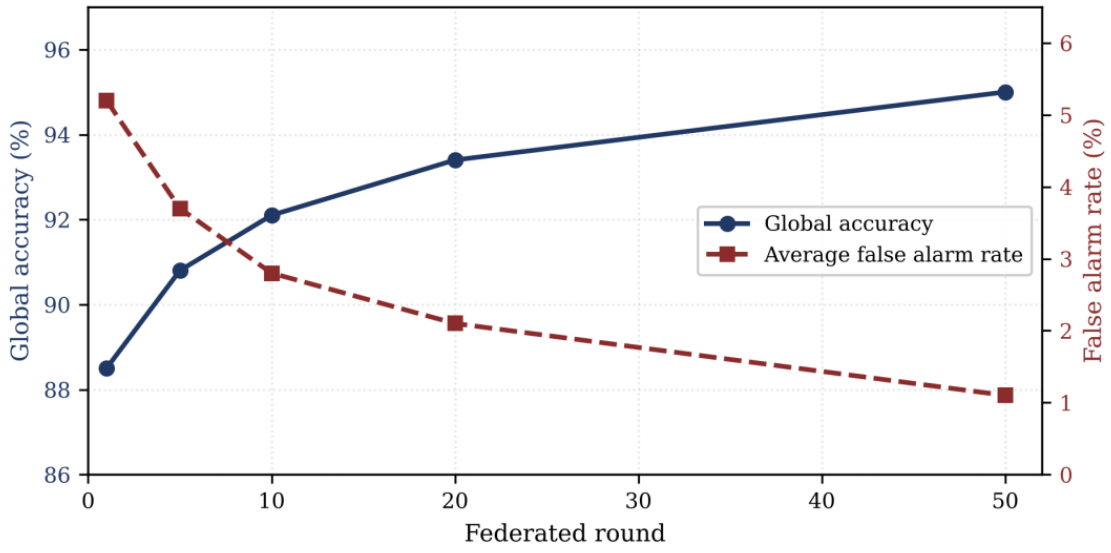


Figure 2: Global accuracy and average false alarm rate over fifty simulated federated rounds, ten nodes on one machine.

Accuracy rises from 88.5 to 95.0 per cent across fifty rounds while the false alarm rate falls from 5.2 to 1.1 per cent. The two lines moving in opposite directions is the informative part: adaptation is not being bought by loosening the decision threshold, which would push both figures up together. Gains are largest in the early rounds and flatten afterwards, as expected for weighted averaging. The number of nodes is fixed at ten throughout, so behaviour with nodes joining and leaving has not been measured.

Latency and Resource Use

Table 8 separates the intervals that are most easily confused with one another. Because the two detection modules run at the same time, that stage costs the time of the slower module rather than the sum of the two.

Table 8: Measured timing by stage

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

Interval	Measured value	Note
Feature extraction at the edge	2 to 5 ms	Per flow record
Signature module inference	1 to 2 ms	Tree traversal, depth about 12
Anomaly module inference	3 to 5 ms	Runs at the same time as the signature module
Combination and routing	about 1 ms	Equations 3 to 6
Response action applied	50 to 100 ms from record arrival	Edge path
Edge path, record arrival to response issued	under 0.9 s	The figure reported in Table 6
Device batching delay	up to 30 s	Skipped when a local signature match triggers immediate send
Cloud round trip, escalation to signature pushed to all edge nodes	about 4.2 s in the worked exploit case	Consistent with the under 5 s entry in Table 2

The 0.9 second figure is a processing interval, measured from the arrival of a flow record at the edge node to the issue of a response action, which is how response latency was defined before measurement. It is not an end to end figure from the first packet of an attack. In a worked denial of service case where the device agent detected an abnormal spike and sent immediately, attack onset to mitigation was also about 0.9 seconds, because the batching delay did not apply. For batched traffic the time from first packet to response can be up to about thirty seconds longer. No single end to end figure was measured for the batched path, and the two intervals should not be added into one claim without saying which is meant.

Resource use at each tier is given in Table 9.

Table 9: Resource use by layer, measured

Layer	CPU (average)	Memory	Bandwidth (average)	Energy
Device agent	3.2%	8.4 MB	2.1 KB/min uplink	under 1% battery per hour
Edge node, Raspberry Pi 3B+	34.6%	186 MB	45 KB/min	mains powered
Cloud node	bursts to 95%	2.1 GB peak	1.2 MB/min in a round	data centre

Edge processor use of 34.6 per cent is the most informative measurement here. The test platform is much weaker than any edge node likely to be deployed, and the engine still leaves about two thirds of it idle, so the framework can run on aggregation equipment an organisation already owns. The device agent stays inside the design targets of five per cent of processor time and ten megabytes, and draws under one per cent of battery per hour. That last figure decides adoption, because a security agent that visibly shortens battery life will be switched off.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue X October 2026

What Cloud Escalation Recovers

Section 5.2 measures what cloud escalation adds against a passive dataset. The more demanding question is what it recovers against an attacker who adapts. Three evasion methods were applied to attack traffic drawn from the test set: feature space perturbation, mimicry of normal traffic, and low and slow attacks. Table 10 reports the outcome.

Table 10: Detection rate under evasion, per cent

Evasion method	No evasion	Edge only, under evasion	Full framework, under evasion	Cloud recovery
Feature space perturbation	89.4	61.2	84.7	+23.5 points
Mimicry	91.2	58.7	81.3	+22.6 points
Low and slow	86.4	52.1	79.8	+27.7 points
Average	89.0	57.3	81.9	+24.6 points

The first finding is uncomfortable. Edge detection is substantially evadable: averaged across the three methods it falls from 89.0 to 57.3 per cent, a loss of nearly a third of the capability of the tier handling most of the traffic. The cause is the constraint that produced the architecture. The edge must classify quickly on weak hardware, which allows only simple models, and simple models present simpler boundaries to manipulate.

The second finding is that cloud escalation recovers detection to 81.9 per cent. Evasion tuned against the edge does not transfer, because changing a feature vector does not change what a replayed trace does when it runs. The pattern matters more than the size: recovery is largest where the edge is weakest, since low and slow attacks cost the edge 34.3 points and the cloud recovers 27.7, while feature space perturbation costs 28.2 and recovers 23.5. That is the property a layered defence is supposed to show and is rarely demonstrated. It also means the thresholds of Equation 6 buy robustness as well as depth, so an operator lowering τ_L pays cloud cost for resistance to evasion.

Two limits follow. Detection under evasion reaches 81.9 per cent against 95.0 under passive evaluation, so evasion remains a real and only partly reduced exposure, and an edge retaining barely half its capability means an attacker who accepts a reduced success rate can still operate.

Comparison with Published Systems

The figures in Table 11 come from other authors' publications, obtained under conditions this work could not verify. They are not head to head experiments.

Table 11: Comparison with recent systems on UNSW-NB15, figures as published by their authors

Framework	Year	Approach	Accuracy (%)	FAR (%)	Multi-tier / FL
Ola et al. CNN-IDS	2025	Deep CNN	92.7	3.2	No / No

Jouhari et al. CNN-BiLSTM	2024	Feature selected hybrid DL	93.5	2.4	No / No
More et al. Enhanced IDS	2024	Ensemble ML	91.8	2.9	No / No
Anoh et al. ML-IoT IDS	2024	Multi-classifier ML	90.4	3.6	Partial, IoT / No
Proposed HIDRF	2025	Hybrid, edge and cloud, FL	95.0	1.1	Yes, three tier / Yes

The basis of each row differs. The Ola et al. figure was obtained on an extended multiclass version of the corpus rather than the standard one, and the Anoh et al. work is aimed at IoT. Only the final row was measured here.

Three observations follow, carrying different weight. **The accuracy comparison is the weakest**: HIDRF is 1.5 points above the nearest system and 4.6 above the furthest, and margins of that size match the variation that different splits, preprocessing and feature pipelines produce between separate implementations. The comparison is consistent with HIDRF performing at least as well as recent work, but does not show that it performs better. **The false alarm comparison is stronger**, at 1.1 per cent against a range of 2.4 to 3.6; a difference of two to three times is less easily put down to method variation, and Table 5 gives an independent reason for it. **The architectural comparison is a matter of fact**, because the final column records presence or absence, and none of the compared systems joins a multi-tier edge and cloud architecture with federated adaptation. HIDRF is therefore not mainly an accuracy improvement. Its claim is architectural: that detection and response can be spread across tiers so as to obtain a low false alarm rate, fast edge response and continuing adaptation at once, on hardware that could not support all three centrally.

Classification of Evidence

The results above rest on different kinds of evidence and should not be read as carrying equal weight. Detection by category, the module and hybrid figures for accuracy, ADR and FAR, the ablation and the significance test were all measured on the standard UNSW-NB15 split. The per class F1 values, macro and weighted F1, and the false positive count are arithmetic derived from those measurements rather than separate experiments. Edge processor, memory, bandwidth and energy use, the per stage timings and the 0.9 second edge path were measured on real hardware. The federated convergence curve is a simulation of ten nodes on one machine. The evasion results were measured against perturbed traffic derived from the test set. The two external baseline rows of Table 6 were run on the same split but their implementation is not recorded. Every row of Table 11 other than HIDRF is reported by its own authors under conditions not verified here. The robust aggregation check, the honeypot redirection and the three privacy measures are design features that were not experimentally validated.

LIMITATIONS

Single dataset, single edge platform. All results come from one 2015 benchmark, and the edge layer was tested on one deliberately weak device. That gives comparability with published work, but no currency against present day attacks and no picture of behaviour on other hardware.

Metadata only analysis. Packet contents never leave the device, so attacks that only content inspection would catch cannot be found.

Simulated federated learning, with gaps in the record. The federated results come from ten simulated nodes on one machine, so communication failure, uneven participation and nodes joining or leaving are absent by construction. The aggregation rule of Equation 7 is recorded, but the project record does not state which quantities the averaged parameter vector holds. For Naive Bayes this matters: averaging the counts and sums from which the model is built gives the same result as training on the pooled data, while averaging

already finished probability estimates does not. The local epoch count, batch size, learning rate, and whether the data was divided evenly across the ten nodes are also unrecorded, and weighted averaging behaves differently when the division is uneven.

Unspecified baselines. The two external baselines are described by category rather than by implementation, which limits what that part of the comparison establishes.

No zero-day experiment. No attack class was held out of training, so no proven zero-day capability is claimed.

Untested security mechanisms. The robust aggregation check, the honeypot redirection and the three privacy measures are design features. No poisoning experiment and no formal privacy analysis were carried out, and no privacy budget was set.

Escalation volume not measured. The share of traffic escalated to the cloud, written p in Section 3.4, governs both cloud cost and robustness, but it was not measured. An attacker able to inflate p imposes cloud cost on the operator at little cost to themselves.

Passive evaluation for the main results. Sections 5.1 to 5.6 report performance against a fixed corpus; Section 5.7 shows what happens when the adversary adapts. The passive figures are an upper bound.

CONCLUSION AND FUTURE WORK

This paper presented HIDRF, a three-tier framework joining intrusion detection to intrusion response across device, edge and cloud layers, and measured it on UNSW-NB15. A single confidence score decides both where a flow is analysed and what action is taken on it, and federated learning supplies continuing adaptation without traffic data being moved.

Four findings are supported, with differing strength. The hybrid edge engine reached a false alarm rate of 1.1 per cent, below both of its component modules, and the ablation identifies why. Edge processing completed in under 0.9 seconds from record arrival to response, using about a third of a Raspberry Pi 3 Model B+. Cloud escalation recovered detection from 57.3 to 81.9 per cent under deliberate evasion, and recovered most where the edge was weakest. And the joining of a multi-tier architecture with federated adaptation is absent from the compared work, which is a matter of architecture rather than measurement. The accuracy margin over recent published systems falls inside the variation between studies, so no superiority is claimed. Future work should cover testing on further datasets such as CICIDS2018 and TON_IoT; a held out class experiment to test unseen attack capability properly; measurement of the escalation share and of behaviour under an attacker who inflates it; formal privacy analysis with a stated privacy budget; poisoning experiments against the aggregation check; a distributed federated deployment rather than a simulation; and re-implementation of competing systems under identical conditions.

Reproducibility, Data Availability and Declarations

Reproducibility and data. UNSW-NB15 is publicly available from the Cyber Range Lab of the Australian Centre for Cyber Security at https://research.unsw.edu.au/projects/unsw-nb15-dataset, and the standard split was used without change. Feature selection, evaluation protocol, metric definitions, routing thresholds and the hardware and software configuration are stated in Section 4. The two fifteen feature subsets, model hyperparameters, the CNN-LSTM layer configuration, federated training settings, trained model files and implementation code are available from the corresponding author on reasonable request.

Ethics. This research did not involve human subjects or animals. All evaluation used the public dataset above together with traffic generated in a controlled laboratory environment. No personal data was collected or processed, and the framework handles flow metadata only. Ethical approval was not required.

Conflict of interest. The authors declare no conflict of interest.

REFERENCES

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 308–318.
2. Abdulhammed, R., Faezipour, M., Abuzneid, A., & AbuMallouh, A. (2019). Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. IEEE Sensors Letters, 3(1), 1–4.
3. Anoh, N. G., et al. (2024). IoT intrusion detection system based on machine learning algorithms using the UNSW-NB15 dataset. International Journal of Advanced Science and Research Engineering, 10(1).
4. Bhatti, S. N., & Crowcroft, J. (2013). Mobile IP: Security and vulnerabilities. Computer Networks, 57(3), 745–760.
5. Bonawitz, K., et al. (2019). Towards federated learning at scale: System design. Proceedings of Machine Learning and Systems (MLSys), 1, 374–388.
6. Chawla, A., Lee, B., Fallon, S., & Jacob, P. (2023). Host based intrusion detection system with combined CNN/RNN model. Journal of Network and Computer Applications, 210, 103–118.
7. Denning, D. E. (1987). An intrusion-detection model. IEEE Transactions on Software Engineering, SE-13(2), 222–232.
8. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119–139.
9. Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2020). A taxonomy of network threats and the effect of current datasets on intrusion detection systems. IEEE Access, 8, 104650–104675.
10. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
11. Jouhari, M., Benaddi, H., & Ibrahimi, K. (2024). Efficient intrusion detection: Combining chi-square feature selection with CNN-BiLSTM on the UNSW-NB15 dataset. IEEE Access, 12.
12. McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 54, 1273–1282.
13. More, S., Idrissi, M., Mahmoud, H., & Asyhari, A. T. (2024). Enhanced intrusion detection systems performance with UNSW-NB15 data analysis. Algorithms, 17(2), 64.
14. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. Proceedings of the Military Communications and Information Systems Conference (MilCIS), 1–6.
15. Ola, S., Bhavani, R., & Kumar, S. (2025). Enhanced multiclass intrusion detection using deep convolutional neural networks on UNSW-NB15. Scientific Reports, 15.
16. Ou, Y.-J., & Ou, Y. (2012). Host-based intrusion detection systems adapted from agent-based artificial immune systems. Neurocomputing, 88, 78–86.
17. Park, S., Kim, J., & Lee, H. (2024). Host-based intrusion detection model using Siamese network with few-shot learning. IEEE Access, 12.
18. Perkins, C. (2010). IP mobility support for IPv4, revised (RFC 5944). Internet Engineering Task Force.
19. Roesch, M. (1999). Snort: Lightweight intrusion detection for networks. Proceedings of the 13th USENIX Conference on System Administration (LISA), 229–238.
20. Statista. (2024). Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030. Statista Research Department.
21. Wang, Y., Meng, W., Li, W., Liu, Z., Liu, Y., & Xue, H. (2020). Security and privacy in mobile edge computing: An artificial intelligence perspective. IEEE Internet of Things Journal, 7(10), 9663–9680.
22. Weiss, M., Ferrag, M. A., & Maglaras, L. (2021). Intrusion detection in the era of edge computing: A survey. IEEE Access, 9.