future internet

MDPI

Article

A Scientific Integrity Framework for Open-Set IoT Intrusion Detection with Device-Disjoint Splits

Chekwas Ifeanyi Chikezie [1,*], Abraham Usman Usman [1], Michael David [1], Sulieman Zubair [1], Henry Ohiani Ohize [2] and Joseph Ojeniyi [3]

[1] Department of Telecommunications Engineering, Federal University of Technology, Minna 920101, Niger State, Nigeria; usman.abraham@futminna.edu.ng (A.U.U.); mikeforheaven@futminna.edu.ng (M.D.); zubairman@futminna.edu.ng (S.Z.)
[2] Department of Computer Engineering, Confluence University of Science and Technology, Osara 264103, Kogi State, Nigeria; ohizeho@custech.edu.ng
[3] Department of Cyber Security Science, Federal University of Technology, Minna 920101, Niger State, Nigeria; ojeniyija@futminna.edu.ng
* Correspondence: chekwas.pg2010285@st.futminna.edu.ng

Abstract

Machine-learning-based intrusion detection for Internet of Things systems has often been evaluated through model-centered pipelines that use weakly governed partitioning, limited leakage auditing, and closed-set assumptions. Consequently, reported performance could reflect data-handling artifacts rather than reliable security intelligence. This paper introduces a scientific integrity framework that treats preprocessing as a primary research object for open-set Internet of Things intrusion detection. The framework integrated device-disjoint split governance, feasibility-aware zero-day isolation, quantified leakage control, train-only preprocessing, shared-safe feature selection, diagnostic-harness verification, baseline split comparison, and auditable artifact generation. Applied to the CICIoT-DIAD 2024 corpus with Institute of Electrical and Electronics Engineers Organizationally Unique Identifier-based vendor enrichment, the protocol locked 28 canonical classes, eight semantic attack families, and five policy labels before constructing a device-disjoint, vendor-aware grouped split. When strict device-level zero-day holdout was infeasible, the framework activated an audited row-level fallback that preserved contamination-free holdout isolation without claiming strict device-novel zero-day evaluation. On 35,672,407 flows from 180 files, the accepted run achieved zero device overlap, zero flow-signature Jaccard leakage risk, 100 percent zero-day purity, a Feature Distribution Stability Score of 0.00518, a Device-Feature Dependency Index of 0.00000, an Attack Invariance Score of 0.92964, and an Attack Semantic Consistency Score of 0.90714. The diagnostic harness produced zero hard failures and zero warnings, while baseline comparison showed stronger preprocessing integrity than random stratified and simple device-disjoint splitting. This study did not claim downstream classifier superiority; rather, it established an auditable preprocessing substrate for later classifier-level experiments.

Keywords: Internet of Things security; information forensics; open-set recognition; dataset preprocessing; leakage control; reproducible machine learning

Academic Editor: Gianluigi Ferrari

Received: 20 April 2026
Revised: 11 May 2026
Accepted: 22 May 2026
Published: 27 May 2026

1. Introduction

The rapid growth of the Internet of Things (IoT) has increased the heterogeneity, scale, and vulnerability of cyber-physical communication environments. One important enabler

of this expansion is enhanced machine-type communication (eMTC), which supports the wide-area connectivity of large populations of low-power IoT devices [1]. In such environments, an Intrusion Detection System (IDS) is expected not only to recognize known malicious behavior, but also to retain diagnostic reliability when confronted with new devices, new traffic patterns, and previously unseen attacks. Recent work on IoT device identification and attack detection has reported strong classification performance using deep learning, hybrid Convolutional Neural Network (CNN) models, Explainable Artificial Intelligence (XAI), autoencoder-based models, and lightweight integrated frameworks [2–6]. Even so, most published studies still treat preprocessing as a brief preparatory stage rather than as a first-class scientific protocol. This paper challenges that practice by arguing that trustworthy IoT security evaluation depends as much on governed preprocessing as on downstream model design. That omission has direct scientific consequences. When training, validation, and test data are partitioned without strict device awareness, device signatures can leak across splits and inflate downstream performance. When feature selection is performed before split finalization, predictor rankings can inherit information from evaluation partitions. When open-set assumptions are ignored, high closed-set accuracy can conceal fragile behavior on genuinely unseen threats. Prior open-set intrusion recognition research has already shown that closed-set evaluation can overstate operational capability when unknown attack classes appear at inference time [7]. The central premise of this manuscript is therefore simple: if the data substrate is not governed rigorously, then the reported behavior of the detector cannot be interpreted rigorously either.

Against that background, this study proposes a governed preprocessing framework in which device-disjoint split construction, holdout feasibility analysis, flow-signature leakage auditing, shared-safe feature selection, and hard scientific gates are treated as primary research objects rather than hidden implementation details. The framework is instantiated on the Canadian Institute for Cybersecurity-IoT Device Identification and Anomaly Detection (CICIoT-DIAD) 2024 dataset, enriched with Organizationally Unique Identifier (OUI)-based vendor metadata, and evaluated on a completed accepted run containing 35,672,407 records distributed across 180 files [3]. The novelty of the work lies not in presenting another classifier, but in formalizing an evidence-bearing preprocessing protocol that can support later open-set, multi-head IoT learning under materially stronger integrity conditions.

The contribution is therefore methodological rather than classifier-centric. Instead of claiming a new detection architecture, the study establishes a reproducible foundation for later multi-head learning by ensuring that downstream experiments inherit trustworthy partitions, transparent metric gates, and verifiable integrity artifacts. In practical terms, this means that the accepted decision can be reconstructed from emitted protocol files, audit summaries, metric reports, and proof artifacts rather than from narrative description alone. This framing aligns with broader calls for reproducible and auditable security machine learning, including recent work that formalizes benchmarking and reproducibility checks for threat detection pipelines [8].

Before introducing the formal methodology, it is useful to state this manuscript's principal contributions explicitly. The following items summarize the technical claims supported by the framework design and by the accepted run:

i. An end-to-end scientific integrity framework was developed to treat preprocessing as an auditable research object for open-set Internet of Things intrusion-detection evaluation.
ii. A taxonomy-locking procedure was implemented to fix 28 canonical classes, eight semantic attack families, and five policy labels before split construction, feature selection, and audit reporting.

iii. A device-disjoint, vendor-aware split-governance protocol was introduced with feasibility-aware zero-day routing and an explicitly reported row-level fallback when strict device-level holdout was infeasible.
iv. A quantified leakage-control layer was developed using device-overlap auditing, flow-signature Jaccard analysis, device-feature dependency control, fingerprint-leakage scoring, train-only preprocessing, and hard scientific gates.
v. A shared-safe feature-selection procedure retained 12 compact protocol-level predictors as an integrity-approved representation, while avoiding any claim that the subset was universally optimal for downstream classifier performance.
vi. A diagnostic and reproducibility layer was introduced through artifact verification, baseline split-protocol comparison, metric reports, retained-feature manifests, diagnostic contracts, and figure registries, thereby making the accepted preprocessing decision inspectable and reusable for future classifier-level experiments.

2. Related Work

Recent Internet of Things security research has achieved strong progress in device identification, anomaly detection, and attack classification, yet a critical reading of the field reveals a persistent architecture-performance bias. Much of the literature has become highly effective at designing powerful predictive models, but relatively less attention has been given to preprocessing governance, leakage auditing, and the scientific validity of evaluation protocols. This imbalance is especially consequential in Internet of Things corpora, where device-level and temporal correlation can cause conventional random splitting or standard cross-validation to overestimate generalization by allowing behaviorally similar traffic from the same physical device to appear across multiple partitions. For that reason, the present review does not treat prior studies as isolated reports of model accuracy. Instead, it examines them as parts of a broader methodological landscape and asks whether they establish trustworthy experimental conditions in addition to strong predictive performance.

A first major strand of the literature is defined by model-centric architectural innovation. Rabbani et al. [3] proposed a lightweight integrated framework for simultaneous device identification and anomaly detection using packet-based and flow-based features, thereby demonstrating the value of joint behavioral descriptors for device and threat analysis. Building on this direction, Jain et al. [5] introduced a hybrid Convolutional Neural Network and Extreme Gradient Boosting architecture with SHapley Additive exPlanations-based interpretability, reporting strong performance on the CIC IoT2024-DIAD dataset. Hossain [6] likewise evaluated a one-dimensional Convolutional Neural Network, Long Short-Term Memory, a Recurrent Neural Network, and hybrid variants for scalable Internet of Things intrusion detection, again emphasizing downstream detection performance. Collectively, these studies establish strong baselines for predictive modeling and demonstrate that the field has matured considerably in terms of classifier design. However, their common limitation is that preprocessing remains subordinate to the model itself. Device-disjoint partitioning, leakage auditing, and open-set preprocessing realism are not treated as primary scientific contributions. As a result, high reported performance may still be partially supported by favorable data-handling conditions rather than by robust security intelligence.

A second line of work begins to engage more directly with the credibility of data handling. AlFuraih et al. [4] compared leakage-prone and leakage-aware Random Forest ranking strategies on the CIC IoT-DIAD 2024 dataset and showed that train-only ranking yields more defensible multi-class results. This was an important methodological advance because it explicitly recognized that feature selection can inherit information from

validation or test data when performed before strict partition control. In other words, the study moved beyond pure model competition and toward preprocessing awareness. Even so, the surrounding pipeline remained largely conventional, relying on chunk-wise standardization, top-k filtering, and standard split logic. The unresolved gap is therefore not the absence of leakage awareness, but the absence of a broader protocol that integrates leakage-aware feature selection with device-disjoint governance, feasibility-aware open-set holdout, and auditable acceptance logic. Without that broader integration, the literature still lacks a complete defense against evaluation bias in Internet of Things security experiments.

A third strand of the literature approaches the problem through reproducibility and benchmarking discipline. Shaikhanova et al. [8] proposed a reproducible machine-learning framework for Internet of Things threat detection and benchmarking, emphasizing transparent configuration, timing, and benchmarking outputs. Ahmed et al. [9] similarly argued for leakage-free experimental design in resource-efficient Distributed Denial of Service studies, showing that protocol construction itself can materially affect the credibility of reported results. These works are important because they shift attention from raw predictive performance to the conditions under which performance is measured. However, their central concern remains benchmarking discipline rather than preprocessing itself. In effect, they strengthen transparency around experiments, but they do not yet formalize preprocessing as the core scientific object. The gap that remains is that preprocessing is still treated as a support layer for evaluation rather than as an evidence-bearing protocol whose own integrity must be demonstrated and audited.

The need for that shift becomes even more pressing when the literature is viewed through the lens of open-set realism. Cruz et al. [10] showed that closed-set intrusion recognition can overstate operational readiness when previously unseen classes appear at inference time and therefore argued for explicit open-set treatment in fine-grained attack categorization. This contribution is foundational because it reframed intrusion detection as a task that must distinguish not only among known classes, but also between known and unknown behavior. That insight remains highly relevant to modern Internet of Things corpora, where previously unseen attack manifestations and device behaviors are operationally plausible. At the same time, practical open-set deployment on large-scale Internet of Things datasets introduces additional challenges, including threshold calibration, heterogeneous class density, and the possibility that ideal device-level holdout may be infeasible for certain target classes. The literature therefore provides the conceptual rationale for open-set evaluation, but it still lacks a preprocessing protocol that transparently manages holdout feasibility while preserving zero contamination through audited fallback logic. In operational terms, this gap means that a model reported as highly accurate in the laboratory may fail when deployed on previously unseen devices, vendors, or traffic regimes, even when those devices belong to familiar functional classes.

A related body of work extends the discussion into heterogeneous, decentralized, and scalable Internet of Things environments. Tajgardan et al. [11] investigated unsupervised federated learning for anomaly detection in heterogeneous Internet of Things settings and emphasized shared-feature reasoning under feature heterogeneity. Moghaddam et al. [12] evaluated an Evo-Transformer-LSTM model across several Canadian Institute for Cybersecurity benchmarks, reinforcing the field's continuing preference for high-capacity architectures and cross-dataset validation. Mgungile and Tonkal [13], together with Fraihat et al. [14], focused on scalable analytics and feature optimization for Internet of Things and Industrial Internet of Things detection tasks, thereby addressing efficiency and large-scale deployment concerns. These studies are valuable because they extend the field beyond single-model experimentation and reflect the real-world demand for scalable and adaptive security intelligence. Nevertheless, their preprocessing pipelines remain comparatively

standard. The specific gap they leave unresolved is that scaling, federation, and feature optimization are not accompanied by a governed protocol that explicitly audits device confounding, open-set routing integrity, and leakage-safe partition construction.

Taken together, the literature reveals a consistent pattern. Prior studies have made substantial progress in classifier design, interpretability, scalability, feature ranking, and reproducibility-oriented benchmarking. What remains missing is a unified preprocessing framework that integrates these concerns into a single scientific protocol. More specifically, the current body of work still lacks a preprocessing system that simultaneously enforces device-disjoint partitioning, feasibility-aware open-set routing, quantitative leakage auditing, train-only transformation, and explicit quantitative decision gates backed by reusable artifacts. This collective gap defines the central novelty of the present study. Rather than proposing another model-centric detection pipeline, this work formalizes preprocessing itself as a governed, auditable, and evidence-bearing scientific contribution.

Table 1 summarizes representative prior work and situates the proposed framework within this methodological landscape. By presenting the reviewed studies side by side, the table makes the collective research gap explicit and prepares the ground for the scientific integrity framework developed in the next section.

Table 1. Prior work and its relationship to the proposed framework.

Study	Core Mechanism	Leakage Handling	Open-Set Handling	Gap Relative to This Work
Rabbani et al. [3]	Packet & flow-based hybrid features	Implicit (Standard dataset splits)	No explicit unknown-class protocol	Focuses on behavioral modeling rather than auditable preprocessing governance.
Jain et al. [5]	CNN-XGBoost with SHapley Additive exPlanations (SHAP) interpretability	No explicit device-disjoint control	Closed-set evaluation	Model-centric; lacks evidence-bearing preprocessing artifacts or split auditing.
Hossain [6]	Deep architectures (1D-CNN, LSTM, RNN)	Conventional preprocessing pipelines	Closed-set evaluation	Prioritizes predictive capacity over governed and reproducible split construction.
AlFuraih et al. [4]	Leakage-aware Random Forest ranking	Explicit train-only feature ranking	Closed-set evaluation	Addresses ranking leakage but lacks device-disjoint governance and holdout logic.
Shaikhanova et al. [8]	Transparent benchmarking & timing	Systematic reproducibility checks	Not open-set specific	Functions as a benchmarking utility rather than a governed data-partitioning protocol.
Cruz et al. [10]	Fine-grained open-set recognition	N/A (general network traffic)	Explicit open-set categorization	Conceptual foundation for open-set logic, but lacks a modern IoT-specific preprocessing pipeline.
Proposed Work	**Four-stage Scientific Integrity Protocol**	**Quantitatively audited leakage gates**	**Feasibility-aware routing and holdout**	**Novelty:** Formalizes preprocessing as a governed, evidence-bearing scientific object.

As Table 1 makes clear, the gap is not the absence of predictive models. The more consequential gap is the absence of a single preprocessing protocol that simultaneously enforces device-disjoint partitioning, feasibility-aware open-set routing, quantitative leakage auditing, train-only transformation, and explicit decision gates. The present study addresses that gap by treating these elements as one connected integrity framework rather than as isolated implementation choices.

3. Dataset, Taxonomy, and Problem Formulation

3.1. Dataset Composition and OUI Enrichment

The governed pipeline operates on the CICIoT-DIAD 2024 raw Parquet directory together with an Institute of Electrical and Electronics Engineers (IEEE) Organizationally Unique Identifier (OUI) vendor map. The scan-level artifact summary records 180 Parquet files, 35,672,407 rows from metadata, 163 profiled devices, and 253,048 rows for which device identity is unknown at the grouped split stage. The OUI lookup enriches device addresses with vendor identity, thereby enabling vendor-aware balancing, vendor-overlap risk estimation, and more explicit confounding diagnostics. This enrichment is useful, but it is not treated as infallible. Randomized, missing, or ambiguously registered prefixes remain unresolved, and those cases are carried forward transparently rather than silently forced into unreliable vendor assignments.

Table 2 provides a compact summary of the accepted run's dataset scale and protocol configuration. It serves as the reference point for the split-governance and open-set analyses developed later in the manuscript.

Table 2. Accepted-run dataset and protocol summary.

Item	Value	Interpretation
Raw Parquet files	180	Dataset card scan summary
Rows from metadata	35,672,407	Dataset card scan summary
Profiled devices	163	Grouped split protocol metadata
Unknown-device rows	253,048	Rows emitted after device-addressable grouping
Target split ratios	0.70/0.15/0.15	Train/validation/test
Actual grouped ratios	0.711945/0.144559/0.143496	Grouped assignment stage
Effective zero-day mode	Row_Level_Fallback_Required	Final protocol outcome

3.2. Canonical Class, Family, and Policy Taxonomy

A preprocessing framework can only be audited reproducibly if the label system is fixed before any split or transform is performed. For that reason, the pipeline locks 28 canonical classes, maps them into 8 semantic families, and associates each class with 1 of 5 operational policy labels. The canonical classes follow normalized corpus attack names, the family groupings aggregate semantically related attack types for stability analysis, and the policy labels capture the downstream response semantics that the later multi-head pipeline is intended to learn. Table 3 provides the full class–family–policy mapping used throughout the accepted run. Its role is foundational because it fixes the semantic vocabulary that every subsequent split, audit, and model must inherit.

Table 4 complements the taxonomy by summarizing the 136 raw features present in every Parquet file. For readability, the table groups the inventory by semantic role and representative fields, while the complete column manifest is preserved in the dataset card and associated feature artifacts. Of these 136 inputs, 43 are explicit device or network identifiers, including Media Access Control (MAC) addresses, OUI fields, Internet Protocol (IP) addresses, ports, and rolling aggregates derived from them. This raw inventory is one reason the framework places such emphasis on Device-Feature Dependency Index

(DFDI) and Fingerprint Leakage Score (FLS) gates, because any retained feature drawn from this identifier-heavy category would create an immediate leakage path under device-disjoint evaluation.

Table 3. Canonical 28-class taxonomy and mappings.

S/N	Class	Family	Policy	S/N	Class	Family	Policy
1	benign_traffic	Benign	Allow	**15**	vulnerability_scan	Reconnaissance	Block
2	backdoor_malware	Mirai/Malware	Quarantine	**16**	dos_http_flood	DoS Attacks	Throttle
3	browser_hijacking	Web-Based	Quarantine	**17**	dos_syn_flood	DoS Attacks	Throttle
4	command_injection	Web-Based	Block	**18**	dos_tcp_flood	DoS Attacks	Block
5	sql_injection	Web-Based	Block	**19**	dos_udp_flood	DoS Attacks	Block
6	uploading_attack	Web-Based	Block	**20**	ddos_http_flood	DDoS Attacks	Block
7	xss	Web-Based	Block	**21**	ddos_slow_loris	DDoS Attacks	Block
8	dictionary_brute_force	Brute-Force	Block	**22**	ddos_tcp_flood	DDoS Attacks	Block
9	dns_spoofing	Spoofing	Block	**23**	ddos_udp_flood	DDoS Attacks	Block
10	mitm_arpspoofing	Spoofing	Block	**24**	ddos_udp_fragmentation	DDoS Attacks	Block
11	recon_host_discovery	Reconnaissance	Alert	**25**	ddos_ack_fragmentation	DDoS Attacks	Block
12	recon_os_scan	Reconnaissance	Alert	**26**	ddos_icmp_fragmentation	DDoS Attacks	Block
13	recon_ping_sweep	Reconnaissance	Alert	**27**	ddos_synonymous_ip_flood	DDoS Attacks	Block
14	recon_port_scan	Reconnaissance	Block	**28**	mirai_greip_flood	Mirai/Malware	Quarantine

Table 4. Raw feature inventory of the CICIoT-DIAD 2024 corpus.

Category	Count	Representative Features (Selected)
Device/Network Identifiers	43	device_mac, src_mac, dst_mac, eth_src_oui, eth_dst_oui, src_ip, dst_ip, src_port, dst_port, all src_ip_mac_* and src_ip_* rolling aggregates
Protocol-Specific	22	handshake_ciphersuites, handshake_extensions_length, http_content_len, http_response_code, icmp_type, icmp_data_size, dns_query_type, l4_tcp, l4_udp, tcp_window_size
Flow/Stream Statistics	51	All stream_*, channel_*, stream_jitter_*, inter_arrival_time, payload_entropy, payload_length, jitter
Statistical Aggregates	14	min_p, max_p, med_p, q1_p, q3_p, iqr_p, var_p, sum_p, average_p
Other/Metadata & Label	6	label, ttl, user_agent, eth_size, most_freq_spot, time_since_previously_displayed_frame

Note: The asterisk (*) denotes wildcard notation and refers to all feature names sharing the indicated prefix. For example, src_ip_mac_* represents all variables beginning with src_ip_mac_, while stream_* represents all variables beginning with stream.

3.3. Formal Problem Statement and Notation

To formalize the preprocessing problem, let the raw corpus be represented as a device- and vendor-aware labeled dataset. Equation (1) expresses this corpus as

$$\mathcal{D} = \left\{ (x_i, y_i, d_i, v_i) \right\}_{i=1}^{N} \tag{1}$$

where x_i denotes the feature vector of the i-th row, y_i denotes its canonical attack label, d_i denotes its device identity, v_i denotes its vendor identity induced from the OUI prefix (when available), and N denotes the total number of rows in the corpus. This notation provides a consistent mathematical reference for the remainder of the manuscript and makes explicit that the preprocessing problem is not merely label-aware, but also device-aware and vendor-aware.

Once the corpus has been defined, the first operational requirement is split proportionality. If r_s denotes the observed ratio of rows assigned to split s, then the observed split ratio is given by Equation (2):

$$r_s = \frac{\text{rows assigned to split } s}{\text{total rows}} \tag{2}$$

where r_s represents the realized proportion of data allocated to split s, and s denotes one of the protocol partitions, namely, training, validation, or test. This quantity is important because it forms the basis for the subsequent drift analysis.

Because the target protocol specifies 70%, 15%, and 15% for training, validation, and test, respectively, the absolute percentage-point drift is measured by Equation (3):

$$\delta_s = \mid r_s - \tau_s \mid \times 100 \tag{3}$$

where δ_s denotes the percentage-point drift for split s, r_s is the observed split ratio from Equation (2), and τ_s is the target ratio for that split. This equation makes it possible to quantify how closely the realized allocation follows the intended protocol.

After establishing split proportionality, the framework turns to leakage control. To support this stage, Equation (4) defines a deterministic quantized flow signature:

$$\text{sig}_i = \text{SHA256}(q(x_{i1}) \parallel q(x_{i2}) \parallel \cdots \parallel q(x_{ik})) \tag{4}$$

where sig_i is the signature of the i-th flow, $x_{i1}, x_{i2}, \ldots, x_{ik}$ are the selected numeric attributes used in the signature, $\parallel$ denotes concatenation, and $\text{SHA256}(\cdot)$ denotes the hash function used to produce the final digest. The quantization operator $q(\cdot)$ maps each selected attribute to a deterministic finite-precision representation before hashing. In the accepted run, continuous values were rounded to six decimal places (as specified in the audit configuration), missing values were mapped to a reserved token, and categorical numeric indicators were preserved as integer codes. This ensured that flow signatures were reproducible across repeated executions and robust to insignificant floating-point representation differences.

Once these signatures have been defined, overlap between any two split-signature sets A and B can be summarized by the Jaccard coefficient in Equation (5):

$$J(A, B) = \frac{\mid S_A \cap S_B \mid}{\mid S_A \cup S_B \mid} \tag{5}$$

where $J(A, B)$ is the Jaccard overlap between the two splits, S_A is the set of flow signatures in split A, S_B is the set of flow signatures in split B, $\cap$ denotes set intersection, and $\cup$ denotes set union. In this framework, the coefficient is interpreted as a leakage-risk proxy and feeds directly into the flow-signature Jaccard gate.

Beyond row-level overlap, the framework also constrains device-identifying content at the feature level. Equation (6) defines the normalized mutual information between a feature and device identity:

$$\text{NMI}(f, D) = \frac{I(f; D)}{\min(H(f), H(D))} \tag{6}$$

where $\text{NMI}(f, D)$ is the normalized mutual information between feature f and device label D, $I(f; D)$ denotes their mutual information, and $H(f)$ and $H(D)$ denote the entropies of the feature and the device label, respectively. The denominator $\min(H(f), H(D))$ was used as a conservative normalization because it scaled mutual information by the smaller available entropy source. Under this formulation, a feature received a high dependency score when it explained a large fraction of either its own entropy or the device-label entropy. This made the dependency gate stricter for low-entropy features that could behave as compact device identifiers.

Using this primitive, the Device-Feature Dependency Index was defined as the maximum normalized device dependency among the retained shared-safe features as defined in Equation (7):

$$\mathrm{DFDI} = \max_{f \in \mathcal{F}_{ss}} \mathrm{NMI}(f, D) \tag{7}$$

where DFDI is the Device-Feature Dependency Index, F_{ss} is the retained shared-safe feature subset, and the max operator selects the highest normalized device dependency among all retained features. This formulation produces a conservative worst-case measure of residual device-dependency leakage.

Feature-distribution stability must also be preserved after split construction. Equation (8) defines the Feature Distribution Stability Score as

$$\mathrm{FDSS} = \frac{1}{|\mathcal{F}_{ss}|} \sum_{f \in \mathcal{F}_{ss}} \left(\mathrm{JS}\left(P_f^{\mathrm{train}}, P_f^{\mathrm{eval}}\right) + \mathrm{KS}\left(F_f^{\mathrm{train}}, F_f^{\mathrm{eval}}\right) \right) \tag{8}$$

where FDSS is the Feature Distribution Stability Score, F_{ss} is the shared-safe feature subset, P_f^{train} and P_f^{eval} are the empirical distributions of feature f in the training and evaluation partitions, F_f^{train} and F_f^{eval} are the corresponding cumulative distribution functions, $\mathrm{JS}(\cdot,\cdot)$ denotes Jensen–Shannon divergence, and $\mathrm{KS}(\cdot,\cdot)$ denotes Kolmogorov–Smirnov distance. This score aggregates distributional divergence across retained features and serves as the principal drift gate in the protocol.

Shared-safe feature selection is then expressed conceptually through Equation (9):

$$\mathrm{Score}(f) = U(f) - (w_1 \cdot M(f) + w_2 \cdot S(f) + w_3 \cdot R(f) + w_4 \cdot P(f)) \tag{9}$$

where $\mathrm{Score}(f)$ is the composite score assigned to feature f, $U(f)$ is its semantic utility with respect to attack labels, $M(f)$ is the device-dependency penalty, $S(f)$ is the drift-instability penalty, $R(f)$ is the missingness or coercion burden, and $P(f)$ is the explicit identifier-leakage penalty. Before aggregation, all score components were normalized to a common scale. In the accepted run, the penalty weights were fixed as $w_1 = w_2 = w_3 = w_4 = 1$, so that device dependency, distributional drift, missingness/coercion burden, and explicit identifier leakage contributed symmetrically to the penalty term. This equal-weight setting was selected a priori and was not tuned on the validation or test partitions, thereby avoiding an additional optimization hyperparameter in the preprocessing protocol. The score was used only as a ranking mechanism; final retention still required each feature to satisfy the hard guards on device dependency, drift stability, explicit identifier leakage, missingness, and transform reliability. Therefore, the final retained feature set was not determined by Equation (9) alone, and a high utility score could not override leakage or stability constraints. This design improved reproducibility while reducing sensitivity to arbitrary penalty-weight choices.

Open-set correctness is summarized through zero-day purity. If H denotes all designated holdout rows routed to the test set and C denotes any contamination of those rows into training or validation, then Equation (10) defines purity as

$$\mathrm{Purity} = \left(1 - \frac{|H \cap C|}{|H|}\right) \times 100\% \tag{10}$$

where Purity is the zero-day purity percentage, H is the set of holdout rows, C is the contamination set, and $|\cdot|$ denotes set cardinality. This metric measures the fraction of holdout rows preserved exclusively in the test split and directly supports the zero-day purity gate.

The framework then evaluates family-level stability through the Attack Invariance Score, defined in Equation (11):

$$\mathrm{AIS} = 1 - \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathrm{TV}\left(P_{\mathrm{fam}}^{\mathrm{train}}, P_{\mathrm{fam}}^{s}\right) \tag{11}$$

where AIS is the Attack Invariance Score, S is the set of evaluation splits considered in the comparison, $\mathrm{TV}(\cdot,\cdot)$ denotes total-variation distance, $P_{\mathrm{fam}}^{\mathrm{train}}$ is the family-level class distribution in the training split, and P_{fam}^{s} is the corresponding family-level distribution in evaluation split s. This score measures how well broad attack-family structure is preserved across partitions.

Finally, the framework measures whether the finer semantic arrangement of classes within each family survives split governance and feature filtering. Equation (12) defines the Attack Semantic Consistency Score as

$$\mathrm{ASCS} = 1 - \frac{1}{|\mathcal{F}| \cdot |\mathcal{S}|} \sum_{f \in \mathcal{F}} \sum_{s \in \mathcal{S}} \mathrm{JS}(P_{\mathrm{class}|f}^{\mathrm{train}}, P_{\mathrm{class}|f}^{s}) \tag{12}$$

where ASCS is the Attack Semantic Consistency Score, F is the set of semantic attack families, S is the set of evaluation splits, $\mathrm{JS}(\cdot,\cdot)$ denotes Jensen–Shannon divergence, $P_{\mathrm{class}|f}^{\mathrm{train}}$ is the class distribution conditioned on family f in the training split, and $P_{\mathrm{class}|f}^{s}$ is the corresponding conditional class distribution in evaluation split s. This final metric captures whether semantic structure survives not only partitioning, but also the aggressive leakage-suppression logic imposed by the shared-safe feature-selection stage.

4. Proposed Scientific Integrity Framework

4.1. Design Principles

The framework is built on five design principles that remain connected throughout the pipeline. First, taxonomy locking precedes split construction so that alias normalization cannot drift across stages. Second, split governance is device-disjoint for device-addressable rows because device signatures are a primary confounding source in IoT traffic. Third, zero-day handling is explicit rather than implicit: holdout classes are defined before routing and their final allocation is audited after writing. Fourth, all transformations that estimate statistics such as scaling, clipping, and imputation are fitted only on training data. Fifth, the pipeline ends with hard scientific gates rather than purely visual dashboard interpretation. Together, these principles create a direct chain of evidence from raw data to final scientific decision and explain why the framework should be understood as a scientific integrity protocol rather than as a preprocessing convenience.

4.2. Full System Architectural Framework

Figure 1 depicts the full system architectural framework of the proposed preprocessing protocol. The figure presents the end-to-end governed flow of the system, beginning with raw ingestion and taxonomy locking, continuing through grouped split governance, holdout feasibility analysis, shared-safe feature selection, train-only transformation, and audit emission, and ending with the final scientific decision. Each block corresponds to a deterministic component of the implemented pipeline, so the architecture serves not merely as a conceptual overview but as a map of the evidence-producing stages of the framework.

4.3. Canonical Taxonomy Lock and Label Normalization

Before any grouped split is constructed, the label space is normalized to the canonical 28-class vocabulary shown earlier in Table 3. This stage removes ambiguity arising from raw file names, attack aliases, case variation, and punctuation differences. Algorithm 1 formalizes taxonomy locking and label normalization. Its role is foundational because, once the canonical mapping is fixed, all downstream summaries, family assignments, and

policy labels become deterministic. In other words, this is the first irreversible stage of the protocol and the first point at which reproducibility is enforced structurally rather than narratively.

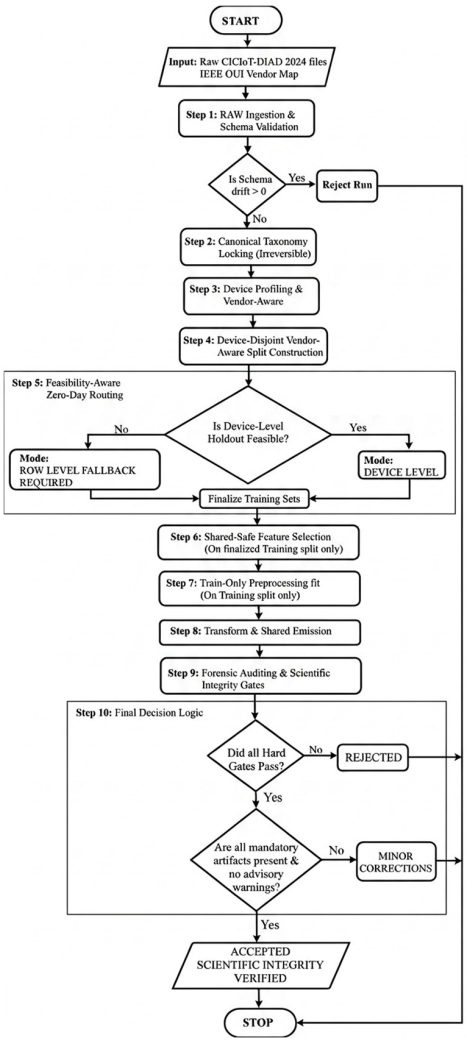

Figure 1. Full system architectural framework flowchart.

Algorithm 1: Canonical taxonomy locking and label normalization

Input: raw labels L_{raw}, canonical class list $\mathcal{C}$, family map $\mathcal{G}$, policy map $\mathcal{P}$
Output: canonical labels y, family labels g, policy labels p
for each raw label ℓ in L_{raw} do
    normalize case, punctuation, whitespace, and delimiter variants
    if ℓ matches a canonical class in $\mathcal{C}$ then
        $y \leftarrow$ canonical form of ℓ
    else if ℓ matches a registered alias then
        $y \leftarrow$ canonical target of the alias
    else
        mark ℓ as unresolved and raise integrity warning
    end if
    $g \leftarrow \mathcal{G}(y)$ ▷ semantic family assignment
    $p \leftarrow \mathcal{P}(y)$ ▷ downstream policy assignment
end for
emit label, family, and policy dictionaries as immutable artifacts

4.4. Device-Disjoint Vendor-Aware Split Construction

Once taxonomy locking is complete, the protocol proceeds to grouped split construction. The purpose of this stage is to assign device-addressable rows to mutually disjoint training, validation, and test groups while simultaneously controlling ratio drift and vendor imbalance. Algorithm 2 is central because it is the mechanism through which the framework controls device-level confounding. The split is not merely a random partition of rows. It is a governed assignment problem in which device identity and vendor composition are treated as integrity constraints.

Algorithm 2: Device-disjoint vendor-aware grouped split construction

Input: device-grouped rows $\mathcal{D}_g$, target ratios τ, vendor map $v(d)$
Output: grouped train, validation, and test device sets
1: group rows by device identifier d
2: compute device-level row counts and vendor affiliations
3: initialize empty split sets S_{tr}, S_{va}, S_{te}
4: sort devices by descending row count and balancing priority
5: for each device group d do
6: evaluate provisional assignment of d to each split $s \in \{\text{tr}, \text{va}, \text{te}\}$
7: compute ratio drift penalty under Equation (3)
8: compute vendor-balance penalty using current vendor composition
9: assign d to the split with minimum total penalty
10: end for
11: iteratively refine assignments if any split remains outside tolerance
12: verify that $S_{\text{tr}} \cap S_{\text{va}} = \emptyset$, $S_{\text{tr}} \cap S_{\text{te}} = \emptyset$, and $S_{\text{va}} \cap S_{\text{te}} = \emptyset$
13: emit split_protocol.json and device_split_assignment.csv

4.5. Feasibility-Aware Zero-Day Routing with Audited Row-Level Fallback

The framework treated open-set handling as a constrained routing problem rather than as a post hoc labeling choice. The requested mode was device-level holdout for selected attack classes, but that request was accepted only when class completeness and minimum training support remained feasible after grouped assignment. In this study, device-level zero-day holdout was considered feasible only when two conditions were simultaneously satisfied. First, all rows belonging to the designated holdout classes had to be routable to the test partition without any contamination into the training or validation partitions. Second, after removing the holdout-class device groups, the remaining training partition had to preserve non-zero supervised support for every non-holdout canonical class. This minimum support condition was deliberately conservative because the present work evaluated preprocessing integrity rather than downstream classifier optimization.

In the accepted run, the requested zero-day mode was device-level holdout, with backdoor_malware and xss designated as holdout classes. However, class-complete device-level holdout was not feasible without weakening the supervised support required for known-class learning. The protocol therefore activated ROW_LEVEL_FALLBACK_REQUIRED. This fallback did not claim strict device-novel zero-day evaluation. Instead, it preserved zero contamination by emitting all designated holdout rows exclusively to the test partition while explicitly recording the weaker but reproducible open-set mode. This distinction was important because the framework was designed to report the strongest feasible open-set condition, not to overstate the experimental setting.

Algorithm 3 formalizes the feasibility-aware routing procedure. Its main purpose was to make the open-set decision auditable: the system first attempted device-level holdout,

checked whether the resulting known-class training support remained valid, and then either accepted device-level routing or activated row-level fallback. In both cases, the final split proof recorded the effective mode, the holdout classes, the holdout-row counts, the contamination count, and the resulting zero-day purity.

Algorithm 3: Feasibility-aware zero-day routing with row-level fallback and purity auditing

Input: grouped splits S_{tr}, S_{va}, S_{te}, holdout class set $\mathcal{H} = \{\text{backdoor\_malware}, \text{xss}\}$
Output: effective open-set routing mode and holdout emission plan
request device-level holdout for all rows with labels in $\mathcal{H}$
estimate remaining supervised support in training after holdout removal
if class-complete device-level holdout preserves feasibility then
    effective_mode ←DEVICE_LEVEL
    route all holdout-class device groups to test
else
    effective_mode ←ROW_LEVEL_FALLBACK_REQUIRED
    keep grouped split assignments for known classes
    emit holdout-class rows only during final write stage into test
    enforce contamination count $C = 0$ for train and validation
end **if**
compute zero-day purity using Equation (10)
emit final_row_level_split_proof.json and validation audit figures

The fallback mechanism was therefore not a relaxation of contamination control. It was a transparent feasibility response to class-support constraints. The resulting open-set condition was weaker than strict device-level novelty, but it remained scientifically auditable because the effective mode, contamination count, and holdout purity were reported explicitly.

4.6. Shared-Safe Feature Selection

The full 136-feature inventory, summarized in Table 4, contained 43 explicit device or network identifier fields, including MAC addresses, OUI fields, Internet Protocol (IP) addresses, ports, and rolling aggregates derived from source and destination identity information. Because such fields could introduce direct or indirect device memorization under a device-disjoint evaluation protocol, the feature-selection stage was executed only after taxonomy locking, grouped split construction, and zero-day routing had been finalized. The purpose of this stage was not to maximize raw predictive capacity at any cost. Instead, the objective was to retain a compact set of predictors that preserved protocol-level security semantics while satisfying strict scientific-integrity constraints on identifier leakage, device dependency, distributional stability, missingness, coercion burden, and train-only transform reliability. Thus, the selector favored features that were interpretable from a network-protocol perspective and that remained stable across the governed train, validation, and test partitions.

Algorithm 4 formalizes the shared-safe feature-selection procedure. The algorithm first excluded direct identifiers and identifier-derived fields, then evaluated the remaining numeric and protocol-level candidate features using semantic utility, device-dependency penalty, distributional-drift burden, missingness/coercion burden, and explicit identifier-leakage penalty. Final retention was not determined by a single ranking score alone. A feature could be retained only if it also passed the hard guards on device dependency, identifier leakage, distributional stability, missingness, and transform reliability. The resulting

subset contained 12 retained shared-safe features. This reduction should be interpreted as a conservative integrity decision rather than as a claim that only 12 variables were universally sufficient for all downstream intrusion-detection models. The retained subset represented the smallest feature group that satisfied the framework's hard leakage and stability requirements while preserving protocol-level attack semantics. Larger leakage-safe feature sets may still be useful for downstream classifier optimization, but the present framework prioritized a defensible preprocessing substrate over maximal closed-set predictive capacity. Table 5 therefore reports the retained features as an integrity-approved representation rather than as a conventional feature-importance ranking.

Table 5. Retained shared-safe features and integrity diagnostics after leakage-aware feature selection.

Feature	Feature Group	Security Interpretation	Identifier Leakage Risk	Device NMI	FDSS Proxy	Retention Reason
handshake_ciphersuites	Protocol-specific	TLS handshake cipher-suite behavior	No explicit identifier field	0.0000	0.0000	Retained for protocol semantics and zero device dependency
handshake_extensions_length	Protocol-specific	TLS extension-vector length	No explicit identifier field	0.0000	0.0012	Stable across splits
ntp_interval	Protocol-specific/timing	NTP timing behavior	No explicit identifier field	0.0000	0.0012	Low drift and no identifier leakage
handshake_cipher_suites_length	Protocol-specific	Count-based TLS cipher-suite descriptor	No explicit identifier field	0.0000	0.0012	Complements TLS handshake characterization
handshake_sig_hash_alg_len	Protocol-specific	TLS signature/hash negotiation length	No explicit identifier field	0.0000	0.0015	Stable cryptographic-handshake descriptor
http_content_len	Protocol-specific	HTTP content-length behavior	No explicit identifier field	0.0000	0.0019	Stable application-layer traffic descriptor
icmp_data_size	Protocol-specific	ICMP payload-size behavior	No explicit identifier field	0.0000	0.0038	Supports ICMP and flood-related traffic characterization
http_response_code	Protocol-specific	HTTP response-status behavior	No explicit identifier field	0.0000	0.0038	Retained for application-layer response semantics
l4_udp	Protocol-specific	UDP traffic indicator	No explicit identifier field	0.0000	0.0047	Supports UDP-based attack-family representation
l4_tcp	Protocol-specific	TCP traffic indicator	No explicit identifier field	0.0000	0.0105	Supports TCP, HTTP, and SYN-family representation
icmp_checksum_status	Protocol-specific	ICMP checksum-integrity state	No explicit identifier field	0.0000	0.0155	Supports ICMP integrity-related traffic characterization
icmp_type	Protocol-specific	ICMP message-type behavior	No explicit identifier field	0.0000	0.0169	Supports ICMP attack-family representation

Note: Label mutual-information values were not reported in this table because the retained representation was not selected to maximize individual feature-label dependence. Instead, the shared-safe selector prioritized protocol-level semantic coverage subject to strict constraints on device dependency, explicit identifier leakage, distributional drift, missingness, and transform stability. Device NMI values were computed after excluding explicit device, vendor, MAC, IP, port, and rolling identifier-derived fields.

Algorithm 4: Shared-safe feature selection under semantic-utility and leakage constraints

Input: numeric feature set $\mathbb{F}$, label utility $\boldsymbol{U}(f)$, device dependency $\boldsymbol{M}(f)$, drift $\boldsymbol{S}(f)$, burden $\boldsymbol{R}(f)$
Output: shared-safe subset $\mathbb{F}_{ss}$
1: compute label utility $\boldsymbol{U}(f)$ for every feature f using normalized label mutual information
2: compute device dependency $\boldsymbol{M}(f)$ using Equation (6)
3: compute drift burden $\boldsymbol{S}(f)$ using Equation (8) primitives
4: compute missingness/coercion burden $\boldsymbol{R}(f)$
5: assign name-based identifier penalty $\boldsymbol{P}(f)$ for explicit device or vendor leakage cues
6: compute Score(f) using Equation (9)
7: remove features violating hard guards on device dependency or instability
8: rank remaining features by descending Score(f)
9: iteratively trim worst features until all scientific gates pass
10: return final shared-safe subset $\mathbb{F}_{ss}$ with $|\,\mathbb{F}_{ss}\,| = \mathbf{12}$

4.7. Train-Only Preprocessing and Transform Stability

All estimated transformations are fitted on the training split only. This includes scaling, clipping, and any imputation statistics, although the accepted run recorded zero-percent missingness across training, validation, and test. Restricting fit statistics to the training partition prevents subtle information transfer from evaluation partitions into the learned representation. The feature-forensics audit further confirmed zero missingness and zero coercion for every retained feature across all splits, which strengthens the argument that the accepted representation is not only leakage-safe but also numerically stable.

4.8. Diagnostic Harness, Scientific Integrity Gates, and Decision Logic

The framework concluded with a diagnostic harness and a set of hard scientific integrity gates. The diagnostic harness served as an auditable verification layer that consolidated the outputs of the preprocessing protocol into machine-readable checks. It did not evaluate downstream classifier performance. Instead, it verified whether the emitted preprocessing substrate satisfied the scientific conditions required for later model training and evaluation.

The harness checked artifact completeness, taxonomy consistency, split and fallback proof, retained-feature consistency, hard scientific gate outcomes, baseline-comparison validity, and manuscript-ready metric values. These checks were mapped to hard diagnostic contracts or advisory statuses so that the final decision could be reconstructed from evidence rather than from visual dashboard interpretation alone. In this sense, the diagnostic harness strengthened the reproducibility of the framework by ensuring that the accepted run was supported by explicit artifacts and not only by narrative reporting.

The hard scientific gates formed the quantitative acceptance layer of the framework. Each gate was defined by a status, severity level, measured value, threshold, optimization direction, and rationale. The quantitative gate layer operationalized the diagnostic logic by converting preprocessing-integrity requirements into pass/fail evidence. Table 6 provides the hard scientific gates used for final decision making, including their statuses, measured values, thresholds, optimization directions, and rationales.

Table 6. Hard scientific gates used for final decision making.

Gate	Status	Severity	Value	Threshold	Goal	Rationale
Device-disjoint MAC leakage	PASS	INFO	0.000000	0.0	MIN	Train/validation/test device sets remain disjoint.
Zero-day purity	PASS	INFO	100.000000	100.0	MAX	Holdout threat rows appear exclusively in test.
Flow-signature Jaccard risk	PASS	INFO	0.000000	0.0002	MIN	Flow overlap remains safely low.
Split-ratio drift	PASS	WARN	0.652849	2.0	MIN	Split-ratio deviation is advisory when within operational tolerance.
Quarantine rate	PASS	WARN	0.000000	1.0	MIN	Operational quarantine remained negligible; policy-mapped quarantine classes are excluded from this integrity-rate gate.
Feature Distribution Stability Score (FDSS)	PASS	INFO	0.005180	0.02	MIN	Processed feature drift remains controlled.
Device-Feature Dependency Index (DFDI)	PASS	INFO	0.000000	0.1	MIN	Device-identifying feature dependency is controlled.
Fingerprint Leakage Score (FLS)	PASS	INFO	0.000000	0.02	MIN	Anti-fingerprinting control remains acceptable.
Attack Invariance Score (AIS)	PASS	INFO	0.929636	0.8	MAX	Threat-family distribution stability is acceptable.
Attack Semantic Consistency Score (ASCS)	PASS	INFO	0.907144	0.8	MAX	Attack semantic consistency remains acceptable.

The hard gates in Table 6 were preprocessing-integrity gates rather than downstream classifier-performance thresholds. They evaluated split leakage, holdout purity, duplicate-flow risk, feature-distribution stability, device-feature dependency, fingerprint leakage, and semantic preservation. No gate was based on classifier accuracy, precision, recall, or macro-F1, because the present study evaluated the trustworthiness of the data substrate rather than the performance of a final detection model.

A run was marked as accepted only when every hard scientific gate passed and no unresolved advisory blocker remained. In the accepted run analyzed here, the final banner reported that all hard scientific gates passed and that no unresolved advisory blocker remained. This point was important because the framework was designed to produce an auditable preprocessing decision, not merely a dashboard display.

The diagnostic harness then provided a secondary verification layer for this decision. It checked whether the required artifacts, taxonomy mappings, split/fallback proof, retained-feature contract, scientific gate outputs, and baseline-comparison files were present and internally consistent. This additional verification step ensured that the accepted run could be inspected through machine-readable evidence and reproduced as a governed preprocessing substrate for later model-focused experiments.

5. Experimental Protocol

5.1. Environment, Inputs, and Outputs

The implementation was executed as a combined shared-dashboard preprocessing run over the raw Parquet source directory and the OUI enrichment file described earlier. The output structure included META, TABLES, REPORTS, FIGURES, SHARDS, LOGS, AUDITS, and MEMMAPS folders. These directories stored the protocol configuration, taxonomy

artifacts, split proofs, scientific flags, diagnostic tables, figure outputs, shard-level emissions, and audit logs needed to reconstruct the accepted run.

The primary preprocessing artifacts included the preprocessing summary, dataset card, taxonomy card, split protocol, train-only preprocessor bundle, final metric summary, final scientific flags, and final row-level split proof. This artifact structure was important because it converted the accepted run into an inspectable scientific record rather than a transient computation. The accepted decision could therefore be traced from the raw dataset scan through taxonomy locking, split construction, fallback routing, feature selection, gate evaluation, and final reporting.

In addition to the primary preprocessing outputs, a diagnostic harness was executed to verify the integrity of the accepted run. The harness served as an independent verification layer for the preprocessing substrate. It did not evaluate downstream classifier accuracy, precision, recall, macro-F1, or open-set AUROC. Instead, it verified artifact completeness, taxonomy consistency, split and fallback proof, scientific gate outcomes, retained-feature consistency, baseline-comparison validity, and manuscript-ready metric values. This additional verification layer strengthened reproducibility by ensuring that the accepted decision was supported by machine-readable contracts as well as by the original preprocessing outputs. Table 7 lists the core artifacts used to reconstruct the accepted run reported in this paper. In effect, these artifacts form the evidentiary backbone of the framework.

Table 7. Core artifacts emitted by the framework and used in this manuscript.

Artifact	Artifact Group	Role	Primary Contents
Preprocessing_Summary.md	Primary preprocessing report	Run-level textual summary	Overall status, final banner, split counts, gate values, and accepted-run interpretation
Dataset_card.md	Primary preprocessing report	Dataset summary card	File count, row count, profiled devices, grouped ratios, unknown-device rows, and effective zero-day mode
Taxonomy_Card.json	Taxonomy artifact	Locked semantic vocabulary	Canonical classes, attack families, policy labels, and mapping dictionaries
Split_Protocol.json	Split-governance artifact	Grouped split proof	Target ratios, actual ratios, grouped device counts, holdout classes, and split-governance configuration
Final_Row_Level_Split_Proof.json	Open-set proof artifact	Fallback and purity proof	Effective zero-day mode, holdout rows by split, contamination count, zero-day purity, and final routing evidence
Final_Scientific_Flags.csv	Gate audit table	Scientific gate decision record	Gate names, measured values, thresholds, optimization directions, statuses, severities, and rationales
Final_Metric_Summary.json	Metric summary artifact	Consolidated integrity metrics	FDSS, DFDI, FLS, AIS, ASCS, split drift, leakage risk, and zero-day integrity values
Train_Only_Preprocessor_Bundle	Transform artifact	Train-only preprocessing evidence	Scaling, clipping, imputation, and transformation statistics fitted strictly on the training partition
Diagnostic_Artifact_Contract.csv	Diagnostic harness table	Artifact-completeness verification	Presence, location, and size of required preprocessing and baseline-comparison artifacts

Table 7. *Cont.*

Artifact	Artifact Group	Role	Primary Contents
Diagnostic_Taxonomy_Contract.csv	Diagnostic harness table	Taxonomy verification	Class-count, family-count, policy-count, class-family mapping, class-policy mapping, and holdout-class checks
Diagnostic_Split_And_Fallback_Contract.csv	Diagnostic harness table	Split and fallback verification	Effective zero-day mode, contamination count, zero-day purity, holdout-class declaration, and split-count availability
Diagnostic_Scientific_Gate_Contract.csv	Diagnostic harness table	Gate-value verification	Gate values, expected thresholds, optimization goals, reported statuses, and threshold-pass checks
Diagnostic_Retained_Feature_Contract.csv	Diagnostic harness table	Shared-safe feature verification	Confirmation that the expected 12 retained shared-safe features were present
Diagnostic_Baseline_Comparison_Contract.csv	Diagnostic harness table	Baseline-comparison verification	Availability of random stratified, simple device-disjoint, and proposed-framework comparison outputs
Diagnostic_Final_Flags.csv	Diagnostic harness table	Final diagnostic decision record	Hard diagnostic contracts, statuses, evidence counts, and acceptance requirements
Diagnostic_Summary.json	Diagnostic harness metadata	Machine-readable final decision	Final diagnostic decision, hard-failure count, warning count, selected run directory, and harness output path

Together, these artifacts formed the evidentiary backbone of the study. The primary preprocessing outputs documented how the dataset was governed, transformed, and audited, while the diagnostic harness outputs verified that the reported substrate satisfied the required artifact, taxonomy, split, feature, gate, and baseline-comparison contracts. This structure made the accepted run reproducible, inspectable, and suitable for later model-focused experiments.

5.2. Evaluation Metrics

The accepted run was evaluated using split-integrity metrics, leakage metrics, feature-stability metrics, open-set purity metrics, and family-semantic stability metrics. The formal definitions of these quantities were introduced earlier in Equations (2)–(12). As clarified after Table 6, these measures were preprocessing-integrity metrics rather than downstream classifier-performance thresholds. Additional supporting quantities, such as device exclusivity score, vendor-conditioned entropy, clipping rates, outlier rates, and duplicate-flow statistics, provided diagnostic context.

Class imbalance was handled as a protocol-governance issue rather than through synthetic resampling. The preprocessing stage did not apply Synthetic Minority Oversampling Technique (SMOTE), random oversampling, or random undersampling because such operations could alter the empirical distribution being audited and could introduce additional leakage risks if performed before split finalization. Instead, the framework protected imbalance-sensitive validity by preserving class visibility where feasible, reporting holdout coverage, separating known-class and zero-day routing logic, and auditing family-level and class-level semantic consistency after split construction. Any later classifier trained on the emitted artifacts may apply class weighting, focal loss, balanced sampling, threshold

calibration, or other imbalance-aware learning strategies, but such operations must be fitted strictly within the training partition.

Table 8 organizes the metric families used to audit preprocessing quality and links each family to its operational purpose. Split-proportionality metrics measured whether the realized training, validation, and test partitions remained close to the target ratios. Leakage-control metrics measured device overlap, flow-signature Jaccard risk, and fingerprint leakage. Deconfounding metrics measured residual dependence between retained features and device identity. Stability metrics measured post-transform distributional drift using the Feature Distribution Stability Score and related distributional diagnostics. Open-set integrity metrics measured holdout contamination, zero-day purity, and holdout coverage. Semantic-preservation metrics measured whether broad attack-family structure and within-family class structure remained stable across partitions.

Table 8. Metric families used to audit preprocessing quality.

Metric Family	Primary Quantities	Operational Purpose
Split proportionality	Observed split ratios and percentage-point drift	Near-target train/validation/test allocation
Leakage control	Device overlap, flow-signature Jaccard risk, FLS	Absence of explicit cross-split memorization channels
Deconfounding	Device-Feature Dependency Index	Suppression of device-identifying predictors
Stability	Feature Distribution Stability Score, mean KS distance	Preservation of post-transform feature consistency
Open-set integrity	Holdout contamination count, zero-day purity, coverage	Correct isolation of unknown threats in test
Semantic preservation	Attack Invariance Score, Attack Semantic Consistency Score	Retention of family- and class-level attack meaning

Together, these metrics ensured that the final accepted decision was based on auditable preprocessing evidence rather than on classifier performance. This distinction was central to the study because the framework was designed to validate the trustworthiness of the data substrate before downstream model training.

6. Results

6.1. Dataset Profile and Class Imbalance

The accepted run first needs to be understood at the level of scale and partition structure. As shown in Figure 2, 25,154,393 rows are allocated to training, 5,117,974 rows to validation, and 5,400,040 rows to test. Figure 2 therefore visualizes the final split structure and shows that the achieved ratios remain close to the protocol targets. This result is important because it demonstrates that strict governance did not collapse the dataset into an unusable partitioning regime.

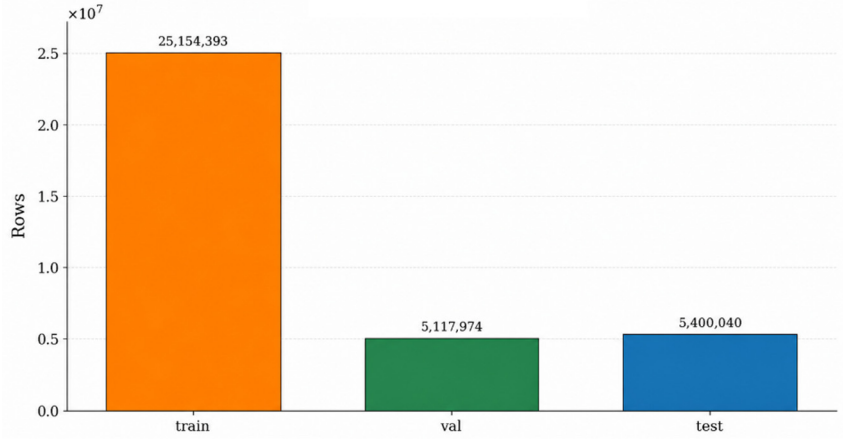

Figure 2. Final row counts by split.

6.2. Split Integrity and Open-Set Isolation

The most important integrity question is whether the final split remains both device-disjoint and open-set honest. Figure 3 addresses this question directly by showing that no holdout rows enter training or validation, while 73,450 holdout rows are emitted exclusively to test. In other words, the framework preserves the unknown-class challenge where it matters most: at evaluation time. This is the central novelty of the routing stage, because it replaces informal open-set claims with a verifiable isolation outcome.

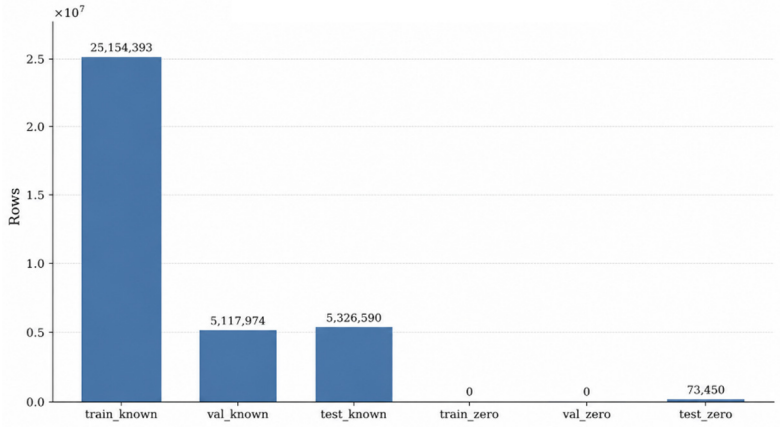

Figure 3. Known vs. zero-day rows by split.

The zero-day audit can be examined more directly through Figures 4 and 5. Figure 4 reports the zero-day purity audit, while Figure 5 reports the class-specific holdout counts for backdoor malware and cross-site scripting. Read together, these figures show not only that the holdout mechanism worked, but also that the final open-set test burden is transparent at the class level.

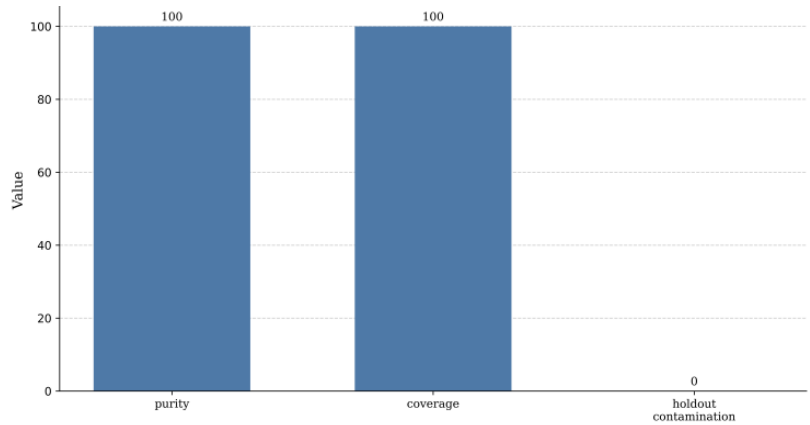

Figure 4. Zero-day purity audit plot.

Table 9 quantifies the final split structure using both row counts and grouped-device counts available at the protocol-lock stage. It complements the visual summary in Figures 2–5 by showing that row-level balance, grouped assignment, and holdout routing can be inspected in one compact view.

Table 9. Final split integrity summary for the accepted run.

Split	Rows	Actual Ratio	Target Ratio	Drift (pp)	Grouped Devices
Training	25,154,393	0.711945	0.70	1.195	115
Validation	5,117,974	0.144559	0.15	0.544	24
Test	5,400,040	0.143496	0.15	0.650	24

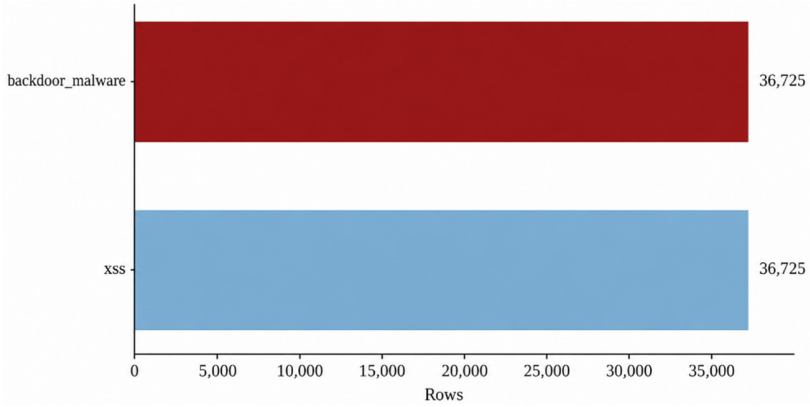

Figure 5. Holdout threat class counts in test split.

Table 10 complements Figures 3–5 by consolidating the final open-set routing outcome into one auditable summary. Its purpose is to show the effective zero-day mode, the holdout rows by split, the contamination count, and the resulting purity in a form that can be verified independently of the figures.

Table 10. Open-set routing outcome and zero-day integrity diagnostics.

Item	Value	Interpretation
Requested zero-day mode	Device-level	Initial protocol request
Effective zero-day mode	Row_Level_Fallback_Required	Final accepted routing mode
Holdout classes	backdoor_malware; xss	All holdout rows emitted only to test
Test-only holdout rows	73,450	No contamination of train or validation
Holdout contamination count	0	Holdout isolation fully preserved
Zero-day purity (%)	100.0	All designated holdout rows retained
Zero-day coverage (%)	100.0	Both held-out classes represented
Zero-day family diversity in test	2	Additional evidence of evaluation novelty
Unseen test device ratio (%)	50.98	All holdout rows emitted only to test

6.3. Leakage and Deconfounding Analysis

The leakage-control results are consolidated in Figures 6 and 7. Figure 6 summarizes the primary integrity metrics, namely, Feature Distribution Stability Score, Device-Feature Dependency Index, Fingerprint Leakage Score, Attack Invariance Score, and Attack Semantic Consistency Score. Figure 7 then provides a master integrity view that situates these values within the broader accepted-run decision. Together, these figures show that leakage suppression did not come at the cost of semantic collapse.

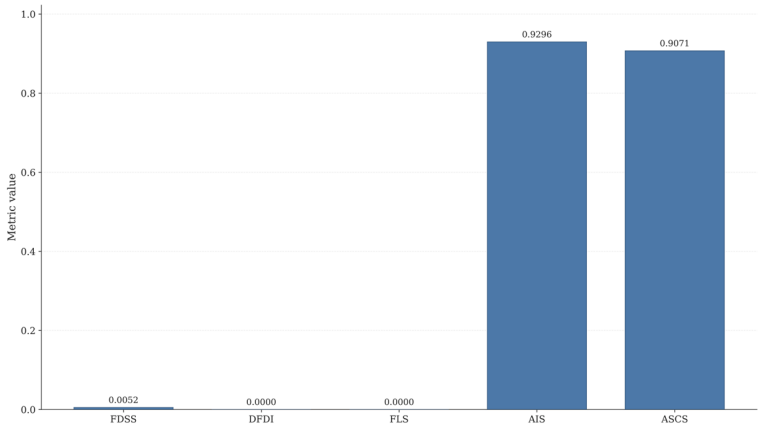

Figure 6. Integrity metrics bar chart.

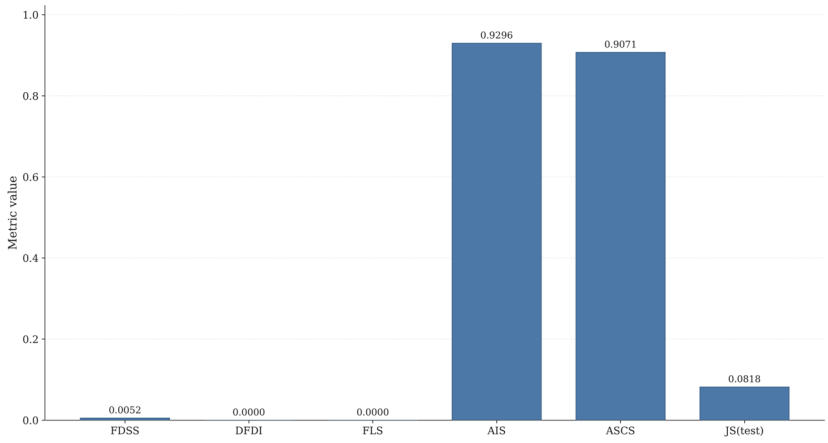

Figure 7. Master integrity summary radar.

Table 11 provides the exact gate outcomes that underlie Figures 6 and 7. By listing the final values, thresholds, optimization directions, and pass statuses together, the framework's acceptance logic is made transparent and reproducible.

Table 11. Final scientific gate outcomes for the accepted run.

Gate	Status	Value	Threshold	Goal
Device-disjoint MAC leakage	PASS	0.000000	0.0	MIN
Zero-day purity	PASS	100.000000	100.0	MAX
Flow-signature Jaccard risk	PASS	0.000000	0.0002	MIN
Split-ratio drift	PASS	0.652849	2.0	MIN
Quarantine rate	PASS	0.000000	1.0	MIN
Feature Distribution Stability Score (FDSS)	PASS	0.005180	0.02	MIN
Device-Feature Dependency Index (DFDI)	PASS	0.000000	0.1	MIN
Fingerprint Leakage Score (FLS)	PASS	0.000000	0.02	MIN
Attack Invariance Score (AIS)	PASS	0.929636	0.8	MAX
Attack Semantic Consistency Score (ASCS)	PASS	0.907144	0.8	MAX

As shown in Figure 8, the family-level attack invariance heatmap indicates that the retained representation preserves broad family structure across splits even after aggressive leakage suppression. Although a heatmap is only a supporting visualization, it reinforces the metric-based conclusion that semantic family behavior remains stable under the governed split.

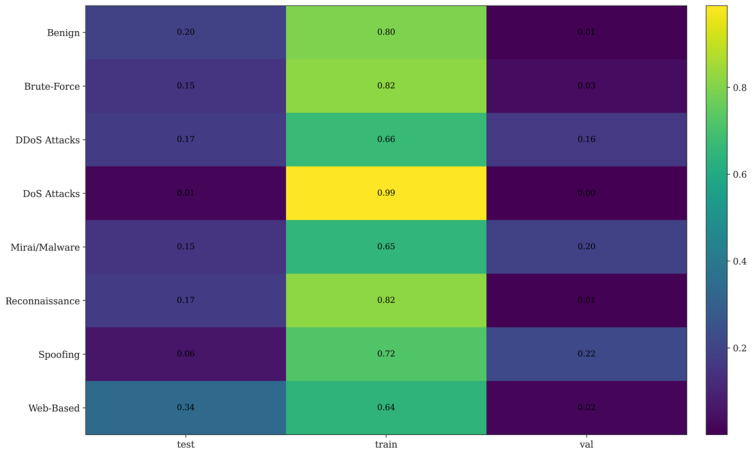

Figure 8. Attack-family invariance heatmap.

6.4. Feature Stability and Retained Representation

The retained 12 shared-safe features are drawn from protocol-level descriptors within the 136-column raw inventory summarized in Table 4. Figure 9 further examines this representation through a correlation heatmap, showing that most retained predictors remain weakly coupled, with only a small number of stronger relationships. This matters because the framework is not merely selecting low-leakage features—it is selecting a compact representation that remains structurally interpretable after aggressive deconfounding.

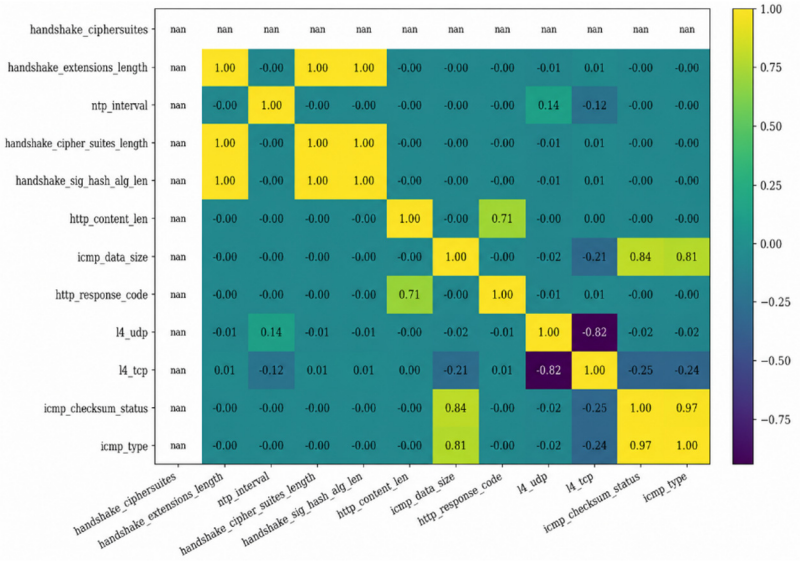

Figure 9. Feature correlation heatmap.

Outlier behavior is examined in Figure 10, which shows the feature-wise outlier distribution under the interquartile-range proxy for the retained shared-safe predictors. In combination with the earlier stability metrics, the figure indicates that the retained representation is not only leakage-safe but also numerically well-behaved.

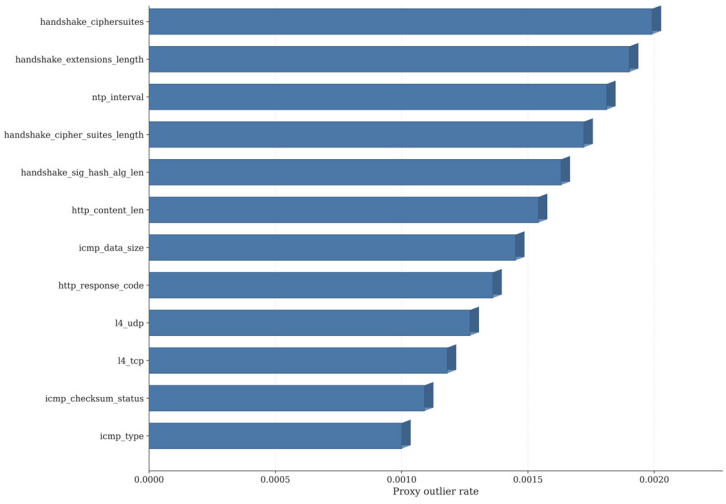

Figure 10. IQR outlier distribution by feature.

Table 12 complements Figures 9 and 10 by providing the scalar diagnostics that characterize feature health and transform stability. These values complete the argument that the final retained subset is compact, low-risk, and operationally stable.

Table 12. Feature-health and transform-stability diagnostics for the accepted run.

Diagnostic	Value	Interpretation
Missing rate (train/validation/test)	0.000%/0.000%/0.000%	No imputation burden
Clipping rate (train/validation/test)	1.311%/0.813%/1.513%	Limited clipping pressure after train-only fit
Variance retention	0.9167	Most retained variance preserved
Near-zero-variance features	1	Compact representation with minimal collapse
Feature entropy median/IQR	0.0217/0.0421	Low-entropy but stable retained representation
Correlation redundancy score/high-correlation pairs	0.151/4	Moderate redundancy only
Outlier rate (IQR/Z-score/MAD)	0.00199/0.01473/0.00000	Outlier burden remains low
Duplicate flow-signature rate (train/validation/test)	0.000%/0.000%/0.000%	No residual duplicate flow leakage

6.5. Final Scientific Decision and Diagnostic Harness Verification

The results culminated in a single scientific decision: the accepted run satisfied the framework's preprocessing-integrity criteria. The PREPROCESSING_SUMMARY artifact recorded an overall status of ACCEPTED together with the banner SCIENTIFIC INTEGRITY VERIFIED. This decision followed directly from the hard-gate structure reported in Table 11, where all required scientific gates passed under their declared thresholds. The final decision therefore closed the loop between protocol design, measured evidence, and auditable decision logic.

To strengthen the reproducibility of this decision, the accepted run was further verified using the diagnostic harness described in Section 4.8. The harness consolidated the emitted preprocessing artifacts into six hard diagnostic contracts: artifact completeness, taxonomy consistency, split/fallback proof, scientific gate verification, retained-feature consistency, and baseline-comparison validity. These contracts were evaluated using machine-readable evidence rather than manual inspection alone. Table 13 summarizes the diagnostic harness verification outcome by showing each diagnostic contract, along with its severity level, pass status, evidence count, and acceptance requirement.

Table 13. Diagnostic harness verification summary.

Diagnostic Contract	Severity	Status	Evidence	Requirement
Artifact contract	HARD	PASS	0 failures	All required artifacts existed
Taxonomy contract	HARD	PASS	0 failures	Taxonomy counts and mappings were valid
Split/fallback contract	HARD	PASS	0 failures	Split proof and fallback proof were valid
Scientific gate contract	HARD	PASS	0 failures	All hard scientific gates passed
Retained-feature contract	HARD	PASS	0 failures	The expected retained feature set was present
Baseline-comparison contract	HARD	PASS	0 failures	Baseline comparison and repeatability outputs were valid

As shown in Table 13, all six diagnostic contracts passed with zero hard failures. The diagnostic harness therefore produced the final decision ACCEPTED—SCIENTIFIC INTEGRITY VERIFIED, with zero hard failures and zero warnings. This result confirmed that the accepted preprocessing run was supported not only by the original gate outcomes, but also by a secondary verification layer that checked artifact completeness, taxonomy validity, split/fallback proof, retained-feature consistency, scientific gate correctness, and baseline-comparison availability.

The diagnostic harness also confirmed the manuscript-level values used in the accepted-run report. These included zero device-disjoint MAC leakage, 100% zero-day purity, zero flow-signature Jaccard risk, split-ratio drift of 0.652849, Feature Distribution Stability Score of 0.005180, Device-Feature Dependency Index of 0.000000, Fingerprint Leakage Score of 0.000000, Attack Invariance Score of 0.929636, and Attack Semantic Con-

sistency Score of 0.907144. These values were consistent with the scientific gate outcomes reported earlier and confirmed that the accepted decision was supported by both primary preprocessing artifacts and independent diagnostic verification.

Accordingly, the final decision should be interpreted as certification of the preprocessing substrate rather than certification of a downstream classifier. The accepted run demonstrated that the dataset partitioning, zero-day routing, retained-feature contract, leakage-control metrics, semantic-preservation metrics, and baseline-comparison artifacts satisfied the declared scientific-integrity requirements. This made the emitted substrate suitable for subsequent model-focused experiments involving baseline classifiers, multi-head intrusion-detection architectures, and ablation studies.

6.6. Baseline Split-Protocol Comparison and Repeatability Analysis

To contextualize the benefit of the proposed framework, a baseline split comparison was conducted against two simpler preprocessing strategies on the same CICIoT-DIAD 2024 corpus. The first baseline was a conventional random stratified row split, which approximated common row-level evaluation practice. The second baseline was a simple device-disjoint split, which assigned known-device rows to mutually exclusive partitions but did not apply the full vendor-aware balancing, zero-day routing, shared-safe feature selection, or hard-gate audit logic used by the proposed framework. Each baseline strategy was repeated across five random seeds and summarized using mean and standard deviation. The proposed framework was deterministic once the taxonomy, holdout classes, split targets, and gate thresholds were fixed; therefore, its accepted run was reported as the governed protocol outcome. Table 14 summarizes the resulting baseline comparison and repeatability analysis.

Table 14. Baseline split comparison and repeatability analysis.

Metric	Random Stratified Row Split	Simple Device-Disjoint Split	Proposed Framework
Split-ratio drift (pp)	0.0047 ± 0.0035	13.9577 ± 7.4512	0.6528
Device overlap count	114.0 ± 1.87	0.0 ± 0.0	0.0
Zero-day purity (%)	14.96 ± 0.16	12.51 ± 1.97	100.0
Flow-signature Jaccard risk	0.3701 ± 0.0288	0.2651 ± 0.0522	0.0000
Feature Distribution Stability Score (FDSS)	0.0011 ± 0.0004	0.0321 ± 0.0101	0.0052
Device-Feature Dependency Index (DFDI)	0.2671 ± 0.0000	0.2671 ± 0.0000	0.0000
Fingerprint Leakage Score (FLS)	0.3281 ± 0.0000	0.3281 ± 0.0000	0.0000
Attack Invariance Score (AIS)	0.9996 ± 0.0001	0.7818 ± 0.0316	0.9296
Attack Semantic Consistency Score (ASCS)	0.999995 ± 0.000001	0.9806 ± 0.0045	0.9071

The comparison showed that conventional random stratified splitting produced near-perfect family and class distribution stability, but at the cost of severe integrity violations. It allowed device overlap across partitions, produced high flow-signature Jaccard risk, and failed to preserve zero-day isolation. The simple device-disjoint baseline removed device overlap, but it introduced substantial split-ratio drift and still failed to isolate the designated zero-day classes. In contrast, the proposed framework was the only strategy that jointly achieved zero device overlap, zero flow-signature Jaccard risk, 100% zero-day purity, zero measured device-feature dependency, and zero fingerprint leakage while keeping split-ratio drift within the accepted tolerance. Although its AIS and ASCS values were lower than those of the random split, they remained higher than the hard acceptance thresholds, indicating that semantic structure was preserved under stricter leakage-control conditions.

This result clarified the role of the framework. The objective was not to maximize superficial distributional similarity between partitions, because such similarity can be

achieved through leakage-prone random mixing. Instead, the framework prioritized a more scientifically defensible evaluation substrate by jointly controlling device overlap, duplicate-flow risk, zero-day contamination, feature-level device dependency, and semantic preservation.

7. Discussion

The accepted run showed that rigorous preprocessing could remain both stringent and productive. The proposed framework preserved 100% zero-day purity, zero device overlap, and zero flow-signature Jaccard leakage risk, while retaining a compact 12-feature shared-safe representation. This result suggested that leakage suppression did not necessarily collapse the analytical substrate. Instead, when feature retention was governed by explicit integrity constraints, a smaller representation could still preserve protocol-level attack semantics while avoiding obvious device-identity and duplicate-flow leakage paths.

The baseline comparison further clarified the contribution of the framework. The random stratified split preserved distributional similarity very strongly, but it did so by allowing device overlap, high flow-signature Jaccard risk, non-zero device-feature dependency, non-zero fingerprint leakage, and poor zero-day purity. This confirmed that apparently stable train–test distributions could be misleading when they were produced by leakage-prone row-level mixing. The simple device-disjoint baseline removed device overlap, but it introduced large split-ratio drift and still failed to preserve the designated zero-day holdout classes. By contrast, the proposed framework was the only evaluated strategy that simultaneously achieved zero device overlap, zero flow-signature Jaccard risk, 100% zero-day purity, zero measured device-feature dependency, zero fingerprint leakage, and acceptable semantic preservation. This demonstrated that the framework's value did not lie merely in producing a balanced split, but in coordinating split governance, leakage auditing, holdout routing, feature safety, and final gate verification within a single auditable protocol.

The retained 12-feature representation should therefore be interpreted carefully. It was not presented as a universally optimal feature set for all downstream intrusion-detection classifiers. Rather, it represented the most conservative integrity-approved feature subset produced by the accepted run. The reduction was useful because it showed that a compact protocol-level representation could satisfy the framework's leakage, stability, missingness, transform-reliability, and semantic-preservation gates. However, larger leakage-safe feature sets may still improve downstream classifier performance, especially for fine-grained 28-class detection. Thus, the 12-feature subset should be understood as a defensible preprocessing substrate rather than as a final predictive feature-optimization claim.

A second important finding concerned open-set honesty. The requested zero-day mode was device-level holdout, but the final accepted run required row-level fallback. This weakened the strictest form of device-novel open-set evaluation, and the result should not be overstated as full device-level zero-day generalization. Nevertheless, the fallback was scientifically useful because it preserved zero contamination from the designated holdout rows and made the effective evaluation mode explicit. Rather than hiding the infeasibility of class-complete device-level holdout, the framework reported it, recorded the fallback, and emitted proof artifacts that allowed the compromise to be audited. This distinction was important because transparent boundary reporting is preferable to stronger but unsupported open-set claims.

The diagnostic harness strengthened this interpretation by verifying that the accepted run was supported by machine-readable evidence. The harness checked artifact completeness, taxonomy consistency, split and fallback proof, retained-feature consistency, scientific gate outcomes, and baseline-comparison validity. Therefore, the accepted decision should

be interpreted as the certification of the preprocessing substrate, not the certification of a downstream intrusion-detection model. This distinction matters because the present study evaluated whether the data foundation was trustworthy enough for later model training, rather than whether a specific classifier achieved the best possible accuracy, macro-F1, or open-set AUROC.

Relative to prior model-centric IoT intrusion-detection studies, the proposed framework shifted attention from classifier architecture to evaluation integrity. Many high-performing models can appear convincing when trained and tested on leakage-prone partitions, but such results may not reflect deployable security intelligence. The present framework addressed this problem by making preprocessing decisions visible, measurable, and auditable. Its contribution was therefore complementary to downstream model development: it provided a stronger data-governance foundation on which future device-identification, attack-classification, zero-day detection, and policy-prediction models could be evaluated more credibly.

The practical implication is that downstream models trained on the emitted artifacts would inherit a more defensible experimental substrate. This does not guarantee downstream excellence, and it does not replace the need for classifier-level validation. However, it reduces the risk that future performance claims will be driven by device overlap, duplicate-flow leakage, feature-level fingerprinting, or poorly documented holdout construction. Future model-focused work should therefore evaluate lightweight and deep classifiers on the proposed substrate, compare larger leakage-safe feature subsets against the current 12-feature representation, and quantify open-set performance using metrics such as macro-F1, per-class precision and recall, AUROC, AUPRC, and FPR@95%TPR for the designated holdout classes.

8. Limitations and Threats to Validity

A primary limitation of this study lies in its scope. The work was designed as a preprocessing and scientific-integrity protocol study rather than as a downstream classifier-performance study. Its contribution was therefore the establishment of a rigorous data-governance foundation upon which later multi-head tasks, including device identification, attack classification, zero-day detection, and policy learning, could be developed and evaluated. To strengthen the evidential value of the protocol, the study also evaluated baseline split comparisons against random stratified and simple device-disjoint strategies. These baselines were required because, without comparison to conventional preprocessing alternatives, the relative benefit of the proposed framework could not be quantified. The random stratified split represented a common distribution-preserving but leakage-prone practice, while the simple device-disjoint split represented a stricter device-aware alternative without the full leakage-audit, fallback-proof, feature-safety, and diagnostic-gate structure of the proposed framework. Comparing these strategies made it possible to show that the proposed framework did not merely generate an accepted preprocessing run, but achieved stronger integrity in terms of device overlap, flow-signature leakage risk, zero-day purity, device-feature dependency, fingerprint leakage, split stability, and semantic preservation. Thus, the baseline comparisons supported the preprocessing contribution without converting the manuscript into a downstream classifier study. A second limitation concerns the effective open-set configuration achieved in the accepted run. Although the protocol preserved zero contamination and 100% zero-day purity, the two held-out classes were not device-novel in the strictest sense because the final execution required row-level fallback rather than class-complete device-level holdout. This distinction is stated explicitly because the transparency of that compromise forms part of the framework's scientific value.

A further limitation concerns portability and computational cost. The protocol was instantiated on a specific IoT benchmark together with an associated Organizationally Unique Identifier enrichment file. While the underlying principles of device-disjoint governance, train-only fitting, and holdout auditing are transferable, threshold calibration, feature-retention behavior, and the computational cost of large-scale hashing and audit generation may vary across other corpora. These constraints do not diminish the validity of the present findings, but they do define the appropriate boundaries of their generalization.

9. Reproducibility and Artifact Availability

Reproducibility was an explicit design goal of the framework. The accepted run emitted versioned reports, metric summaries, split proofs, scientific flags, taxonomy dictionaries, retained-feature manifests, figure registries, diagnostic contracts, and baseline-comparison outputs. The core preprocessing artifacts included preprocessing_summary.md, dataset_card.md, taxonomy_card.json, split_protocol.json, final_row_level_split_proof.json, final_metric_summary.json, final_scientific_flags.csv, and the train-only preprocessor bundle. Together, these materials provided the evidence needed to reconstruct the accepted preprocessing decision without rerunning the full pipeline from scratch.

The framework also emitted diagnostic-harness artifacts that verified the internal consistency of the accepted run. These included artifact-completeness checks, taxonomy-contract checks, split/fallback-contract checks, scientific-gate verification, retained-feature verification, baseline-comparison verification, and the final diagnostic summary. These diagnostic outputs were important because they converted the accepted decision into a machine-readable audit trail rather than a purely narrative claim.

The raw source files were stored in Parquet format with embedded schema, and the framework maintained SHA-256 manifests for source tracking and verification. This design supported bit-level provenance checking for the audited inputs and helped ensure that the evidence trail remained stable across independent verification attempts. The artifact structure also separated grouped split proof, row-level fallback proof, train-only preprocessing evidence, retained-feature diagnostics, final metric consolidation, and baseline split-comparison outputs. This separation was essential because open-set feasibility, feature safety, and baseline comparison were evaluated through different evidence layers.

For the final publication package, the authors intend to release the preprocessing code, configuration files, figure-generation scripts, diagnostic-harness templates, baseline-comparison scripts, and non-sensitive derived artifacts in a controlled repository, subject to institutional approval and dataset-licensing conditions. Because the raw CICIoT-DIAD 2024 files and IEEE OUI registry were externally sourced, the repository will not redistribute third-party data unless redistribution is explicitly permitted. Instead, it will provide configuration files, artifact schemas, run manifests, and instructions that allow authorized users to reproduce the preprocessing and audit workflow using their own lawful copies of the source data. Until repository release, the generated audit artifacts and supporting outputs can be made available for scholarly inspection upon reasonable request.

10. Conclusions

This paper presented a scientific integrity framework for open-set Internet of Things intrusion-detection preprocessing with device-disjoint split governance. The framework formalized taxonomy locking, vendor-aware device profiling, grouped split construction, feasibility-aware zero-day routing, shared-safe feature selection, train-only preprocessing, diagnostic-harness verification, and hard scientific gate evaluation as one end-to-end protocol. Its main contribution was not a new classifier architecture, but a reusable and auditable preprocessing substrate for more trustworthy IoT security evaluation.

On the accepted run over 35,672,407 records from the CICIoT-DIAD 2024 corpus, the framework achieved zero device overlap, zero flow-signature Jaccard leakage risk, 100% zero-day purity, a Feature Distribution Stability Score of 0.00518, a Device-Feature Dependency Index of 0.00000, a Fingerprint Leakage Score of 0.00000, an Attack Invariance Score of 0.92964, and an Attack Semantic Consistency Score of 0.90714. These results showed that strong leakage control, feature-level deconfounding, and semantic preservation could coexist when preprocessing was governed through explicit scientific gates.

The baseline comparison further demonstrated the value of the proposed protocol. Random stratified splitting produced strong distributional similarity, but at the cost of device overlap, flow-signature leakage, and poor zero-day isolation. The simple device-disjoint baseline removed device overlap, but introduced substantial split-ratio drift and still failed to preserve the designated zero-day holdout classes. By contrast, the proposed framework was the only evaluated strategy that simultaneously satisfied the core integrity requirements for device overlap, duplicate-flow leakage, zero-day purity, device-feature dependency, fingerprint leakage, and semantic preservation.

The accepted run also clarified the boundary of the open-set claim. Strict device-level zero-day holdout was requested, but it was infeasible for the selected holdout classes without weakening supervised known-class support. The framework therefore activated an audited row-level fallback. This fallback preserved contamination-free holdout isolation, but it should not be interpreted as strict device-novel zero-day evaluation. This transparent reporting of the strongest feasible open-set condition was part of the framework's scientific value.

The retained 12-feature representation should likewise be interpreted as an integrity-approved compact substrate rather than as a universal predictive optimum. The feature-selection stage prioritized protocol-level semantic coverage, low device dependency, low identifier leakage, distributional stability, missingness control, and train-only transform reliability. Future model-focused studies may evaluate larger leakage-safe feature sets, but the present results showed that a compact representation could satisfy the required scientific-integrity gates.

Future work will extend this preprocessing substrate to downstream classifier validation using lightweight and deep learning models, including attack classification, device identification, open-set zero-day detection, and policy-aware response prediction. Such studies should report classifier-level metrics such as accuracy, macro-F1, per-class precision and recall, AUROC, AUPRC, and FPR@95%TPR, while preserving the same leakage-control and artifact-audit principles established in this work.

Author Contributions: Conceptualization, C.I.C., A.U.U., S.Z. and J.O.; methodology, C.I.C., A.U.U. and M.D.; software, C.I.C.; validation, C.I.C., A.U.U., M.D., S.Z., H.O.O. and J.O.; formal analysis, C.I.C. and H.O.O.; investigation, C.I.C., M.D. and S.Z.; resources, A.U.U., H.O.O. and J.O.; data curation, C.I.C.; writing original draft preparation, C.I.C.; writing review and editing, A.U.U., M.D., S.Z., H.O.O. and J.O.; visualization, C.I.C.; supervision, A.U.U., M.D. and S.Z.; project administration, A.U.U.; funding acquisition, M.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the TETFund National Research Fund (NRF) Intervention, grant reference number TETF/ES/DR&D-CE/NRF2024/SETI/ICT/00148/VOL1.

Data Availability Statement: Publicly available datasets were analyzed in this study. The study used the CICIoT-DIAD 2024 dataset together with the IEEE Organizationally Unique Identifier (OUI) registry. The derived preprocessing artifacts, split-protocol files, audit summaries, and supporting outputs generated during the study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors acknowledge the support of the Department of Telecommunications Engineering, Federal University of Technology, Minna, Nigeria, and all institutional collaborators who contributed technical and administrative support during the study.

Conflicts of Interest: The authors declare no conflicts of interest. The funder had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Chikezie, C.; Usman, A.; David, M.; Zubair, S.; Ohize, H.; Ojeniyi, J. Network Anomaly Detection in Enhanced Machine-Type Communication IoT Applications on 5G and Beyond Networks: A Convolutional Autoencoder Approach. *Conflu. Univ. J. Sci. Technol.* **2025**, *2*, 105. [CrossRef]
2. Chikezie, C.I.; Okpara, T.C.; Mmadumbu, A.C. Autoencoders for Anomaly Detection: A Comprehensive Architectural Review, Comparative Insights, and Practical Guidance. *Int. J. Eng. Res. Technol.* **2025**, *8*. [CrossRef]
3. Rabbani, M.; Gui, J.; Nejati, F.; Zhou, Z.; Kaniyamattam, A.; Mirani, M.; Piya, G.; Opushnyev, I.; Lu, R.; Ghorbani, A.A. Device Identification and Anomaly Detection in IoT Environments. *IEEE Internet Things J.* **2025**, *12*, 13625–13643. [CrossRef]
4. AlFuraih, D.; Mhamdi, L.; Karar, A.S. Explainable Hybrid CNN–XGBoost Framework for Multi-Class IoT Intrusion Detection with Leakage-Aware Feature Selection. *Appl. Syst. Innov.* **2026**, *9*, 49. [CrossRef]
5. Jain, P.; Rathour, A.; Sharma, A.; Chhabra, G.S. Bridging Explainability and Security: An XAI-Enhanced Hybrid Deep Learning Framework for IoT Device Identification and Attack Detection. *IEEE Access* **2025**, *13*, 127368–127390. [CrossRef]
6. Hossain, M.A. Deep Learning-Based Intrusion Detection for IoT Networks: A Scalable and Efficient Approach. *EURASIP J. Inf. Secur.* **2025**, *2025*, 28. [CrossRef]
7. Korba, A.A.; Boualouache, A.; Ghamri-Doudane, Y. Zero-X: A Blockchain-Enabled Open-Set Federated Learning Framework for Zero-Day Attack Detection in IoV. *IEEE Trans. Veh. Technol.* **2024**, *73*, 12399–12414. [CrossRef]
8. 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* **2025**, *25*, 7519. [CrossRef] [PubMed]
9. Ahmed, N.; Saleem, G.; Naveed, A.; Zaman, M.I. A Resource-Efficient Machine Learning Pipeline for DDoS Attack Detection: A Comparative Study on CIC-IDS2018 and CIC-DDoS2019. *ICCK Trans. Inf. Secur. Cryptogr.* **2026**, *2*, 55–69. [CrossRef]
10. Cruz, S.; Coleman, C.; Rudd, E.M.; Boult, T.E. Open Set Intrusion Recognition for Fine-Grained Attack Categorization. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
11. Tajgardan, M.; Shiranzaei, A.; Rabbani, M.; Khoshkangini, R.; Jamali, M. An Efficient Unsupervised Federated Learning Approach for Anomaly Detection in Heterogeneous IoT Networks. *arXiv* **2026**, arXiv:2602.24209. [CrossRef]
12. Sadatian Moghaddam, P.; Mahmoudi, M.; Serrano, N.; Hernando-Gallego, F.; Martín, D. Unified Anomaly Detection in IoT and Cyber-Physical Networks Using Evo-Transformer-LSTM: Validation on Four CIC Benchmarks. *Preprints* **2025**. [CrossRef]
13. Mgungile, J.A.; Tonkal, Ö. Scalable Intrusion Detection in IoT Networks: A Big Data Analytics Approach. *Turk. J. Eng.* **2026**, *10*, 230–243. [CrossRef]
14. Fraihat, S.; Yaseen, Q.; Sanjalawe, Y.; Abu-Errub, A.; Makhadmeh, S.N.; Al-Betar, M.A. Intrusion Detection in Industrial Internet of Things Network Using Feature Optimization and Hybrid Deep Learning. *Discov. Internet Things* **2026**, *6*, 34. [CrossRef]