Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Look-Ahead Leakage, Drift and the Narrow Value of Device Telemetry in IIoT Intrusion Detection

Okoi Michael Obeten

michealobeten@auchipoly.edu.ng

Auchi Polytechnic

Omowumi Hafsat Aliu

Auchi Polytechnic

Research Article

Keywords:

Posted Date: November 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11599661/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Look-Ahead Leakage, Drift and the Narrow Value of Device Telemetry in IIoT Intrusion Detection

Okoi Michael Obeten[1*] and Omowumi Hafsat Aliu[1]

[1*]Department of Artificial Intelligence, Auchi Polytechnic, Auchi, Edo State, Nigeria.

*Corresponding author(s). E-mail(s): michealobeten@auchipoly.edu.ng;
Contributing authors: aliuomowumi@auchipoly.edu.ng;

Abstract

DataSense pairs network traffic with device telemetry for Industrial Internet of Things (IIoT) intrusion detection, and many studies assume that adding telemetry helps. We test this with a protocol that splits every attack session by time, and measure what telemetry adds with a modality information gain (MIG), the calibrated cross-entropy gain in bits with cluster-bootstrap intervals. A random row split inflates seven-class macro-F1 from 0.731 to 0.973. Random splits by time block that leave a minute between training and test windows still score 0.952, so about 91% of the inflation comes from look-ahead, training on later parts of the same session, and little from window overlap. Under the time-ordered split, brute-force and man-in-the-middle (MITM) detection collapse late in sessions, where the traffic changes; a model trained on the same number of more recent windows restores macro-F1 to 0.903. Telemetry adds +0.493 to +0.798 bits for MITM at 1 to 4 s windows (Holm-adjusted $\boldsymbol{p \leq 0.010}$) and nothing from 5 s on, also with training sets of equal size. Almost all of it comes from one scenario, impersonation, whose recall rises from 0.02 to 0.47 at 1 s; the two spoofing scenarios gain little. At 10 s telemetry lowers detection of an unseen malware family from 98.1% to 90.4%. Class-wise conformal sets stay close to 95% coverage and turn most MITM errors into explicit ambiguity. Telemetry is worth collecting only for attacks that change what the message broker sees, at short windows, and test windows must come after training windows.

Keywords: Industrial Internet of Things, intrusion detection, data leakage, multimodal fusion, conformal prediction, concept drift

1 Introduction

The Industrial Internet of Things (IIoT) connects sensors, actuators, gateways and cloud services across manufacturing, energy and process control [1, 2]. This connectivity widens the attack surface in ways that classic information-technology security was not designed for [3, 4], and attacks on industrial systems can cause outages and physical harm [2, 5]. Machine-learning intrusion detection systems (IDSs) are the main automated defence, and tree ensembles remain strong on tabular IoT traffic [6–8]. Reported accuracies on recent IoT benchmarks often exceed 95% [9–12].

Benchmarks have also become richer. Early datasets held only network traffic [13], while newer ones add more devices and attack types [10, 14]. DataSense, released by the Canadian Institute for Cybersecurity in 2025, goes a step further: it records network traffic and device telemetry from the same industrial testbed at the same time, and publishes both as aligned feature windows of 1

to 10 seconds [1]. This invites a natural hypothesis, common in work on multimodal and cyber-physical fusion [5, 15, 16]: that adding telemetry to network features should improve detection, at least for attacks that are hard to see on the wire.

Testing that hypothesis on DataSense is harder than it looks, for two reasons. First, DataSense windows are not independent. Each attack scenario is a continuous session, and about half of successive 10-second windows start only 5 s apart, so they share half of their traffic. A random row split, the default in most IDS studies, then places near-copies of test windows in the training set. It also trains on parts of each session that come later than the windows it tests, which a deployed detector never sees. Such leakage is a known source of inflated results in machine learning and in security in particular [17–20]. Second, the usual tool for judging what a modality contributes, SHapley Additive exPlanations (SHAP) attribution [21, 22], measures how a fitted model *uses* a feature. It does not measure what that feature adds that other features do not already carry. A model can use telemetry heavily while telemetry adds nothing, and the reverse.

This paper re-evaluates multimodal intrusion detection on DataSense with both problems addressed. It asks a sharper question than whether fusion helps: for which attack category, under which network monitoring limit, and at which window length is device telemetry worth collecting? The answer it reaches is narrower and more useful than the usual claim in either direction.

Our contributions are as follows.

1. **A leakage-controlled protocol for DataSense and an explained inflation.** We split every session by time with purged boundaries, keep all devices of one time slot together, and add an unseen-scenario test that holds out whole attack scenarios. The common random row split inflates seven-class macro-F1 by 24 points (from 0.731 to 0.973), consistently across three model families, while binary detection is unaffected. Random splits by time block, with a median gap of 63 s between a test window and the nearest training window, still score 0.952. About 91% of the inflation at 10 s (87% at 1 s) therefore comes from look-ahead, training on later parts of the same session, and only 0.021 from overlap and adjacency.
2. **A modality information gain (MIG) with cluster-bootstrap intervals.** We measure, in bits and per attack category, how much telemetry reduces the calibrated cross-entropy of a model that already sees network features. MIG is a class-conditional, log-loss version of leave-one-covariate-out importance [23] applied to a whole modality. Read next to SHAP usage and permutation reliance, it separates what a model uses from what a modality adds.
3. **A precise, conditional answer about telemetry.** Across ten window lengths and three seeds, telemetry adds information about man-in-the-middle (MITM) attacks at 1 to 4 s (MIG +0.493 to +0.798 bits, Holm-adjusted $p \leq 0.010$) and none from 5 s on, and the gain falls with window length (Spearman $\rho = -0.78$). The gain survives training sets cut to the 10-second size. It is concentrated in one scenario, impersonation (+4.3 to +6.4 bits at 1 to 4 s), with a smaller gain for IP spoofing and almost none for ARP spoofing. At 10-second windows telemetry adds nothing under full or reduced network visibility, and it hurts generalisation to an unseen malware family. The random split shows at most 37% of the short-window gain, so leakage can hide the value of a modality as well as inflate accuracy.
4. **A diagnosis of the real failure mode.** Under the controlled split, MITM and brute-force windows late in a session are classified as other categories while the model stays confident. Brute-force recall falls from 0.67 in the validation period to 0.00 as soon as the held-out period begins, and MITM recall to 0.09 by the end of the session. The traffic itself changes: brute-force windows reach 62 to 72 destination ports in the held-out period against at most 22 before it. A model trained on the same number of windows taken from later in the session raises macro-F1 from 0.731 to 0.903, so recency, not data quantity, repairs it, and the stacking gain we first observed is this same adaptation, not telemetry.
5. **Class-wise conformal prediction and a network-first escalation rule with a coverage guarantee.** Conformal sets calibrated on recent windows keep coverage under this

drift and mark the uncertain windows. Conformal modality escalation (CME) collects telemetry only when the network evidence is ambiguous, and keeps class-wise coverage for any fixed gate, a direct property of split conformal prediction that lets the gate be chosen for cost alone. On DataSense at 10 seconds it brings no saving, which we report as a direct consequence of the MIG result.

These results also correct an earlier, unpublished version of our own analysis of DataSense. Run under a random row split, it found that network features dominate and that MITM was the single category where telemetry contributed. Under leakage control, network dominance holds but absolute performance falls sharply, and the MITM exception holds only at windows of 4 s or less and mainly for one scenario.

The rest of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the data, the protocol and the methods. Section 4 reports the results, Section 5 discusses them, Section 6 states the limitations and Section 7 concludes.

2 Related work

2.1 IIoT intrusion detection datasets

KDD-Cup99 and NSL-KDD were criticised for not reflecting modern network traffic [13]. Edge-IIoTset covers 14 attacks against IoT and IIoT connectivity protocols [14]. CICIoT2023 captures traffic from more than 100 IoT devices with 33 attacks in seven groups [10], and the Gotham Dataset 2025 focuses on reproducibility with an emulated testbed of 78 devices [24]. DataSense is the first of these to release synchronised network and telemetry features from industrial sensors, IoT devices and infrastructure, with network-derived (`network_*`) and log-derived (`log_*`) columns kept apart [1]. DeepSense, by the same group, proposed an adaptive ensemble framework built on DataSense; to our knowledge it does not report the separate contribution of each modality [25].

2.2 Machine learning for IIoT intrusion detection

Tree ensembles remain competitive with deep models on tabular IoT data. XGBoost is often the strongest base learner in stacking and comparison studies [6, 26, 27], and Random Forest, LightGBM and XGBoost all perform well under top-K feature selection [28, 29]. Deep models such as CNN-GRU, attention-based recurrent units and vision transformers have been applied to Edge-IIoTset, CICIoT2023 and related data [30–32], usually with modest gains at higher cost. Nearly all of these studies evaluate with random row splits.

2.3 Multimodal fusion for IoT security

Data fusion for intrusion detection is surveyed by Ahmad and Alsmadi, who group methods by data, feature and decision level [5]. Multimodal representations of network bytes and text [33], fusion of traffic feature graphs and network feature sets for few-shot detection [34] and fused physical sensor signals [15] all report gains from combining sources. Few studies break this claim down by attack category or compare against tuned single-modality baselines under a protocol that controls leakage.

2.4 Evaluation pitfalls, leakage and temporal bias

Kapoor and Narayanan list leakage as a leading cause of irreproducible machine-learning results [17]. In security, Arp et al. identify sampling bias and data snooping among the most common pitfalls [18], Engelen et al. show labelling and flow-construction errors in CICIDS2017 [19], and Pendlebury et al. show that ignoring time inflates malware-detection results [20]. For data with temporal or grouped structure, blocked and grouped cross-validation is the standard remedy [35, 36]. We are not aware of a study that measures this effect on DataSense.

2.5 Concept drift and conformal evaluation in security

Concept drift, a change in the relation between inputs and labels over time, is well studied [37].

Transcend and its successor use conformal evaluation to detect drift in malware classifiers and to reject uncertain decisions [38, 39]. Split conformal prediction gives finite-sample coverage under exchangeability [40, 41], class-wise (Mondrian) versions give coverage per class [42], and the least ambiguous set-valued classifier (LAC) score yields the smallest sets for a given coverage [43]. We use these tools to make the effect of drift visible and to bound errors per attack category.

2.6 Measuring what a modality contributes

SHAP is the standard way to explain IDS decisions [21, 44–46]. Permutation importance measures how much a fitted model relies on a feature [47, 48]. Neither measures what a feature adds beyond the others. That requires comparing models with and without the feature group. Leave-one-covariate-out (LOCO) importance refits a model without a feature and measures the change in prediction error [23], and the conditional predictive impact of Watson and Wright tests the same kind of loss difference with knockoff features and gives formal inference [49]. Information-decomposition methods split what two modalities carry into redundant, unique and synergistic parts [50]. With cross-entropy as the loss, the difference can be read in bits as usable information [51, 52]. Our MIG is a class-conditional, log-loss version of LOCO applied to a whole modality, with refitting rather than knockoffs and with intervals from a clustered bootstrap. Its contribution is its use for modality decisions under a leakage-controlled protocol, not a new estimator.

2.7 MITM detection and window length

MITM attacks in IoT are usually carried out by Address Resolution Protocol (ARP) or IP spoofing. They are studied with machine learning for the trade-off between accuracy and cost [53], with cross-protocol evidence [54] and with deep models in smart homes [55]. Yang et al. found that the time-window length is an important setting for MITM detection from network flows, with their best results at long windows of 1,800 s [56]. We study the much shorter windows that DataSense publishes and ask the related question of when telemetry adds information, scenario by scenario.

3 Materials and methods

3.1 Data and preprocessing

We use the DataSense feature release [1]. Each row is one time window for one device. We use the 10-second windows as the main setting (30,030 windows: 16,350 attack, 13,680 benign) and the 1-second windows for replication (227,191 windows: 90,391 attack, 136,800 benign). Each file has 76 network and 8 telemetry features, plus labels, device names and timestamps. As in the earlier version of this analysis, we drop the 13 string-valued columns (lists of IP addresses, MAC addresses, ports and protocols, and the list of telemetry data types). Each has a numeric count companion that we keep, and dropping the raw strings prevents a model from memorising attacker addresses. This leaves 64 network and 7 telemetry features (`log_data-ranges_avg`, `log_data-ranges_max`, `log_data-ranges_min`, `log_data-ranges_std_deviation`, `log_data-types_count`, `log_interval-messages`, `log_messages_count`). There are no missing values. Missing values, had there been any, would be filled with training-part medians only.

Labels come at four levels. We use the seven attack categories plus benign (`label2`) as the main task. The categories are brute force, distributed denial of service (DDoS), denial of service (DoS), malware, MITM, reconnaissance (recon) and web attacks. Binary attack-or-benign detection is a second task, and the fine-grained attack labels (`label4`, 83 attack labels plus benign) as a third. The data contain 38 devices. Table 1 gives the category counts.

3.2 Testbed, feature pipelines and monitoring points

DataSense was recorded in the IoT and IIoT laboratory of the Canadian Institute for Cybersecurity [1]. Industrial sensors attached to Arduino boards and commercial IoT devices such as cameras and smart plugs send their data over a dual-band Wi-Fi access point to an Eclipse Mosquitto broker for Message Queuing Telemetry Transport (MQTT)

running on a Raspberry Pi; further Raspberry Pi units act as edge nodes and attackers, and a server hosts the cloud layer [1]. The two modalities are therefore observed at two different points of the system. The network features come from packet capture: a managed switch mirrors the traffic of the access point, router and attacker nodes to a capture host running TShark [1]. The 64 numeric network features we use describe one device's traffic in one window: packet counts by direction, packet, header and payload sizes, inter-arrival times, TCP flag counts and the numbers of distinct addresses, ports and protocols. The telemetry features come from the broker: Filebeat on the MQTT broker forwards its logs to an Elasticsearch cluster [1], and the 7 features summarise one device's messages in one window (number of messages, interval between messages, minimum, mean, maximum and standard deviation of the reported values, and number of data types). Collecting telemetry for intrusion detection therefore means reading and shipping broker logs in addition to the mirrored traffic. The visibility profiles in Section 3.7 mimic monitors that see less than a full packet capture: a monitor without per-packet timing, a vantage point that does not see addresses and ports, for example behind address translation or with anonymised exports, and a monitor that keeps only counters.

3.3 Sessions, overlap and the leakage-controlled protocol

We define a session as all windows of one fine-grained attack label, or the whole benign capture, which gives 84 sessions. Within a session, windows are ordered by start time. Successive 1-second windows of one device start 1 s apart. For 10-second windows the step alternates: 52% of steps are 5 s and 47% are 10 s, so about half of neighbouring windows overlap by half (Fig. 1d). The 8-second and 9-second files behave in the same way (steps of 4 or 8 s, and 9 or 5 s), while windows of 7 s or less never overlap.

Blocked split. Inside each session we sort the distinct time slots and assign the first 55% to training and the next 15% to validation. The last 30% form the held-out period. At each boundary the time slot nearest the boundary is purged, so that no training window shares time with a validation or test window. The held-out time slots are then split at random, whole slot by whole slot, into calibration A (7.5%), calibration B (7.5%) and test (15%), so that all devices of one time slot stay together. Sessions with fewer than eight time slots go entirely to training (dos_syn-flood-port-557 at 10 s, five windows), so that scenario is never tested. Purging removed 281 windows at 10 s and 324 at 1 s. Fig. 1a shows the design and Table 1 the resulting counts.

Random split. For comparison we also use the common row-level split with the same proportions, stratified by category.

Unseen scenarios. To test generalisation to attacks never seen in training, we run a five-fold grouped cross-validation in which every fine-grained attack label is a group, so each fold holds out whole attack scenarios. The benign capture is cut into ten time blocks that act as groups. Within each training fold a stratified 15% is kept for early stopping.

3.4 Models, tuning and calibration

The main model is XGBoost [57]. To give every feature set the same chance, we tune it separately for the network-only, telemetry-only and fused sets with an equal budget of 16 candidates each (the default plus 15 random draws over depth, learning rate, minimum child weight, row and column sampling and L2 penalty), chosen by validation macro-F1. The selected settings were: network, depth 8, learning rate 0.1, min child weight 1, subsample 1.0, column sample 0.8, L2 5.0; telemetry, depth 4, learning rate 0.2, min child weight 1, subsample 0.8, column sample 0.8, L2 5.0; fused, depth 10, learning rate 0.1, min child weight 1, subsample 0.8, column sample 1.0, L2 1.0. Trees are added up to 1,000 with early stopping after 50 rounds without improvement on validation. Random Forest [47] (300 trees, minimum leaf size 2, balanced class weights) and LightGBM [58] (400 trees, 63 leaves, learning rate 0.1, balanced class weights) serve as robustness checks. Every configuration runs with three seeds (42, 7 and 123). The blocked split itself is fixed, so seeds vary model randomness only. Predicted probabilities are calibrated by temperature scaling on validation [59].

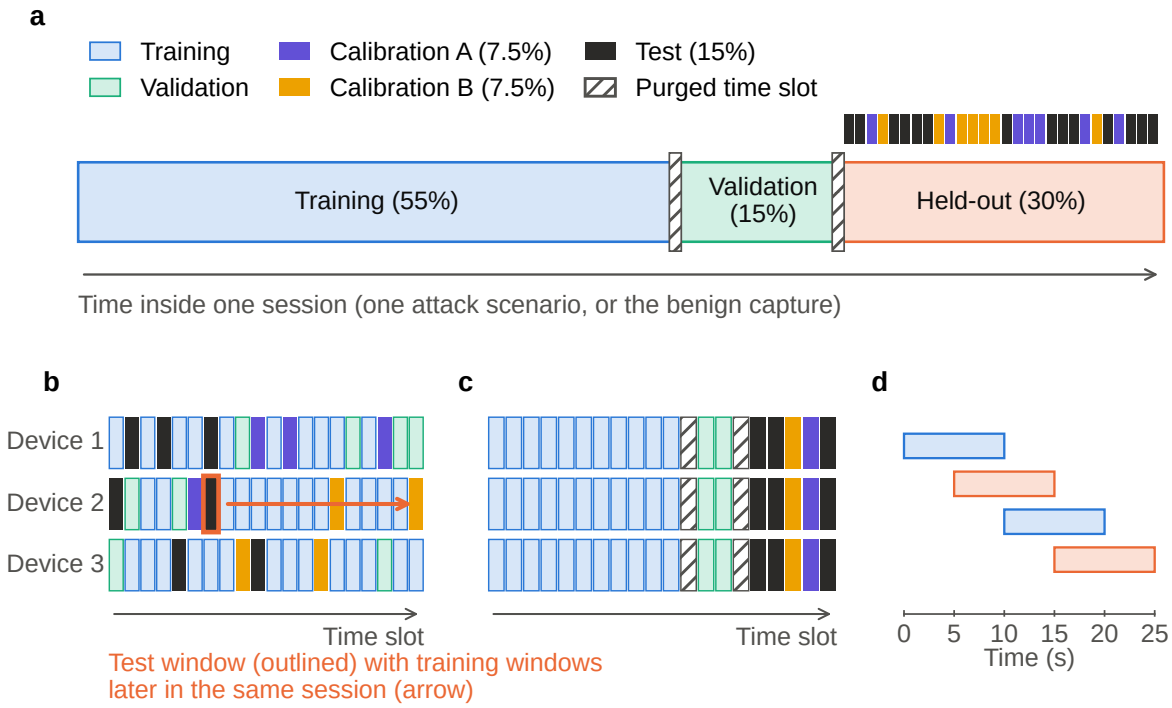


Fig. 1 Leakage-controlled protocol. **a** Each session is split by time into training, validation and held-out periods, with the boundary time slot purged. Held-out time slots are assigned at random, whole slot by whole slot, to two calibration sets and a test set. **b** A random row split assigns each window on its own, so the training set contains windows from later in the same session than a test window (arrow). **c** The blocked split keeps all devices of one time slot together and places every test window after the training period. **d** About half of successive DataSense 10-second windows of one device start 5 s apart and overlap by half; colours only separate successive windows. Panels b and c are schematic

3.5 Modality information gain

Let $\hat{p}_N(y \mid x^{\text{net}})$ be the calibrated network-only model and $\hat{p}_F(y \mid x^{\text{net}}, x^{\text{tel}})$ the calibrated fused model, both trained on the same windows. For test windows of category c,

$$\text{MIG}_c = \frac{1}{n_c} \sum_{i:\, y_i = c} \Big[\log_2 \hat{p}_F(y_i \mid x_i^{\text{net}}, x_i^{\text{tel}}) - \log_2 \hat{p}_N(y_i \mid x_i^{\text{net}}) \Big]. \quad (1)$$

MIG is the drop in cross-entropy, in bits, that telemetry gives a model that already sees the network features. It is a held-out, model-class-specific quantity in the spirit of usable information [51, 52]: it measures what this model class can extract from telemetry beyond the network features on these data. It is not an estimate of the conditional mutual information $I(Y; X^{\text{tel}} \mid X^{\text{net}})$ itself, and it depends on calibration. The two models in each comparison use the same training windows, seed and early-stopping data, so the difference reflects the added features rather than random state. A positive value means telemetry makes the true category more probable. A negative value means that adding telemetry made the model worse on held-out data, which is a sign of overfitting or of a shift in the telemetry signal. We also report the reverse direction, network given telemetry, and an overall value over all windows.

Windows that share a time slot are correlated, so we compute 95% intervals with a cluster bootstrap [60, 61] that resamples whole time slots, with 2,000 resamples. Differences in per-category F1 between two models use the same paired cluster bootstrap.

3.6 Three lenses on one modality

For the fused model we report three quantities side by side. *Usage* is the share of exact TreeSHAP

attribution [22] that falls on telemetry features for the true class, on a class-balanced sample of 2,064 test windows. *Reliance* is the drop in F1 when all telemetry columns are shuffled together across test windows (30 repeats) [48]. *Information* is MIG. The three answer different questions, and the gap between them is itself informative.

3.7 Network visibility profiles

A monitor does not always see every network feature. We therefore repeat the analysis under reduced network visibility: *no timing* removes inter-packet time features (59 network features kept); *no addressing* removes IP, MAC and port counts (55 kept), as at a vantage point beyond a switch; *counters only* keeps only count features (21 kept); and two random subsets keep 25% (16) and 10% (6) of the network features, each drawn three times. Telemetry is always the same 7 features.

3.8 Fusion strategies

To answer the concern that concatenation is not real fusion, we compare five strategies: early fusion (one model on concatenated features); late fusion by averaging the network and telemetry probabilities; late fusion with a network weight tuned on validation; stacking [62], a multinomial logistic regression fitted on validation to the standardised log-probabilities of the network and telemetry models; and CME (below).

3.9 Conformal prediction and conformal modality escalation

We use split conformal prediction with the LAC score $s(x, y) = 1 - \hat{p}(y \mid x)$ [43] and class-wise (Mondrian) thresholds [42]. For category c, let $\hat{q}_c$ be the $\lceil (n_c + 1)(1 - \alpha) \rceil$-th smallest score among the n_c calibration windows of that category. The prediction set is $C(x) = \{y : s(x, y) \leq \hat{q}_y\}$. If a category has fewer than $1/\alpha - 1$ calibration windows, its threshold falls back to the pooled threshold and its guarantee becomes marginal. We report $\alpha \in \{0.10, 0.05, 0.02\}$ and use 0.05 as the main level. The seven-class accuracy of the network model under the blocked split is 0.924, so at $\alpha = 0.10$ the procedure must leave some windows uncovered and returns an empty set for 5.3% of them; at $\alpha = 0.05$ this falls to 1.9%.

CME reads network features first and collects telemetry only when a gate $E(x) \in \{0, 1\}$ fires. Its score is

$$s_{\mathrm{CME}}(x, y) = \big(1 - E(x)\big)\big(1 - \hat{p}_N(y \mid x^{\mathrm{net}})\big) + E(x)\big(1 - \hat{p}_F(y \mid x^{\mathrm{net}}, x^{\mathrm{tel}})\big). \quad (2)$$

Proposition 1 (Class-wise coverage for any fixed gate) *Let the models $\hat{p}_N, \hat{p}_F$ and the gate E be built only from training, validation and calibration A data. Let the class-wise thresholds be computed from s_{CME} on calibration B. If the calibration B windows of category c and a test window of category c are exchangeable, then $\Pr\{Y \in C_{\mathrm{CME}}(X) \mid Y = c\} \geq 1 - \alpha$ for every category c with at least $1/\alpha - 1$ calibration windows.*

Proof Given the data used to build $\hat{p}_N$, $\hat{p}_F$ and E, the function s_{CME} is fixed and does not depend on calibration B or the test window. The statement is then the class-conditional split-conformal guarantee applied to the fixed score s_{CME} [41, 42]. □

The guarantee does not depend on how good the gate is, so the gate can be chosen for cost. We compare four gates at matched acquisition rates. The *conformal gate* fires when the network-only conformal set (thresholds from calibration A) is not a single label. The *MIG-informed gate* fires when that set is empty or contains a category in a set G, where G holds the categories whose MIG on validation plus calibration A is at least 0.10 bits with an interval above zero. The *confidence gate* fires when the maximum network probability is below a threshold set to match the rate, and the *random gate* fires at random at the same rate. Because the held-out slots are interleaved within each session, calibration and test windows come from the same period. This matches a deployment that recalibrates on recent labelled windows.

3.10 Window length, adaptation and drift analyses

Window lengths. DataSense publishes windows of 1 to 10 s. For every length we repeat the network-only, telemetry-only and fused XGBoost runs under the blocked split and under the random split, with three seeds each, and compute F1 and MIG as above. The tuned settings from the 10-second data are reused at every length; as a check,

Table 1 Windows per category in the 10-second data and in each part of the blocked split, with the 1-second totals. Calibration combines calibration A and B

Category	10 s total	Train	Validation	Calibration	Test	1 s total
Benign	13,680	7,524	2,014	2,014	2,052	136,800
Brute force	364	199	52	54	55	1,868
DDoS	3,234	1,767	458	471	488	18,056
DoS	3,280	1,793	452	469	494	18,420
Malware	1,444	804	206	211	212	7,541
MITM	1,471	816	223	212	214	8,062
Recon	6,005	3,400	849	849	855	33,648
Web	552	302	77	80	83	2,796
Total	30,030	16,605	4,331	4,360	4,453	227,191

we also tune the network and fused models on the 1-second data (eight candidates each, the default plus seven random draws from the same grid). To measure leakage directly, we count the share of random-split test windows whose time span overlaps a training window of the same session, either from the same device or from any device.

Block splits. To separate look-ahead from overlap and adjacency, we add random splits by time block. Inside every session, consecutive time slots are grouped into blocks of one slot or of 15, 30 or 60 s; whole blocks are assigned at random to training, validation and held-out parts in the same 55/15/30 proportions, and training slots next to a non-training slot are purged. Training and test blocks are therefore interleaved in time, so the model still sees later parts of a session, but near neighbours are removed as the blocks grow. For each test window we record the time to the nearest training window of its session. These splits are run at 10 s and 1 s with three seeds.

Training size. To test whether the short-window telemetry gain is only a data-size effect, we cut the training set at 1 to 4 s to the size of the 10-second training set (16,605 windows) by drawing whole time slots at random inside each session, with three seeds.

Statistics across window lengths. We pool the three seeds by averaging each window's cross-entropy over seeds, compute a two-sided slot-bootstrap p-value for MIG at each length, and apply a Holm correction across the ten lengths [63]. A Spearman correlation tests for a trend with window length. Because MITM has only three sessions (ARP spoofing, IP spoofing and impersonation) and brute force two, we also report MIG for each session and a two-stage bootstrap that resamples sessions and then time slots inside them [64]. We count test windows whose true-class probability falls below 10^{-6} or 10^{-3} to check that MIG is not driven by a few extreme values.

Drift in the data. To see whether late windows really differ, we report medians of selected network features in tenths of each session, and we label each window by its nearest category centroid in the training data (network features scaled by training medians and interquartile ranges), which needs no trained model. To separate recency from data quantity, we train network models with a fixed tree count and temperature set on calibration A on three sets of windows: the original training period (0 to 0.55 of each session), a period of the same size taken later (0.15 to 0.70), and training plus validation (0 to 0.70).

Telemetry volume. From the count columns we compare the number of broker messages and of network packets per device and window.

Stacking controls. Stacking is fitted on the validation period, which lies closer in time to the test windows than the training period. We separate the two possible sources of its gain with two controls: a stacker fitted in the same way on the network outputs only, and network and fused models refitted on training plus validation windows, with the tree count found by early stopping and the temperature set on calibration A. Both are run at 10 s (four visibility profiles) and 1 s (full and no addressing) with three seeds.

Drift by position. For each window we compute its relative position inside its session, on the same time-slot scale as the blocked split, and report recall, accuracy and mean confidence of the

network-only model in four bins: the validation period (0.55–0.70) and three tenths of the held-out period. Class-wise conformal coverage at $\alpha = 0.05$ is reported for the same held-out bins, with thresholds from calibration A and B together. These analyses use seed 42 at full visibility.

Telemetry feature ablation. To find which telemetry feature carries a signal, we add each telemetry feature alone to the network features and, separately, remove each from the fused set, and compute MIG against the corresponding baseline. All models in this analysis use the same fused settings, so the comparison isolates the feature (seed 42, 1 s and 10 s).

3.11 Inference latency

We time XGBoost inference on one CPU thread (Intel Xeon CPU @ 2.20GHz, 2 logical cores) after 30 warm-up calls. We report the median and 95th percentile over 1,000 single-window predictions and the per-window time over 30 batches of 1,000 windows. Times cover model inference only, not packet capture, feature extraction or telemetry collection.

3.12 Reproducibility

All experiments run in three Colab notebooks on a CPU runtime, each reusing the cached data and models of the previous one (Python 3.13.16, NumPy 2.1.3, pandas 2.2.3, scikit-learn 1.6.1, XGBoost 3.4.1, LightGBM 4.6.0, SciPy 1.16.3). Every number in this paper is read from the result tables the notebook writes. The notebooks, the result tables and the scripts that build the figures and tables will be released publicly (see Statements and Declarations).

Use of AI tools. Claude (Anthropic), a large language model, was used to help write the analysis notebooks and plotting scripts and to draft and edit the text of this manuscript. The authors directed the work, ran all experiments, checked the code, verified every reported number against the result tables and take full responsibility for the content.

4 Results

4.1 The random split inflates multi-class results by about 24 points

Table 2 and Fig. 2 compare the two splits. For binary detection the split makes no difference: network-only XGBoost reaches 0.988 under the blocked split and 0.987 under the random split, and the largest difference across models and feature sets is 0.004. For the seven categories plus benign, the random split gives 0.973 and the blocked split 0.731, an inflation of 0.242. Across all three model families and both the network-only and fused sets, the inflation lies between 0.225 and 0.258. The fine-grained task shows the same pattern: Random Forest drops from 0.788 to 0.576 and XGBoost from 0.786 to 0.539.

The ranking of feature sets does not change. Under the blocked split, fused minus network-only macro-F1 is -0.005 (95% interval $[-0.011, 0.002]$), and telemetry alone is weak (0.118). What changes is how good any of them is at telling attack categories apart.

Where does the inflation come from? Table 3 adds random splits by time block. Even when the median gap between a test window and the nearest training window is 63 s, and only 12% of test windows have a training window within one step, macro-F1 at 10 s is 0.952, close to the random row split (0.973). The 1-second windows agree (0.888 against 0.909). Overlap and adjacency therefore account for only about 0.021 of the inflation at both lengths. The rest, 0.221 at 10 s and 0.141 at 1 s, or about 91% and 87% of the gap, appears only when every test window comes after the training windows of its session. The inflation thus comes from look-ahead: a random split, by rows or by blocks, trains on later parts of each session than the windows it tests, and a deployed detector never has that information. Section 4.2 shows what changes later in a session, and Section 4.6 confirms the small role of overlap across window lengths.

4.2 Where the errors go: drift inside attack sessions

Table 4 breaks the result down by category. Most categories lose a few points under the blocked

Table 2 Macro-F1 under the random and blocked splits, 10-second windows: mean over three seeds, with the standard deviation in brackets

		Random split			Blocked split		
Task	Model	Network	Telemetry	Fused	Network	Telemetry	Fused
Binary	XGBoost	0.987 (0.000)	0.602 (0.011)	0.987 (0.001)	0.988 (0.000)	0.565 (0.000)	0.987 (0.000)
	Random Forest	0.985 (0.002)	0.603 (0.011)	0.985 (0.002)	0.988 (0.000)	0.566 (0.001)	0.988 (0.000)
	LightGBM	0.987 (0.002)	0.599 (0.012)	0.987 (0.002)	0.986 (0.000)	0.564 (0.001)	0.987 (0.001)
Seven-class	XGBoost	0.973 (0.004)	0.218 (0.005)	0.975 (0.005)	0.731 (0.005)	0.118 (0.003)	0.724 (0.003)
	Random Forest	0.968 (0.005)	0.192 (0.003)	0.968 (0.004)	0.732 (0.015)	0.093 (0.001)	0.743 (0.008)
	LightGBM	0.973 (0.005)	0.187 (0.005)	0.974 (0.004)	0.715 (0.002)	0.103 (0.001)	0.718 (0.001)

Table 3 Where the inflation comes from: macro-F1 of XGBoost under random splits by row, by time slot and by time block, and under the time-ordered blocked split, mean over three seeds. Gap: median time in seconds from a test window to the nearest training window of the same session. In the random splits, training and test blocks are interleaved in time; in the blocked split every test window comes after the training period

	10-second windows			1-second windows		
Split	Gap (s)	Network	Fused	Gap (s)	Network	Fused
Random row split	–	0.973	0.975	–	0.909	0.913
Random time slots	10	0.975	0.975	1	0.908	0.911
Random 15-s blocks	20	0.960	0.959	7	0.901	0.905
Random 30-s blocks	30	0.953	0.953	12	0.901	0.906
Random 60-s blocks	63	0.952	0.951	24	0.888	0.893
Blocked split (time-ordered)	1,190	0.731	0.724	1,146	0.747	0.754

split, but two collapse. Brute-force F1 falls from 0.952 to 0.000 and MITM from 0.977 to 0.365. The confusion matrices (Fig. 3) show where these windows go. 82% of brute-force test windows are labelled malware and 15% DoS. This is plausible because the Mirai family spreads by Telnet brute force [65], so the two look alike on the wire. MITM test windows scatter over DoS (27%), recon (22%), malware (14%) and DDoS (10%), and only 22% are recognised.

Because the test windows lie later in each session than the training windows, this pattern points to a change inside sessions over time. Direct evidence is in the tuning step: the same tuned network model scores between 0.868 and 0.876 macro-F1 on the validation period, which comes right after training, but 0.731 on the test period, which comes later. The network model is also confidently wrong on these windows: its mean cross-entropy on the true class is 4.71 bits for MITM and 6.85 bits for brute force, against 0.492 bits over all windows.

Fig. 4 shows the change directly. At 10 s, network-only macro-F1 falls from 0.876 in the validation period to 0.778, 0.704 and 0.698 in the three tenths of the held-out period. Brute-force recall drops from 0.67 to 0.00 as soon as the held-out period begins. MITM recall holds at 0.50 in the first held-out tenth and then falls to 0.09 in the last two. The 1-second windows show the same pattern: macro-F1 falls from 0.896 to 0.730, brute-force recall from 0.64 to 0.01 and MITM recall from 0.74 to 0.28 in the middle tenth. The other categories stay stable. Confidence does not warn of these errors: at 10 s the mean probability of the predicted label is 0.923 or higher in every bin, while accuracy falls from 0.949 to 0.911.

The traffic itself changes (Fig. 4c and d). At 10 s, the median number of destination ports in a brute-force window is 5 to 22 in the first 60% of the session and 62 to 72 in the held-out period. Median packets per MITM window rise from about 348 in the first 80% of the session to between 1,238 and 1,318 in the last fifth, and median destination ports from 2 to between 11 and 13. These late windows are not empty: the share of held-out brute-force and MITM windows with no packets is 0%. A check that needs no trained model agrees. When each window is labelled by its nearest category centroid in the training data, early brute-force windows are nearest brute force in 49% of cases, middle ones are nearest malware (100%), and held-out ones are never nearest

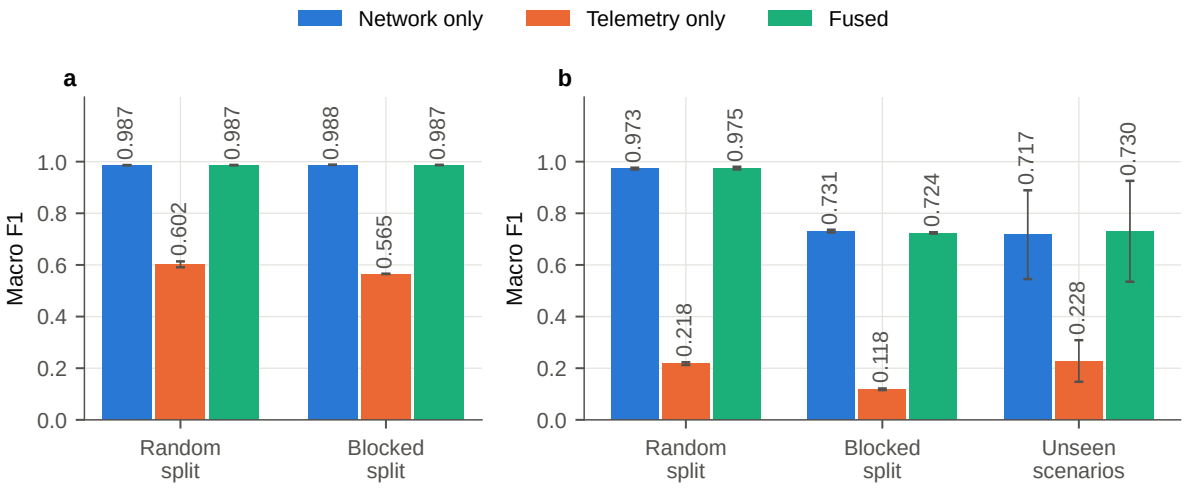


Fig. 2 Macro-F1 of XGBoost by protocol, 10-second windows. **a** Binary detection. **b** Seven attack categories plus benign, including the unseen-scenario test (mean over five folds). Error bars show the standard deviation over seeds or folds

brute force (0%). Brute-force sessions thus pass through phases, and the held-out phase resembles no brute-force traffic seen in training.

Recency, not quantity, repairs the model. A network model trained on the same number of windows (16,366 against 16,605) but taken from positions 0.15 to 0.70 of each session instead of 0 to 0.55 reaches 0.903 macro-F1 at 10 s, against 0.731 for the original model and 0.906 for a model trained on 26% more windows (0 to 0.70). Brute-force F1 rises from 0.000 to 0.794 and MITM F1 from 0.365 to 0.766. The 1-second windows agree (0.747, 0.808 and 0.806).

4.3 Unseen attack scenarios are still detected as attacks

When whole attack scenarios are held out (Fig. 5), network-only XGBoost still flags 96.1% of their windows as attacks, with 0.53% false alarms on benign windows. Category macro-F1 on the categories present in training is 0.717 (standard deviation 0.172 over folds). Telemetry alone flags only 15.4%. Adding telemetry does not help, and in one case it hurts. In fold 5 all 1,444 malware windows were held out, so malware was a family the model had never seen. Network-only XGBoost flagged 98.1% of them as attacks and the fused model only 90.4%. Web attacks are the hardest unseen family (59.8% network only, 67.8% fused).

4.4 The fine-grained task fails on sibling scenarios, not on imbalance

The 83 fine-grained attack labels are highly imbalanced: from 5 to 789 windows (median 121), with 22 labels under 30 windows (Fig. 6a). Imbalance is not what drives the errors, however. Across labels, training size has no rank correlation with blocked-split F1 (Spearman $\rho = -0.006$, $p = 0.96$; Fig. 6b). The worst labels are large ones: 8 of the 9 recon scenarios (OS scan, port scan, vulnerability scan and host-discovery pings), each with about 400 training windows, score F1 between 0.00 and 0.18, even though recon as a category scores 0.951. These are sibling scenarios that look alike in aggregated window features. MITM labels also fall under the blocked split (ARP spoofing 0.94 to 0.42, IP spoofing 1.00 to 0.66). Weighted F1 is higher than macro-F1 (0.693 against 0.576 for Random Forest) because the large benign class is easy.

4.5 Three lenses: usage, reliance and information

Table 5 and Fig. 7 put the three lenses side by side for the fused model at full visibility. The fused model does use telemetry for MITM, which receives 3.1% of its SHAP attribution, against at most 0.4% for other categories, so the usage signal

Table 4 Per-category F1 of XGBoost (mean over three seeds), 10-second windows. The last column is fused minus network under the blocked split. Network-only brute-force F1 is exactly zero under the blocked split, which alone lowers macro-F1 by about 0.12

	Random split		Blocked split			
Category	Network	Fused	Network	Telemetry	Fused	Fused − network
Benign	0.986	0.985	0.985	0.643	0.985	−0.000
Brute force	0.952	0.955	0.000	0.000	0.000	+0.000
DDoS	0.984	0.985	0.920	0.053	0.919	−0.001
DoS	0.981	0.982	0.889	0.031	0.889	−0.001
Malware	0.977	0.977	0.780	0.000	0.762	−0.018
MITM	0.977	0.980	0.365	0.042	0.328	−0.038
Recon	0.966	0.966	0.951	0.178	0.952	+0.001
Web	0.961	0.969	0.956	0.000	0.957	+0.001

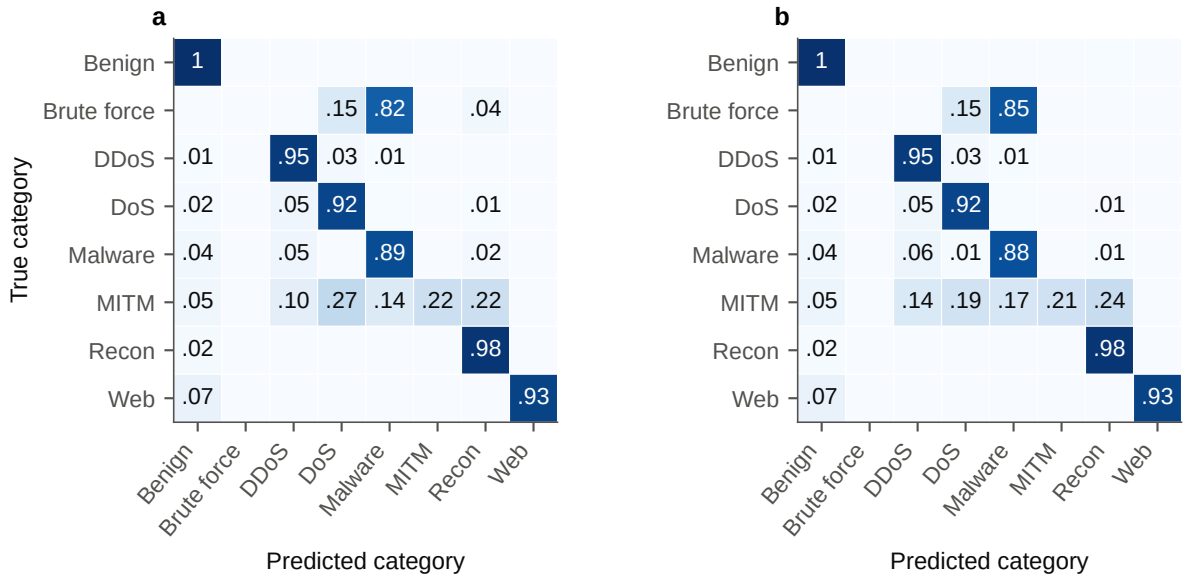


Fig. 3 Row-normalised confusion matrices under the blocked split, XGBoost, 10-second windows, seed 42. **a** Network only. **b** Fused

seen in our earlier analysis is still there. But shuffling all telemetry columns changes MITM F1 by only +0.004, and MIG for MITM is −0.051 bits with an interval of [−0.186, 0.072] that includes zero. So the model uses telemetry for MITM, yet telemetry does not improve its held-out predictions. Over all windows MIG is −0.004 bits ([−0.012, 0.005]), against 1.688 bits for network given telemetry. The highest-ranked telemetry feature is number 49 of 71 in mean absolute SHAP.

Two categories have MIG intervals above zero: brute force (+0.356 bits, [0.041, 0.670]), where telemetry raises the probability of the true class without changing a single decision (F1 stays 0.000), and recon (+0.029 bits). Telemetry makes the model worse on malware (−0.107 bits, [−0.163, −0.057]) and web (−0.046 bits). These negative values match the unseen-family result above: telemetry patterns learned in training do not carry over to later windows.

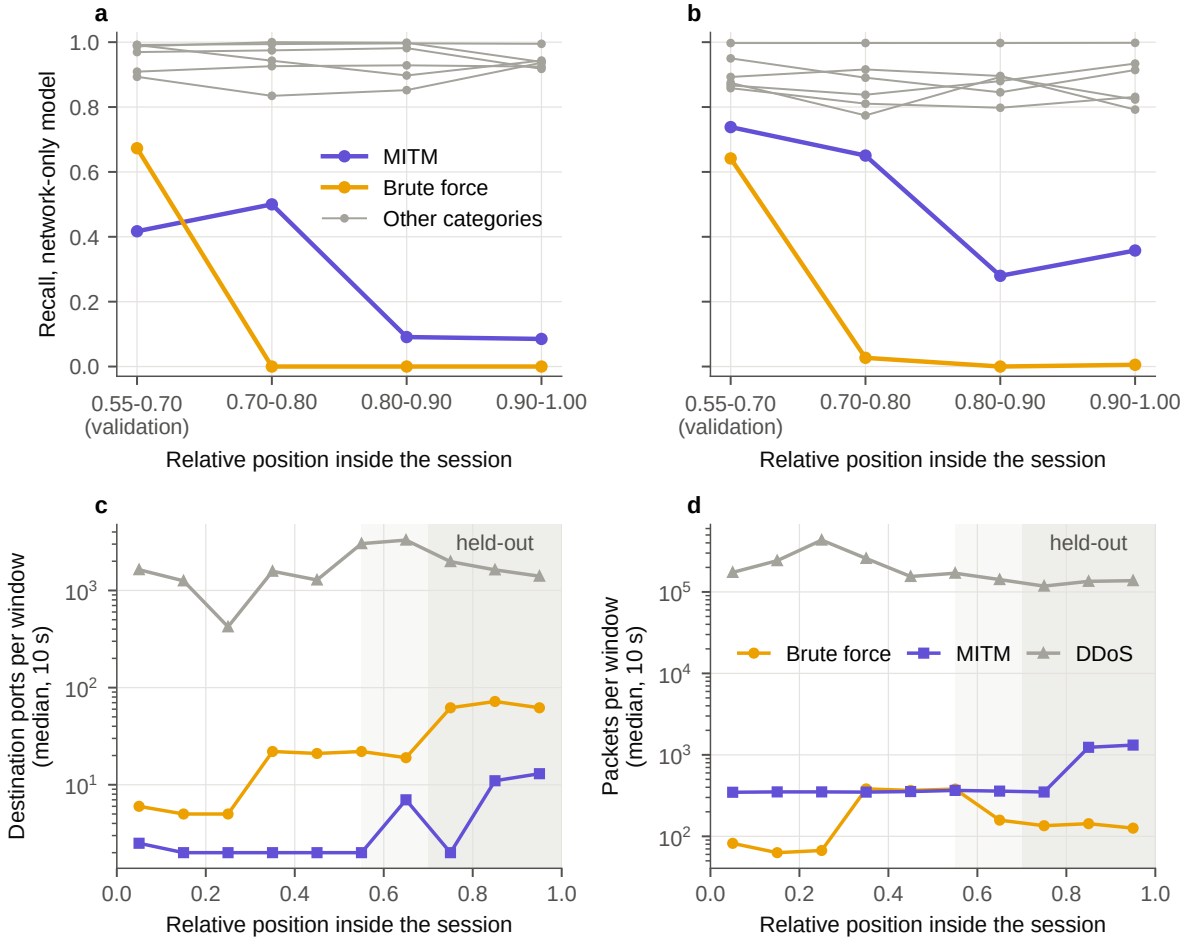


Fig. 4 Recall of the network-only XGBoost model against the relative position of a window inside its session, blocked split, full visibility, seed 42. Position 0.55–0.70 is the validation period and 0.70–1.00 the held-out period (calibration and test windows). MITM and brute force are highlighted and the other categories are shown in grey. **a** Recall, 10-second windows. **b** Recall, 1-second windows. **c** Median number of destination ports per window, 10-second windows, by tenth of the session (log scale). **d** Median number of packets per window (log scale). In c and d the darker band marks the held-out period and the lighter band the validation period; DDoS is shown as a stable reference

4.6 Window length: leakage and the value of telemetry

DataSense publishes windows of 1 to 10 s, which lets us test both the leakage mechanism and the role of window length. Fig. 8a and b answer the first question. Successive windows of one device overlap only at 8 s and above: for windows of 7 s or less, no random-split test window overlaps a training window of the same device, against 39% to 42% at 8 to 10 s. Yet the optimism of the random split is already 0.163 macro-F1 at 1 s and grows only to 0.242 at 10 s. Overlap therefore explains only a small part of the inflation, which matches the block splits of Section 4.1. Even at short windows, 75% to 76% of random-split test windows share a time slot with a training window of another device (92% to 93% at 8 to 10 s), but removing such neighbours barely lowers the score; what matters is that the random split also trains on later parts of each session. Under the blocked split no test window shares time with a training window (0%).

Table 6 and Fig. 8c and d answer the second question. At 1 to 4 s, telemetry adds +0.493 to +0.798 bits for MITM, with intervals above

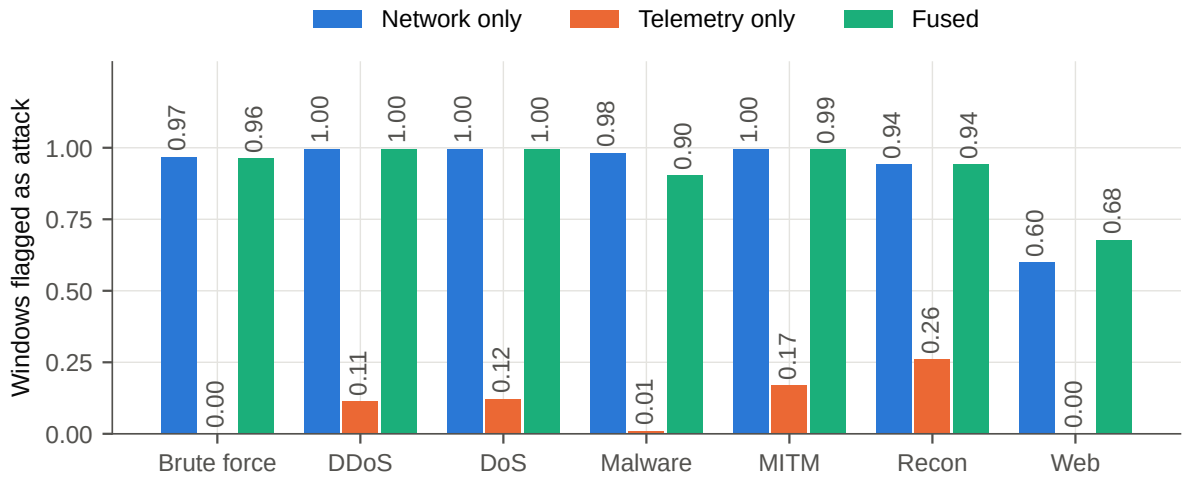


Fig. 5 Share of windows flagged as an attack (any attack label) when whole attack scenarios are held out in grouped five-fold cross-validation, pooled over folds

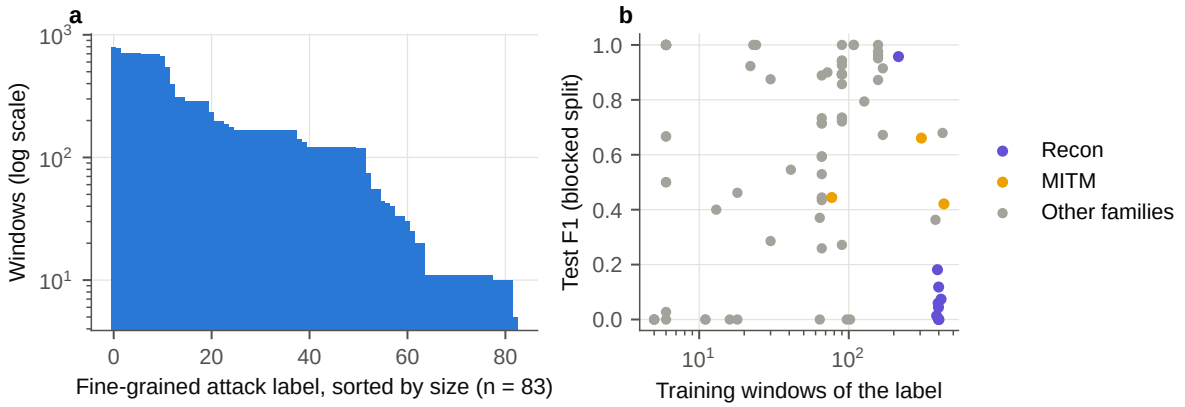


Fig. 6 Fine-grained attack labels. **a** Windows per label (log scale). **b** Test F1 under the blocked split against training windows per label (Random Forest, network only)

zero for every seed, and fused MITM F1 exceeds network-only MITM F1 by 0.06 to 0.13. From 5 s the effect is gone: no MITM interval lies above zero, and at 8 s telemetry hurts (−0.269 bits, [−0.450, −0.079]). Over all categories telemetry adds little at any length. The overall MIG is at most +0.015 bits and lies above zero for every seed only at 1 and 2 s, and fused and network-only macro-F1 differ by less than 0.02 at every length. Brute force moves the other way: at 5 of the ten lengths its MIG lies between −0.703 and −1.082 bits, with intervals below zero for every seed. Pooling the three seeds, the MITM gain at 1 to 4 s stays significant after a Holm correction across the ten lengths (adjusted $p \leq 0.010$), the loss at 8 s also survives (adjusted $p = 0.036$), and MIG falls with window length (Spearman $\rho = -0.78$, $p = 0.008$). No test window had a true-class probability below 10^{-6} under either model (at most 16% below 10^{-3}), so these values are not driven by clipped probabilities. The best macro-F1 is reached at 3 s, 0.771 fused against 0.756 network

Table 5 Three lenses on telemetry for the fused XGBoost model, blocked split, 10-second windows, seed 42. SHAP: share of attribution on telemetry for the true class. Rel.: reliance, the F1 drop when telemetry is shuffled. MIG and fused-minus-network F1 with 95% cluster-bootstrap intervals

Category	SHAP	Rel.	MIG (bits)	ΔF1
Benign	0.3%	+0.000	−0.006 [−0.01, −0.00]	−0.000 [−0.00, 0.00]
Brute force	0.0%	+0.000	+0.356 [0.04, 0.67]	+0.000 [0.00, 0.00]
DDoS	0.2%	+0.000	−0.013 [−0.03, 0.01]	−0.009 [−0.02, −0.00]
DoS	0.4%	−0.000	−0.011 [−0.03, 0.00]	+0.012 [0.00, 0.02]
Malware	0.4%	−0.001	−0.107 [−0.16, −0.06]	−0.020 [−0.03, −0.01]
MITM	3.1%	+0.004	−0.051 [−0.19, 0.07]	−0.019 [−0.06, 0.02]
Recon	0.2%	+0.001	+0.029 [0.02, 0.04]	−0.000 [−0.01, 0.01]
Web	0.0%	+0.000	−0.046 [−0.09, −0.00]	+0.000 [0.00, 0.00]
All windows	–	+0.001	−0.004 [−0.01, 0.00]	−0.005 [−0.01, 0.00]

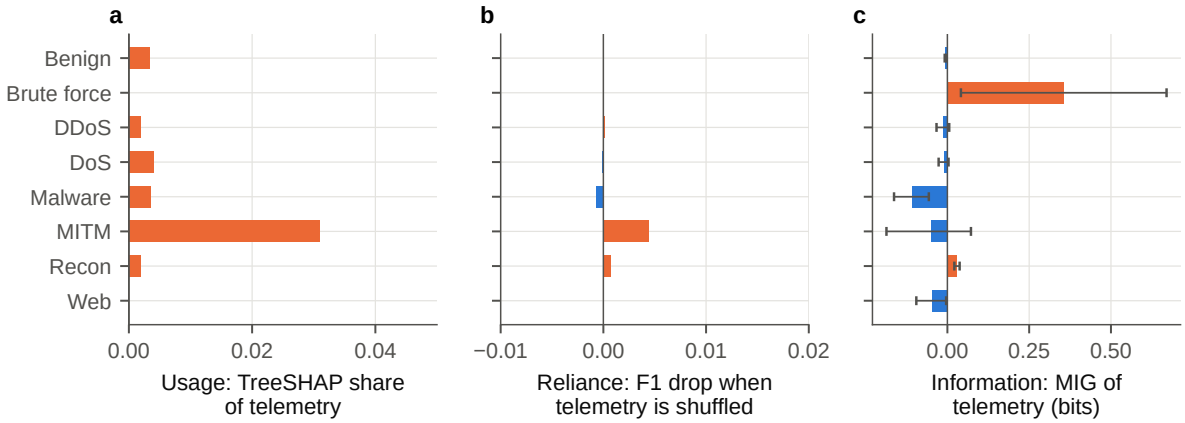


Fig. 7 Three lenses on the telemetry channel, fused XGBoost, blocked split, 10-second windows. **a** Usage: TreeSHAP share. **b** Reliance: F1 drop when telemetry is shuffled. **c** Information: MIG with 95% cluster-bootstrap intervals. Orange bars are positive and blue bars negative

only. Telemetry alone scores between 0.118 and 0.134 at every length.

The random split hides this result. Under it, MITM MIG at 1 to 4 s is only +0.116 to +0.277 bits, at most 37% of the blocked value. When later parts of the same session are in training, network features alone already place a test window, and telemetry has little left to add.

The gain is not a data-size effect. With the training sets at 1 to 4 s cut to the 10-second size (16,604 windows), MITM MIG is +0.350 to +0.683 bits, with intervals above zero for every seed, and MITM F1 still rises with telemetry (for example from 0.444 to 0.505 at 1 s and from 0.452 to 0.545 at 4 s). Retuning on the 1-second data lowers the gain to +0.232 bits ([0.108, 0.364]) but keeps it (MITM F1 0.575 to 0.630). The block splits give the same picture: at 1 s telemetry adds +0.156 to +0.285 bits for MITM under every block length, and at 10 s between −0.020 and +0.005 bits.

Which MITM attacks gain? Table 7 splits the result by scenario. Almost all of the gain comes from impersonation. Telemetry adds +4.3 to +6.4 bits for its windows at 1 to 4 s, with intervals above zero from 1 to 7 s, and lifts its recall from 0.02 to 0.47 at 1 s and from 0.08 to 0.86 at 2 s. At 1 s impersonation provides 92% of the MITM gain, although it has only 124 of the 1,172 MITM test windows. IP spoofing gains less (+0.08 to +0.38 bits at 1 to 4 s), and ARP spoofing gains clearly only at 4 s. A plausible reading is that an impersonating device connects to the broker or publishes under another identity, which shows in the broker logs, whereas ARP and IP spoofing redirect traffic

below the application layer and leave the broker's view largely unchanged. At 10 s the pattern reverses for impersonation (MIG −1.11 bits; recall 0.70 network only against 0.38 fused). Because the effect rests mainly on one session of 140 10-second windows, a bootstrap that also resamples sessions gives wide intervals (for example [0.01, 4.20] bits at 1 s, and including zero at 2 s), and the median gain per window is small (+0.059 bits at 1 s). The claim the data support is therefore narrow: telemetry helps when an attack changes what the broker sees, and only at short windows. A leaky protocol can therefore understate what a second modality contributes, not only overstate accuracy.

Which telemetry feature carries the MITM signal at 1 s? Added alone to the network features, 6 of the 7 telemetry features each add information about MITM (+0.248 to +0.409 bits, intervals above zero), the largest being `log_data-types_count`; `log_data-ranges_std_deviation` hurts (−0.125 bits). Removing any one feature from the fused model changes MITM MIG by at most 0.037 bits. The signal is thus spread over several largely redundant telemetry features rather than held by one. With the same model settings for both sets, all telemetry together adds +0.324 bits for MITM at 1 s ([0.209, 0.449]) and −0.400 bits at 10 s ([−0.514, −0.289]), where 5 of the 7 features hurt when added alone. At 1 s the MIG-informed gate of the main run selected MITM as its only telemetry category, under full visibility and without addressing features. A short-lived change in broker message counts or timing during an impersonation would be averaged away in a long window. Yang et al. also found window length to be an important setting for MITM detection, although their best window for network flows was much longer [56]; the best window appears to depend on the signal and the modality.

4.7 Reduced network visibility does not make telemetry useful at 10 s

One might expect telemetry to matter more when the network monitor sees less. At 10-second windows it does not (Fig. 9). Of the 54 category-by-profile MIG values, 5 have intervals above zero in every run and 19 have intervals below zero in every run. Removing timing or addressing features makes telemetry more harmful for MITM (−0.257 and −0.184 bits) and brute force (−0.731 and −0.473 bits). The only clearly positive MITM value appears when just 10% of network features remain (+0.098 bits).

4.8 Fusion strategies and the stacking gain

Table 8 compares the fusion strategies. Early fusion, both late-fusion variants and CME stay close to the network-only model in every profile, and the tuned late-fusion weight on the network model varies from 0.45 to 0.95 across runs, a sign that no stable telemetry weight exists. Stacking stands apart: at full visibility it reaches 0.803 against 0.731 for network only, and raises MITM F1 to 0.625, brute force to 0.185 and malware from 0.780 to 0.881. The stacker, however, is fitted on the validation period, which lies closer in time to the test windows than the training period. Its gain could therefore come from adapting to recent windows rather than from telemetry.

The two controls settle this (Table 9, Fig. 10). At 10 s, a stacker fed only the network outputs already reaches 0.797 macro-F1, against 0.803 with the telemetry outputs added, so almost all of the stacking gain comes from fitting on recent windows. Telemetry does add something at this level for MITM: the MIG of stacking with telemetry over stacking without it is +0.657 bits ([0.508, 0.803]), and MITM F1 rises from 0.598 to 0.625. Over all windows the same MIG is +0.002 bits ([−0.011, 0.015]). Refitting is far stronger than stacking. The network model refit on training plus validation reaches 0.907 macro-F1, with brute-force F1 rising from 0.000 to 0.846 and MITM F1 from 0.365 to 0.744. The refit gain over network only lies between 0.151 and 0.176 in all four 10-second profiles. After refitting, telemetry hurts at 10 s: the refit fused model scores 0.902, its MITM F1 is 0.682, and MITM MIG is −0.263 bits ([−0.361, −0.166]). The small MITM gain from telemetry in the stacker is therefore dominated by a network model trained on more recent data, and Section 4.2 showed that recency, not the extra windows, is what helps.

At 1 s the result reverses. Stacking brings no gain (0.753 and 0.753 against 0.747), and after refitting the fused model beats the network model

Table 6 Telemetry given network across window lengths, blocked split, full visibility, XGBoost, mean over three seeds. MIG in bits; $^{+}$ and $^{-}$ mark intervals above or below zero for every seed. The random-split MITM MIG (mean over three seeds) is shown for comparison. N: network only. F: fused

	MIG, MITM and brute force			MITM F1		Macro-F1		MIG
Window (s)	MITM	MITM, random	Brute force	N	F	N	F	all windows
1	$+0.493^{+}$	$+0.116^{+}$	-0.362	0.566	0.628	0.747	0.754	$+0.015^{+}$
2	$+0.750^{+}$	$+0.277^{+}$	-0.988^{-}	0.549	0.655	0.749	0.763	$+0.013^{+}$
3	$+0.798^{+}$	$+0.246^{+}$	-0.899	0.531	0.649	0.756	0.771	$+0.015$
4	$+0.698^{+}$	$+0.189^{+}$	-0.703^{-}	0.456	0.588	0.744	0.761	$+0.011$
5	$+0.127$	$+0.099$	-0.717^{-}	0.397	0.418	0.735	0.740	-0.008
6	-0.070	$+0.150^{+}$	$+0.054$	0.374	0.362	0.726	0.724	-0.007
7	$+0.188$	$+0.129$	-0.880^{-}	0.474	0.543	0.744	0.755	$+0.006$
8	-0.269^{-}	$+0.075^{+}$	$+0.358$	0.463	0.519	0.749	0.756	-0.015
9	-0.128	$+0.028$	-1.082^{-}	0.448	0.469	0.749	0.749	-0.026^{-}
10	-0.035	$+0.009$	$+0.064$	0.365	0.328	0.731	0.724	-0.006

Table 7 MIG of telemetry given network for each MITM scenario, blocked split, full visibility, mean over three seeds (bits; $^{+}$ and $^{-}$ mark 95% slot-bootstrap intervals above or below zero), with the recall of impersonation and IP spoofing for network-only (N) and fused (F) models

	MIG (bits)			Impersonation recall		IP spoofing recall	
Window (s)	ARP spoofing	Impersonation	IP spoofing	N	F	N	F
1	$+0.02$	$+4.27^{+}$	$+0.08^{+}$	0.02	0.47	0.85	0.90
2	-0.01	$+6.36^{+}$	$+0.25^{+}$	0.08	0.86	0.78	0.86
3	$+0.19$	$+6.29^{+}$	$+0.11^{+}$	0.01	0.78	0.66	0.80
4	$+0.23^{+}$	$+4.33^{+}$	$+0.38^{+}$	0.07	0.66	0.47	0.64
5	-0.20^{-}	$+1.97^{+}$	$+0.10$	0.12	0.17	0.36	0.44
6	-0.02	$+2.18^{+}$	-0.77^{-}	0.12	0.37	0.23	0.16
7	$+0.06$	$+2.73^{+}$	-0.40^{-}	0.18	0.67	0.49	0.52
8	-0.19^{-}	$+1.13$	-0.80^{-}	0.44	0.52	0.32	0.43
9	-0.12	-0.32	-0.08	0.33	0.26	0.33	0.41
10	-0.23^{-}	-1.11^{-}	$+0.57^{+}$	0.70	0.38	0.02	0.04

(0.831 against 0.806 macro-F1; MITM F1 0.651 against 0.597), with MITM MIG of +0.194 bits ([0.101, 0.291]). The same holds without addressing features (+0.252 bits, [0.162, 0.349]). The window-length result thus survives adaptation: telemetry informs MITM at short windows and not at 10 s.

4.9 Conformal coverage under drift and the cost of telemetry

Table 10 reports the conformal results at $\alpha = 0.05$. At 10 s, network-only sets cover 0.951 of test windows, and 82.2% of windows receive a single label, which is correct 98.8% of the time. The remaining windows receive an empty set or several labels, which marks them for review. So although macro-F1 is only 0.731, the system knows which of its decisions to trust. At 1 s the sets are larger (mean size 3.81), only 46.6% are single labels, and those are correct 99.4% of the time.

Class-wise coverage is close to the 0.95 target for most categories (Fig. 11). The lowest values are for MITM (0.921 network only, 0.914 CME) and for web under CME (0.924). These classes have only 106 and 42 calibration windows and 214 and 83 test windows. The guarantee in Proposition 1 holds on average over calibration sets, and for these sizes the standard deviation of realised coverage is about 0.026 (MITM) and 0.041 (web), so these values lie within 1.5 standard deviations of the target. At $\alpha = 0.02$, 2 small categories fell back to the pooled threshold.

Coverage also holds across the held-out period. With thresholds from calibration A and B together (seed 42), network-only coverage over all test windows is 0.964, 0.950 and 0.935 in the three held-out tenths at 10 s, and between 0.968 and 0.973 at 1 s. It falls slightly below the target only

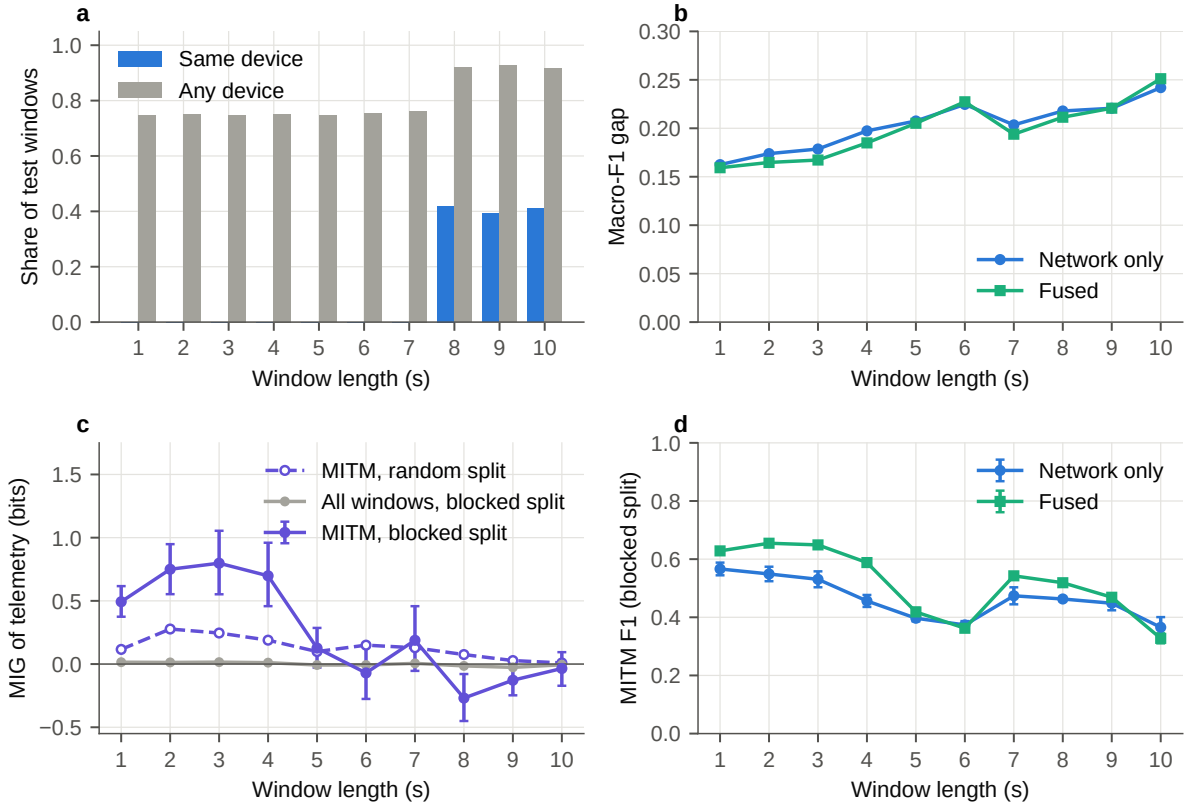


Fig. 8 Window length, leakage and the value of telemetry, windows of 1 to 10 s. **a** Share of random-split test windows that overlap in time a training window of the same session, from the same device or from any device. **b** Optimism of the random split: macro-F1 under the random split minus the blocked split (both mean over three seeds). **c** MIG of telemetry given network for MITM under the blocked split (mean over seeds; bars show the 95% cluster-bootstrap interval averaged over seeds) and under the random split, and over all windows. **d** MITM F1 under the blocked split, mean and standard deviation over seeds

Table 8 Macro-F1 by fusion strategy and network visibility profile, blocked split, 10-second windows (mean over runs; best per row in bold). Numbers in brackets give the network features kept. CME-conf. and CME-MIG: conformal modality escalation with the conformal and MIG-informed gates. Telemetry alone scores 0.117 to 0.118 in every profile

Profile	Network	Early	Late avg.	Late wtd.	Stacking	CME-conf.	CME-MIG
Full (64)	0.731	0.724	0.722	0.723	**0.803**	0.724	0.725
No timing (59)	0.746	0.718	0.725	0.748	**0.808**	0.717	0.746
No addressing (55)	0.726	0.719	0.714	0.716	**0.803**	0.721	0.726
Counters only (21)	0.627	0.627	0.649	0.650	**0.718**	0.627	0.627
Random 25% (16)	0.684	0.688	0.698	0.684	**0.765**	0.688	0.686
Random 10% (6)	0.622	0.636	0.636	0.637	**0.674**	0.636	0.639

in the last tenth at 10 s, where drift is strongest. The sets also show where the errors go. At 10 s the network model's top label is right for only 22% of MITM test windows, yet 90% of them receive a set of several labels that contains MITM; DoS appears in 45% of MITM sets, brute force in 78% and malware in 58%. For brute force the top label is right for 0% of windows, but 98% receive a set that contains brute force, and every such set also contains malware (100%). At 1 s the picture is similar (MITM: top label right for 43%, multi-label sets

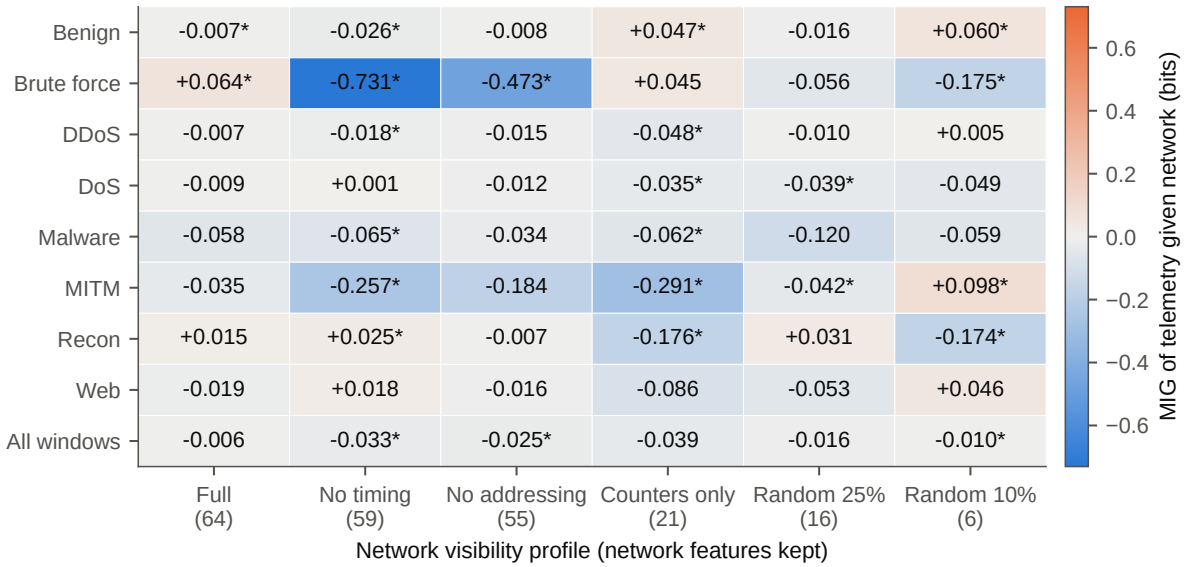


Fig. 9 MIG of telemetry given network (bits) by category and network visibility profile, blocked split, 10-second windows, mean over runs. An asterisk marks cells whose 95% interval excludes zero in every run. Numbers under each profile give the network features kept

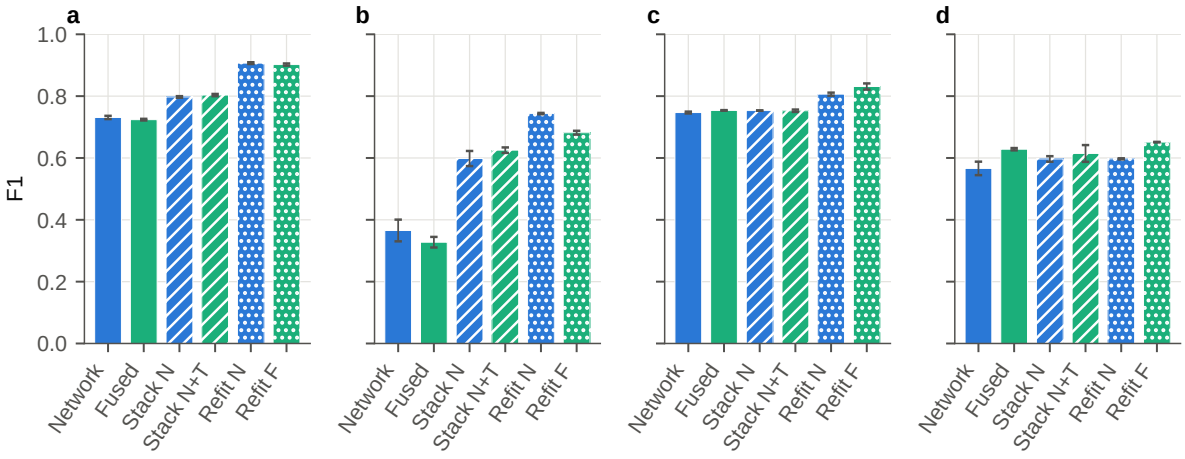


Fig. 10 Stacking and refitting controls, blocked split, full visibility, mean and standard deviation over three seeds. Network: network only. Fused: early fusion. Stack N and Stack N+T: stacker fitted on the validation period to the outputs of the network model, or of the network and telemetry models. Refit N and Refit F: network or fused model refitted on training plus validation windows. Blue bars use no telemetry and green bars use telemetry; hatching marks stacking and dots mark refitting. **a** Macro-F1, 10 s. **b** MITM F1, 10 s. **c** Macro-F1, 1 s. **d** MITM F1, 1 s

containing MITM for 88%). Conformal prediction thus turns the confident errors of Section 4.2 into explicit ambiguity between attack categories that look alike, which an analyst can resolve.

CME keeps coverage at every gate setting, as Proposition 1 says, but at 10 s it buys nothing (Fig. 12). The MIG-informed gate collects telemetry for 10.5% of windows at full visibility, and its

Table 9 Is the stacking gain telemetry or adaptation? F1 of XGBoost variants, blocked split, full visibility, mean over three seeds. Stacking: logistic regression fitted on the validation period to the outputs of the network model, or of the network and telemetry models. Refit: model refitted on training plus validation windows

	10-second windows			1-second windows		
Method	Macro	MITM	Brute force	Macro	MITM	Brute force
Network only	0.731	0.365	0.000	0.747	0.566	0.027
Fused (early)	0.724	0.328	0.000	0.754	0.628	0.020
Stacking, network outputs	0.797	0.598	0.195	0.753	0.597	0.037
Stacking, network and telemetry	0.803	0.625	0.185	0.753	0.615	0.037
Refit, network	0.907	0.744	0.846	0.806	0.597	0.428
Refit, fused	0.902	0.682	0.874	0.831	0.651	0.572

macro-F1 (0.725) lies between that of never collecting telemetry (0.731) and always collecting it (0.724). Its category set G was empty in 12 of 18 runs, because no category reliably gained from telemetry. Matched confidence and random gates give the same results. This is the expected outcome when MIG is near zero: an escalation rule cannot create information the second modality does not carry.

4.10 Inference latency and telemetry volume

Table 11 gives the inference times. A single seven-class prediction takes a median of 0.49 ms (95th percentile 0.69 ms) with network features and 0.50 ms with fused features. In batches, the network model processes about 81 thousand windows per second and the fused model about 53 thousand. With the CME escalation rate of 22.7%, the expected cost is 0.60 ms per window. These times exclude feature extraction and telemetry collection. Telemetry is light in volume. A device sends on average 5.14 broker messages per 10-second window (median 0), against 99,590 packets on average (median 56), so telemetry messages amount to about 0.005% of the packet count; MITM windows carry 12.5 messages on average. Brute-force and web windows carry no broker messages at all, so telemetry cannot help those categories, and its negative MIG for brute force reflects noise added to the fused model. The cost of telemetry is therefore less about bandwidth or inference time than about running and joining a second collection pipeline, which is what CME would save.

5 Discussion

5.1 Why the random split misled our earlier analysis

Our earlier, unpublished analysis reported near-perfect seven-class detection, network dominance, and a MITM exception based on a 10% SHAP share. Two of these claims did not hold up as stated. The near-perfect scores came from look-ahead: a random row split trains on later parts of each session than the windows it tests. Splitting by time blocks with a minute between training and test does not remove this; only a split in which every test window follows the training windows does. Overlap between windows adds only about 0.02. The MITM exception was a usage effect at 10 s: the fused model attends to telemetry for MITM, but this attention does not improve held-out predictions. A real exception does exist at windows of 4 s or less, but it belongs mostly to one scenario, impersonation, and the random split understated it. Leakage thus misled in both directions, inflating accuracy while hiding the one place where telemetry helps. Network dominance survives, now with intervals and significance tests. We report the correction openly because DataSense is new, and other studies are likely to repeat the random split.

5.2 What the results mean for practitioners

Binary detection on DataSense is robust: under every protocol, including unseen scenarios, the network-only model flags almost all attack windows at a false-alarm rate below 1%. Naming the attack category is the hard part, and it is hard because behaviour within a session changes over time, not because the network channel lacks information. Three practical steps follow from

Table 10 Conformal prediction at $\alpha = 0.05$ (target coverage 0.95), blocked split, mean over runs. Min class: lowest class-wise coverage. Singleton and Sing. acc.: share of single-label sets and their accuracy. Telemetry: share of windows for which telemetry is collected. The 1-second rows come from a single run

Method	Coverage	Min class	Set size	Singleton	Sing. acc.	Telemetry
10-second windows, full visibility						
Network only	0.951	0.921	1.25	0.822	0.988	0.000
Fused	0.954	0.941	1.29	0.815	0.986	1.000
CME, conformal gate	0.956	0.923	1.28	0.823	0.983	0.206
CME, MIG gate	0.955	0.914	1.32	0.785	0.984	0.105
Confidence gate	0.955	0.916	1.28	0.808	0.985	0.099
Random gate	0.953	0.906	1.31	0.783	0.986	0.107
10-second windows, no addressing						
Network only	0.953	0.930	1.27	0.826	0.985	0.000
Fused	0.955	0.929	1.30	0.818	0.981	1.000
CME, conformal gate	0.952	0.924	1.27	0.826	0.983	0.224
CME, MIG gate	0.952	0.916	1.29	0.805	0.983	0.013
Confidence gate	0.951	0.916	1.29	0.803	0.983	0.019
Random gate	0.951	0.915	1.29	0.804	0.983	0.020
1-second windows, full visibility						
Network only	0.970	0.939	3.81	0.466	0.994	0.000
Fused	0.972	0.945	3.75	0.489	0.993	1.000
CME, conformal gate	0.972	0.951	3.76	0.492	0.993	0.532
CME, MIG gate	0.971	0.951	3.80	0.465	0.994	0.467
Confidence gate	0.973	0.951	3.80	0.465	0.993	0.472
Random gate	0.972	0.949	3.80	0.464	0.994	0.475

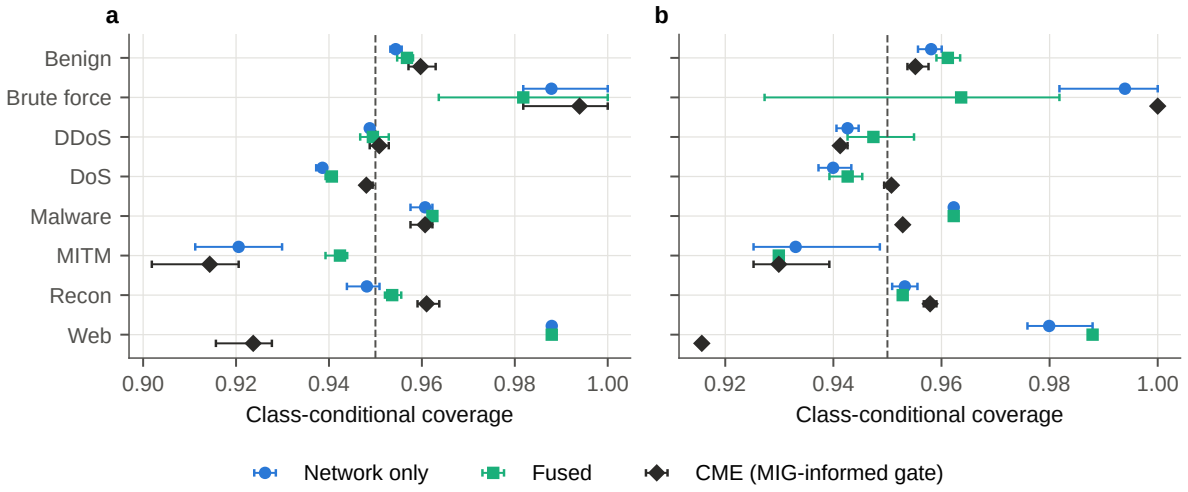


Fig. 11 Class-wise coverage at target 0.95 (dashed line), blocked split, 10-second windows. Markers show the mean over seeds and bars the range. **a** Full visibility. **b** No addressing

the results. First, adapt to recent data: a network model trained on the same number of more recent labelled windows raised 10-second macro-F1 from 0.731 to 0.903, a larger gain than any fusion strategy or window length gave. Second, use class-wise conformal prediction, recalibrated on recent windows. It keeps coverage, gives single, highly accurate labels for most windows and turns the remaining errors into small sets of similar categories for review. Third, collect telemetry only for attacks that change what the broker sees, such as impersonation, and only if the monitoring pipeline aggregates at 4 s or less. There, fusion

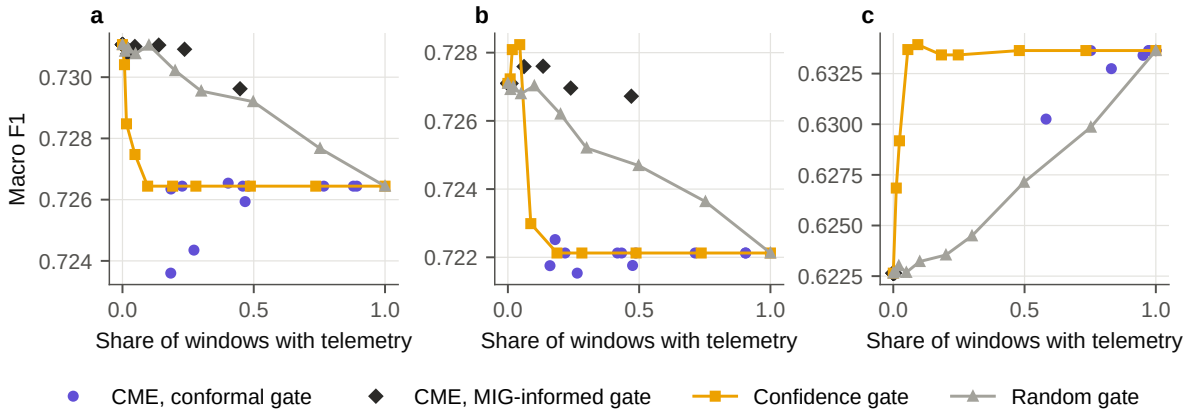


Fig. 12 Macro-F1 against the share of windows for which telemetry is collected, for CME with the conformal and MIG-informed gates and for confidence and random gates, $\alpha = 0.05$, 10-second windows, seed 42. The vertical scales are narrow: all differences within a panel are below 0.012. **a** Full visibility. **b** No addressing. **c** Counters only

Table 11 Inference latency of XGBoost on one CPU thread (Intel Xeon CPU @ 2.20GHz). Single window: median and 95th percentile over 1,000 calls. Batch: per-window time in batches of 1,000 and the resulting throughput

Task	Features	Inputs	Trees	Median (ms)	p95 (ms)	Batch (μs)	k windows/s
Binary	Network only	64	326	0.546	0.663	9.0	112
	Telemetry only	7	153	0.404	0.519	2.9	345
	Fused	71	173	0.444	0.542	5.8	173
Seven-class	Network only	64	59	0.487	0.691	12.4	81
	Telemetry only	7	58	0.581	0.693	13.0	77
	Fused	71	49	0.500	0.743	18.8	53

raises MITM F1 by 0.06 to 0.13, almost all of it through impersonation, and the refit fused model is the best 1-second model. Telemetry is cheap to carry, so its cost is mainly a second pipeline. At 10-second aggregation it brings no detection benefit, and it lowered detection of the one unseen malware family we could test.

5.3 Why SHAP alone misled

SHAP answers the question of what a fitted model looks at. Modality decisions need a different question: what one source adds when the other is already available. The two differ whenever sources are redundant or when a source's pattern changes between training and deployment, and both conditions hold on DataSense. MIG answers the second question directly, in bits, with intervals, and its sign also flags harm. We suggest that modality claims in multimodal IDS research report an information measure of this kind next to attribution.

5.4 Recommendations for DataSense users

We recommend four practices. Split by time within sessions so that every test window comes after the training windows of its session; random splits by time block are not enough, even with gaps of a minute. Purge boundary windows and keep the devices of one time slot together. Report an unseen-scenario test. Report the window length and overlap, and do not assume that non-overlapping windows remove leakage. And report per-category results with intervals, because macro-F1 hides collapses such as the brute-force and MITM cases above.

5.5 An alternative explanation for late-session errors

Late-session errors could reflect drift, but they could also reflect labels. Every window of an attack session carries the attack label, even if attack traffic thins out or changes phase late in the

session. The data argue for a real change in the traffic rather than empty windows: late windows carry packets, their destination ports and packet counts differ from those earlier in the session (Fig. 4c and d), and a model-free nearest-centroid check finds them closer to other categories than to their own training traffic. Without packet-level ground truth we still cannot tell whether every late window contains attack traffic, or whether a phase such as scanning new targets should carry a different label. Either way, the practical conclusion is the same: random splits hide the problem, and evaluation must respect time.

6 Limitations and future work

This study uses one dataset. DataSense is the most complete synchronised IIoT benchmark we know of, but the results should be repeated on other multimodal datasets. We use tree ensembles. Deep cross-modal models might extract telemetry information that trees do not, and MIG can evaluate them in the same way. The telemetry features in DataSense are aggregates. Richer telemetry, such as per-topic message timing, may carry information these features do not. The blocked split assumes that time order within a session reflects deployment order. Class-wise coverage for small categories is noisy because their calibration sets are small. The window-length sweep uses the settings tuned at 10 s and covers full visibility only; retuning at 1 s lowered but kept the MITM gain. The MITM telemetry gain rests mainly on one impersonation session of 140 10-second windows, so it should be tested on other impersonation attacks and testbeds before it is generalised. The drift, conformal-by-position and feature-ablation analyses use one seed. Refitting assumes that labelled recent windows are available, which in practice requires analyst effort. Latency was measured on a server CPU and excludes feature extraction; edge gateways will be slower.

Future work includes drift-aware training, such as windows weighted by recency or scheduled refitting, phase-aware labels for long attack sessions, active selection of the windows to label, measuring MIG for richer telemetry at short windows, and session-level detection that uses the order of windows rather than treating each window alone.

7 Conclusion

We re-evaluated multimodal intrusion detection on DataSense with a time-ordered protocol and an information-theoretic measure of what each modality adds. The common random split inflates seven-class macro-F1 by about 24 points, and almost all of this comes from look-ahead: training on later parts of the same session. Overlapping windows add little, and random splits by time block do not fix the problem. Under a time-ordered split, the real difficulty is change inside attack sessions: brute-force and MITM traffic changes late in a session, and the model stays confident while it fails. Training on more recent windows of the same size repairs most of the loss, and it explains the gain we first credited to stacking. Device telemetry carries information about MITM at windows of 1 to 4 s and none from 5 s on, and this holds with equal training sizes and after a correction for testing ten window lengths. The gain comes mainly from impersonation, an attack that changes what the MQTT broker sees. At 10 s telemetry adds no reliable information beyond network features for models trained on the training period, whether the monitor sees all network features or only some, and it lowered detection of an unseen malware family. Class-wise conformal prediction keeps coverage close to its target under this drift and turns confident errors into explicit ambiguity between similar attacks. Device telemetry matters on DataSense, but only for attacks visible at the broker and only at short windows, and only evaluations in which test windows follow training windows can show this.

Acknowledgements. We thank the Canadian Institute for Cybersecurity for releasing the DataSense dataset.

Statements and Declarations

Funding statement. The authors declare that no funding was received for this research.

Ethics statement. Not applicable. This study uses a public benchmark of machine-generated network and device data and involves no human participants, human data or animals.

Author contributions statement. O.M.O.: conceptualisation, methodology, software, data

curation, formal analysis, investigation, visualisation, writing (original draft) and writing (review and editing). O.H.A.: validation and writing (review and editing). Both authors read and approved the final manuscript.

Competing interests. The authors declare no competing interests.

Consent to participate and consent for publication. Not applicable.

Data availability. This study uses the publicly available DataSense dataset released by the Canadian Institute for Cybersecurity [1]. No new data were generated. All result tables are included with the code.

Code availability. The experiment notebooks, result tables and the scripts that build every figure and table will be made publicly available upon acceptance of this manuscript. Until then they are available from the corresponding author on reasonable request.

References

[1] Firouzi, A., Dadkhah, S., Maret, S.A., Ghorbani, A.A.: DataSense: A real-time sensor-based benchmark dataset for attack analysis in IIoT with multi-objective feature selection. Electronics **14**(20), 4095 (2025) https://doi.org/10.3390/electronics14204095

[2] Sharma, S.B., Bairwa, A.K.: Leveraging AI for intrusion detection in IoT ecosystems: A comprehensive study. IEEE Access **13**, 66290–66317 (2025) https://doi.org/10.1109/ACCESS.2025.3550392

[3] Saheed, Y.K., Omole, A.I., Sabit, M.O.: GA-mADAM-IIoT: A new lightweight threats detection in the industrial IoT via genetic algorithm with attention mechanism and LSTM on multivariate time series sensor data. Sensors International **6**, 100297 (2025) https://doi.org/10.1016/j.sintl.2024.100297

[4] Khan, I.A., Keshk, M., Pi, D., Khan, N., Hussain, Y., Soliman, H.: Enhancing IIoT networks protection: A robust security model for attack detection in internet industrial control systems. Ad Hoc Networks **134**, 102930 (2022) https://doi.org/10.1016/j.adhoc.2022.102930

[5] Ahmad, R., Alsmadi, I.: Data fusion and network intrusion detection systems. Cluster Computing **27**(6), 7493–7519 (2024) https://doi.org/10.1007/s10586-024-04365-y

[6] Almotairi, A., Atawneh, S., Khashan, O.A., Khafajah, N.M.: Enhancing intrusion detection in IoT networks using machine learning-based feature selection and ensemble models. Systems Science & Control Engineering **12**(1), 2321381 (2024) https://doi.org/10.1080/21642583.2024.2321381

[7] Azam, Z., Islam, M.M., Huda, M.N.: Comparative analysis of intrusion detection systems and machine learning-based model analysis through decision tree. IEEE Access **11**, 80348–80391 (2023) https://doi.org/10.1109/ACCESS.2023.3296444

[8] Alashjaee, A.M., Alqahtani, F.: Enhanced intrusion detection system IoT network security model by feed forward neural network and machine learning. Scientific Reports **15**(1), 36085 (2025) https://doi.org/10.1038/s41598-025-20047-0

[9] Jony, A.I., Arnob, A.K.B.: A long short-term memory based approach for detecting cyber attacks in IoT using CIC-IoT2023 dataset. Journal of Edge Computing **3**(1), 28–42 (2024) https://doi.org/10.55056/jec.648

[10] Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., Ghorbani, A.A.: CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors **23**(13), 5941 (2023) https://doi.org/10.3390/s23135941

[11] Herzalla, D., Lunardi, W.T., Andreoni, M.: TII-SSRC-23 dataset: Typological exploration of diverse traffic patterns for intrusion detection. IEEE Access **11**, 118577–118594 (2023) https://doi.org/10.1109/ACCESS.2023.3319213

[12] Khan, M.A., Khan, M.A., Jan, S.U., Ahmad, J., Jamal, S.S., Shah, A.A., Pitropakis, N., Buchanan, W.J.: A deep learning-based intrusion detection system for MQTT enabled IoT. Sensors **21**(21), 7016 (2021)

https://doi.org/10.3390/s21217016

[13] Kasongo, S.M., Sun, Y.: Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. Journal of Big Data **7**(1), 105 (2020) https://doi.org/10.1186/s40537-020-00379-6

[14] Ferrag, M.A., Friha, O., Hamouda, D., Maglaras, L., Janicke, H.: Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access **10**, 40281–40306 (2022) https://doi.org/10.1109/ACCESS.2022.3165809

[15] Latif, M., Abro, A.A., Daniyal, S.M., Algarni, A.D., Ahmad, S., Ateya, A.A., Abbasi, M.M.: AI-based intelligent sensing detection of cybersecurity threats using multimodal sensor data in smart devices. Scientific Reports **16**(1), 11091 (2026) https://doi.org/10.1038/s41598-026-40614-3

[16] Zhou, Z.: Multimodal fusion anomaly detection model for agricultural wireless sensors. Engineering Reports **6**(12), 13021 (2024) https://doi.org/10.1002/eng2.13021

[17] Kapoor, S., Narayanan, A.: Leakage and the reproducibility crisis in machine-learning-based science. Patterns **4**(9), 100804 (2023) https://doi.org/10.1016/j.patter.2023.100804

[18] Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: 31st USENIX Security Symposium (USENIX Security 22). USENIX Association, Boston, MA (2022)

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

[20] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: TESSERACT: Eliminating experimental bias in malware classification across space and time. In: Proceedings of the 28th USENIX Security Symposium, pp. 729–746. USENIX Association, Santa Clara, CA (2019)

[21] Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), pp. 4768–4777. Curran Associates Inc., Red Hook, NY, USA (2017)

[22] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.-I.: From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence **2**(1), 56–67 (2020) https://doi.org/10.1038/s42256-019-0138-9

[23] Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R.J., Wasserman, L.: Distribution-free predictive inference for regression. Journal of the American Statistical Association **113**(523), 1094–1111 (2018) https://doi.org/10.1080/01621459.2017.1307116

[24] Belarbi, O., Spyridopoulos, T., Anthi, E., Rana, O., Carnelli, P., Khan, A.: Gotham Dataset 2025: A reproducible large-scale IoT network dataset for intrusion detection and security research. arXiv:2502.03134 (2025). https://doi.org/10.48550/arXiv.2502.03134

[25] Firouzi, A., Ghorbani, A.A.: DeepSense: An adaptive scalable ensemble framework for industrial IoT anomaly detection. Sensors **26**(9), 2662 (2026) https://doi.org/10.3390/s26092662

[26] Akif, M.A., Butun, I., Williams, A., Mahgoub, I.: Hybrid machine learning models for intrusion detection in IoT: Leveraging a real-world IoT dataset. arXiv:2502.12382 (2025). https://doi.org/10.48550/arXiv.2502.12382

[27] Ibrahim, M., Al-Wadi, A.: Enhancing IoMT network security using ensemble learning-based intrusion detection systems. Journal

of Engineering Research **13**(4), 3166–3175 (2025) https://doi.org/10.1016/j.jer.2024.12.003

[28] Kouassi, B.M., Ballo, A.B., Ayikpa, K.J., Mamadou, D., Coulibaly, M.Z.J.: Top-K feature selection for IoT intrusion detection: Contributions of XGBoost, LightGBM, and random forest. Future Internet **17**(11), 529 (2025) https://doi.org/10.3390/fi17110529

[29] Shaikhanova, A., Kuznetsov, O., Tokkuliyeva, A., Ayapbergenov, K., Olzhas, S., Danir, T.: Security audit of IoT device networks: A reproducible machine learning framework for threat detection and performance benchmarking. Sensors **25**(24), 7519 (2025) https://doi.org/10.3390/s25247519

[30] Saadouni, R., Khacha, A., Harbi, Y., Gherbi, C., Harous, S., Aliouat, Z.: Secure IIoT networks with hybrid CNN-GRU model using Edge-IIoTset. In: 2023 15th International Conference on Innovations in Information Technology (IIT), pp. 150–155 (2023). https://doi.org/10.1109/IIT59782.2023.10366486

[31] Ullah, S., Boulila, W., Koubâa, A., Ahmad, J.: MAGRU-IDS: A multi-head attention-based gated recurrent unit for intrusion detection in IIoT networks. IEEE Access **11**, 114590–114601 (2023) https://doi.org/10.1109/ACCESS.2023.3324657

[32] Gueriani, A., Kheddar, H., Mazari, A.C., Sağıroğlu, Ş., Ceran, O.: SE-enhanced ViT and BiLSTM-based intrusion detection for secure IIoT and IoMT environments. In: 2025 18th International Conference on Information Security and Cryptology (ISCTürkiye), pp. 1–6 (2025). https://doi.org/10.1109/ISCTrkiye68593.2025.11224819

[33] Ullah, F., Turab, A., Ullah, S., Cacciagrano, D., Zhao, Y.: Enhanced network intrusion detection system for Internet of Things security using multimodal big data representation with transfer learning and game theory. Sensors **24**(13), 4152 (2024) https://doi.org/10.3390/s24134152

[34] Xu, C., Zhan, Y., Wang, Z., Yang, J.: Multimodal fusion based few-shot network intrusion detection system. Scientific Reports **15**(1), 21986 (2025) https://doi.org/10.1038/s41598-025-05217-4

[35] Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F.: Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography **40**(8), 913–929 (2017) https://doi.org/10.1111/ecog.02881

[36] Bergmeir, C., Benítez, J.M.: On the use of cross-validation for time series predictor evaluation. Information Sciences **191**, 192–213 (2012) https://doi.org/10.1016/j.ins.2011.12.028

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

[38] Jordaney, R., Sharad, K., Dash, S.K., Wang, Z., Papini, D., Nouretdinov, I., Cavallaro, L.: Transcend: Detecting concept drift in malware classification models. In: Proceedings of the 26th USENIX Security Symposium, pp. 625–642. USENIX Association, Vancouver, Canada (2017)

[39] Barbero, F., Pendlebury, F., Pierazzi, F., Cavallaro, L.: Transcending Transcend: Revisiting malware classification in the presence of concept drift. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 805–823 (2022). https://doi.org/10.1109/SP46214.2022.9833659

[40] Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer, New York (2005)

[41] Angelopoulos, A.N., Bates, S.: Conformal prediction: A gentle introduction. Foundations and Trends in Machine Learning **16**(4), 494–591 (2023) https://doi.org/10.1561/2200000101

[42] Vovk, V.: Conditional validity of inductive conformal predictors. In: Proceedings of the Asian Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 25, pp. 475–490 (2012)

[43] Sadinle, M., Lei, J., Wasserman, L.: Least ambiguous set-valued classifiers with bounded error levels. Journal of the American Statistical Association **114**(525), 223–234 (2019) https://doi.org/10.1080/01621459.2017.1395341

[44] Gaspar, D., Silva, P., Silva, C.: Explainable AI for intrusion detection systems: LIME and SHAP applicability on multi-layer perceptron. IEEE Access **12**, 30164–30175 (2024) https://doi.org/10.1109/ACCESS.2024.3368377

[45] Mohale, V.Z., Obagbuwa, I.C.: Evaluating machine learning-based intrusion detection systems with explainable AI: Enhancing transparency and interpretability. Frontiers in Computer Science **7**, 1520741 (2025) https://doi.org/10.3389/fcomp.2025.1520741

[46] Khan, N., Ahmad, K., Al Tamimi, A., Alani, M.M., Bermak, A., Khalil, I.: Explainable AI-based intrusion detection systems for industry 5.0 and adversarial XAI: A systematic review. Information **16**(12), 1036 (2025) https://doi.org/10.3390/info16121036

[47] Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001) https://doi.org/10.1023/A:1010933404324

[48] Fisher, A., Rudin, C., Dominici, F.: All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. Journal of Machine Learning Research **20**(177), 1–81 (2019)

[49] Watson, D.S., Wright, M.N.: Testing conditional independence in supervised learning algorithms. Machine Learning **110**, 2107–2129 (2021) https://doi.org/10.1007/s10994-021-06030-6

[50] Liang, P.P., Cheng, Y., Fan, X., Ling, C.K., Nie, S., Chen, R., Deng, Z., Allen, N., Auerbach, R., Mahmood, F., Salakhutdinov, R., Morency, L.-P.: Quantifying and modeling multimodal interactions: An information decomposition framework. In: Advances in Neural Information Processing Systems 36 (NeurIPS 2023). Curran Associates, Inc., Red Hook, NY (2023)

[51] Xu, Y., Zhao, S., Song, J., Stewart, R., Ermon, S.: A theory of usable information under computational constraints. In: International Conference on Learning Representations (ICLR) (2020)

[52] Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Wiley, Hoboken, NJ (2006)

[53] Alsaaidah, A., Almomani, O., Abu-Shareha, A.A., Abualhaj, M.M., Achuthan, A.: ARP spoofing attack detection model in IoT network using machine learning: Complexity vs. accuracy. Journal of Applied Data Sciences **5**(4), 1850–1860 (2024) https://doi.org/10.47738/jads.v5i4.374

[54] Rahman, M.M., Bouhafs, F., Hoseini, S.A., den Hartog, F.: ARProof: A cross-protocol approach to detect and mitigate ARP-spoofing attacks in smart home networks. Journal of Network and Computer Applications **246**, 104396 (2026) https://doi.org/10.1016/j.jnca.2025.104396

[55] Karmous, N., Ben Dhiab, Y., Ould-Elhassen Aoueileyine, M., Youssef, N., Bouallègue, R., Yazidi, A.: Deep learning approaches for protecting IoT devices in smart homes from MitM attacks. Frontiers in Computer Science **6**, 1477501 (2024) https://doi.org/10.3389/fcomp.2024.1477501

[56] Yang, Y.-M., Chang, K.-C., Luo, J.-N.: A lightweight training approach for MITM detection in IoT networks: Time-window selection and generalization. Applied Sciences **15**(22), 12147 (2025) https://doi.org/10.3390/app152212147

[57] Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of

the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM, San Francisco, CA, USA (2016). https://doi.org/10.1145/2939672.2939785

[58] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.-Y.: LightGBM: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3146–3154. Curran Associates, Inc., Red Hook, NY (2017)

[59] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1321–1330 (2017)

[60] Cameron, A.C., Gelbach, J.B., Miller, D.L.: Bootstrap-based improvements for inference with clustered errors. Review of Economics and Statistics **90**(3), 414–427 (2008) https://doi.org/10.1162/rest.90.3.414

[61] Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall, New York (1993)

[62] Wolpert, D.H.: Stacked generalization. Neural Networks **5**(2), 241–259 (1992) https://doi.org/10.1016/S0893-6080(05)80023-1

[63] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**(2), 65–70 (1979)

[64] Field, C.A., Welsh, A.H.: Bootstrapping clustered data. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **69**(3), 369–390 (2007) https://doi.org/10.1111/j.1467-9868.2007.00593.x

[65] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the Mirai botnet. In: 26th USENIX Security Symposium (USENIX Security 17), pp. 1093–1110. USENIX Association, Vancouver, BC (2017)