



Kent Academic Repository

Varshani, Varshani, Huang, Yuxiang, Li, Haoxiang, Wang, Yichao, Brierley, Calvin, Arief, Budi, Oikonomou, George and Pope, James (2026) *Unsupervised Machine Learning for Anomaly Detection in Thread IoT Networks*. In: 2026 22nd International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT). . pp. 663-671. IEEE Computer Society

Downloaded from

<https://kar.kent.ac.uk/116433/> The University of Kent's Academic Repository KAR

The version of record is available from

<https://doi.org/10.1109/DCOSS-IoT69657.2026.00106>

This document version

Author's Accepted Manuscript

DOI for this version

Licence for this version

UNSPECIFIED

Additional information

Versions of research works

Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in **Title of Journal** , Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries

If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our [Take Down policy](https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies) (available from <https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies>).

Unsupervised Machine Learning for Anomaly Detection in Thread IoT Networks

Varshani Varshani[†], Yuxiang Huang^{*✉}, Haoxiang Li^{*✉}, Yichao Wang^{‡✉}, Calvin Brierley^{‡✉},
Budi Arief^{‡✉}, George Oikonomou^{*✉}, James Pope^{†✉}

^{*}School of Electrical, Electronic and Mechanical Engineering, University of Bristol, UK

[†]School of Engineering Mathematics and Technology, University of Bristol, UK

[‡]School of Computing, University of Kent, Canterbury, UK

Abstract—Thread is a low-power, IPv6-based mesh networking protocol increasingly deployed in smart home and industrial Internet of Things (IoT) environments. However, security monitoring is largely designed for traditional IP or Wi-Fi traffic and unexplored for Thread networks. Existing intrusion detection systems are ill-suited to the protocol’s constrained, fragmented, and protocol-specific communication patterns. In this paper, we address this gap by formulating anomaly detection in Thread networks as an unsupervised machine learning problem, where models are trained exclusively on normal traffic. We present an anomaly detection pipeline tailored to low-power Thread networks, incorporating packet-level annotation, session-based segmentation, and the extraction of 33 statistical and protocol-aware features. Using this representation, we design a deep autoencoder to detect anomalies by learning normal behaviour. To determine the autoencoder efficacy, we compare against the classical One-Class Support Vector Machine (OCSVM). We conduct experiments using Carnegie Mellon University’s (CMU) Thread dataset. In addition to normal behaviour, the dataset provides energy-depletion, session jamming, spoofing, and password guessing attacks. Our results show that the autoencoder consistently outperforms the OCSVM, achieving an overall F1-score of 87% compared to 60%. However, we show that the model has limitations detecting the spoofing and password guessing attacks. These results highlight the effectiveness of unsupervised approaches for modelling complex traffic patterns in low-power IoT networks and underscores the need for more adaptive detection strategies.

Index Terms—Internet of Things, Anomaly Detection, Intrusion Detection Systems, Unsupervised Learning, Autoencoder

I. INTRODUCTION

Thread is a low-power IPv6 mesh networking protocol [1] widely deployed in smart home environments, including smart locks, thermostats, and safety sensors [2]. As such devices increasingly control safety-critical functions, securing Thread networks has become essential. However, anomaly detection for Thread traffic remains largely unexplored, with existing work focusing mainly on protocol design and attack feasibility rather than detection mechanisms [3]. Anomaly detection in Thread networks differs fundamentally from that in traditional IP networks. Operating over IEEE 802.15.4, Thread produces lightweight and frequently fragmented packets, while 6LoWPAN header compression and short addressing schemes limit the applicability of deep packet inspection. To address these challenges, this paper proposes an anomaly detection

pipeline tailored to Thread traffic. Raw IEEE 802.15.4 packet traces are transformed through packet annotation and session-based segmentation from which 33 statistical and protocol-aware features are extracted. Two unsupervised models, a One-Class Support Vector Machine (OCSVM) baseline and a deep autoencoder, are trained exclusively on normal traffic and evaluated across four attack scenarios.

Experiments on the Carnegie Mellon University’s (CMU) Thread dataset [4] show that protocol-aware unsupervised anomaly detection can effectively identify representative attacks in Thread networks. The dataset has packets representing normal behaviour along with energy-depletion, session jamming, spoofing (i.e. PAN-ID), and password guessing attacks. Our results show that the deep autoencoder consistently outperforms the OCSVM baseline, demonstrating the effectiveness of unsupervised approaches in low-power IoT environments. Overall, the deep autoencoder achieves an 87% F1 versus the OCSVM of 60%. We further analyse the per-attack results and find that the autoencoder performs well on energy depletion and session jamming but poorly on spoofing and password guessing attacks. This paper makes the following contributions:

- We develop a Thread anomaly detection pipeline that enables systematic analysis of low-power Thread traffic using unsupervised learning.
- We perform packet-level annotation and protocol-aware preprocessing on the CMU Thread dataset, enabling session and window level analysis of Thread traffic for anomaly detection.
- We design a deep autoencoder and compare against the OCSVM on the Thread dataset, showing their performance across different attack scenarios.

II. RELATED WORK

A. Thread Protocol and Security Threats

Thread is an Internet of Things (IoT) networking protocol designed to address many limitations of earlier low-power wireless systems. It was adopted as the mesh component of the Matter standard, based on IEEE 802.15.4, to provide an IPv6-based mesh network for IoT devices [2], [5].

Thread networks offer resilience, scalability, and energy efficiency through mesh routing and sleepy end devices. Security

TABLE I: Thread Network Attack Characteristics

Attack	Targeted component	Mechanism	Impact	Detection difficulty
PAN-ID spoofing	PAN ID & Leader election/role	Rogue beacons/device advertises same PAN ID, triggering conflicts	Join disruption, disconnects, unstable topology	Beacon changes are observable but can resemble benign reconfiguration
Energy depletion	Sleepy End Devices (battery-powered)	Inject traffic or spoof pending/ACK to prevent sleeping and force retries	Accelerated battery drain, reduced availability	Traffic increase may be low-rate and hard to separate from normal churn
Password guessing	Commissioning (DTLS)	Induce handshake retries to test weak passcodes or impersonate a joiner	Unauthorised join, loss of integrity	Commissioning is rare; sporadic retries yield weak, noisy indicators
Selective jamming	Specific frame types (e.g., MLE, Data Request, MAC Data)	Targeted interference and spoofed frames aligned with legitimate timing/content	Targeted loss, poll/wake storms, energy drain	Effects mimic congestion/interference; malicious frames can appear legitimate

is a core design principle: all devices must be authenticated before joining, and communications are encrypted at the network layer using AES [5]. The availability of OpenThread, an open-source implementation, has established Thread as a standard for both commercial and academic research; nevertheless, it remains susceptible to protocol-specific attacks. Four primary attack types threaten Thread networks, as summarised in Table I. PAN-ID spoofing exploits network identifiers to cause device disconnections or prevent new devices from joining [3], [6]. Energy depletion attacks force battery-powered sleepy end devices into prolonged activity, rapidly draining their batteries through spoofed packets or forced retransmissions [7]. Password guessing targets the DTLS commissioning process, and attackers can exploit interruptions during handshakes to attempt credential spoofing. Selective jamming disrupts specific packet types such as Data Requests, and is particularly difficult to detect because the resulting traffic closely resembles legitimate activity [3].

B. Anomaly Detection in IoT Networks

Intrusion Detection Systems (IDS) are broadly categorised as signature-based or anomaly-based [8]. While signature-based systems quickly detect known threats, they fail against novel attacks, making them less suitable for dynamic IoT environments. Anomaly-based IDS build profiles of normal activity and flag deviations, offering better protection against zero-day threats [9]. OCSVM is a common unsupervised method for anomaly detection, learning a boundary around normal data without requiring labelled attack data [10]. However, their performance often degrades on high-dimensional IoT traffic. For this reason, it is often used as a baseline in comparative studies. Autoencoders have emerged as a powerful alternative for IoT anomaly detection. These neural networks learn to compress input data into a compact latent representation and reconstruct it, effectively modelling normal behaviour [11]. Anomalies are detected when reconstruction error exceeds a threshold. Autoencoders offer several advantages for IoT contexts: they require only normal data for training, capture complex non-linear relationships, naturally handle class imbalance, and are computationally lightweight for deployment on resource-constrained devices [12]. Recent studies have demonstrated their effectiveness, with Ayad et al. [12] achieving 99.99% accuracy on IoT datasets, and Tien et al. [11] reporting 98.36% F1-score in smart factory environments.

C. Datasets and Feature Engineering

Effective IDS evaluation requires suitable datasets. While datasets such as UNSW-NB15 and Bot-IoT have advanced IoT security research, they lack traffic from low-power IEEE 802.15.4 protocols [13], [14]. Akestoridis et al. [4] released the CMU Thread Dataset, which captures OpenThread-based network traffic using Nordic nRF52840 development boards, recording IEEE 802.15.4 packets in PCAP format. The dataset includes 35 experiments across three categories: packet analysis experiments establishing normal behaviour baselines, energy depletion attack experiments, and online password guessing attack experiments [3], [4]. All traces include encryption keys, enabling full traffic analysis down to MAC and 6LoWPAN layers. Effective anomaly detection hinges on feature engineering, as raw packet-level data often obscures malicious patterns emerging over interaction sequences. Research has shown that features aggregated into sessions or time windows substantially improve detection performance [15], [16]. This approach aligns with established practices like NetFlow and IPFIX, which group packets into flows to derive features such as duration and packet counts [17]. While well-studied in IP networks, such approaches remain rarely applied to Thread.

III. METHODOLOGY

A. Data Acquisition and Preprocessing

We used raw network traffic from the publicly available CMU Thread dataset, which contains PCAP traces collected from OpenThread-enabled development boards. The dataset includes four attack tracesets, PAE (PAN-ID Spoofing), EDAE (Energy-Depletion Attacks), and OPGAE (Password-Guessing and Session Jamming)—comprising a total of 35 PCAP files.

To establish ground truth, all PCAP files were manually analysed using Wireshark to identify attack onset and validate packet structures. Based on these analyses and the dataset documentation, packet-level annotation rules were defined for each attack type and subsequently automated using custom Scapy-based Python parsers. Each packet was labelled as normal or as a specific attack class and augmented with protocol-aware raw features.

To validate the annotation process, representative Wireshark captures were included for each attack type (attack_panID, attack_ED, attack_PG, attack_SJ) and normal. Figure 1 highlight the characteristic packet patterns that informed the packet-level annotation rules. Each example demonstrates how specific

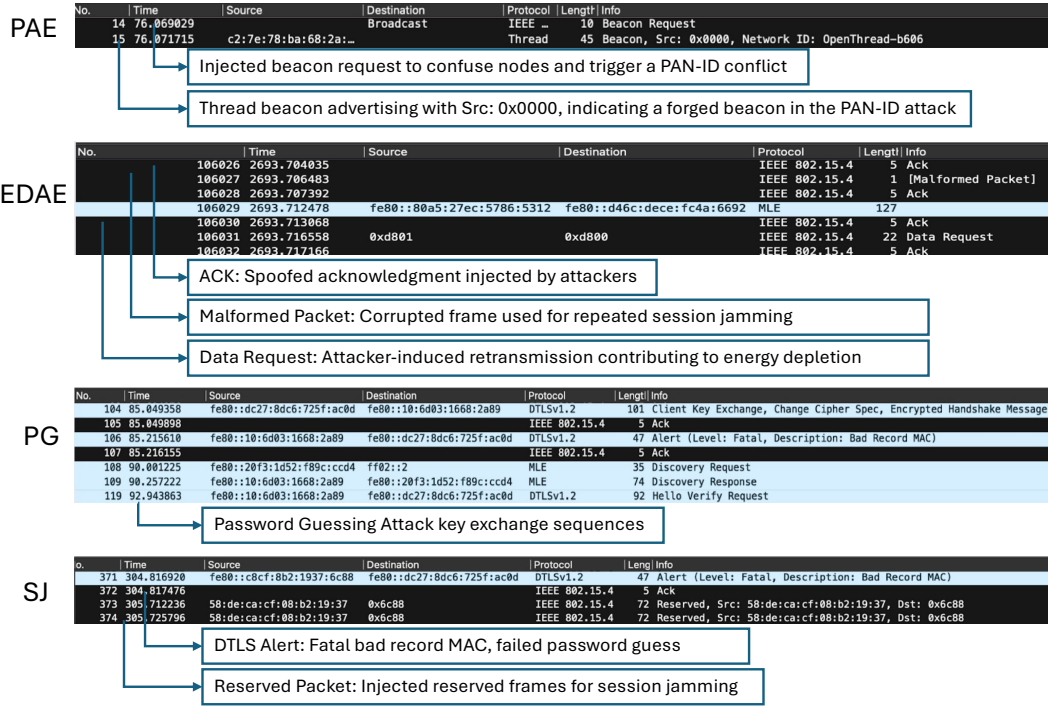


Fig. 1: Malicious packets during PAN-ID conflict attack

anomalies were identified and subsequently formalised into annotation criteria:

PAE (PAN-ID Spoofing): Beacon requests and PAN-ID assertions with anomalous or undefined source addresses (e.g., 0x0000, 0xDEAD, blank).

EDAE (Energy-Depletion Attacks): Attacker-induced Data Request, Forged ACK bursts and malformed packets (e.g., Bad FCS or ACK around malformed frames).

OPGAE – PG (Password Guessing): DTLS handshake sequences such as Alert (Level: Fatal, Description: Bad Record MAC), Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message, Client Hello, etc.

OPGAE – SJ (Session Jamming): Reserved/Malformed MAC frames continuous bursts.

B. Session-Based Segmentation

This study applies session-based segmentation to Thread network traffic to aggregate packet-level data into higher-level communication units suitable for feature extraction. Following concepts from NetFlow and IPFIX, a *flow* is defined as a unidirectional packet sequence ($src \rightarrow dst$), while a *session* represents a bidirectional exchange between two endpoints ($src \leftrightarrow dst$). This aggregation aims to generate stable and interpretable features for anomaly detection while preserving label purity, i.e., ensuring that each flow or session contains packets from only one traffic class.

1) *Session Construction:* Session-based segmentation was performed on the packet-level annotated dataset described in Section III-A, where each packet carries a traffic label. Packets were grouped into flows and sessions using composite session

keys that incorporate the traffic label, experiment identifier, endpoint pair, and temporal context. Fragmented packets were handled consistently to ensure correct grouping. By embedding both the label and experiment ID into the session key, the procedure prevents mixing of normal and attack traffic within a session and avoids overlap across experiments, thereby preserving reliable session-level labels.

2) *No address Exception:* Session construction primarily relies on source and destination addresses, consistent with conventional IP traffic analysis. This approach applies directly to most Thread traffic, including Energy Depletion Attacks (EDAE), which contain address-bearing packets.

However, Session Jamming (SJ) traffic frequently consists of bursts of addressless packets. Under standard address-based rules, such packets would be discarded, rendering the attack unobservable at the session level. To address this, an explicit no-address (NOADDR) exception was introduced for `attack_SJ`. Addressless packets were grouped into dedicated NOADDR sessions rather than being forced into artificial endpoint-based groupings.

This design preserves the visibility of jamming behaviour while maintaining semantic consistency in the dataset. As a result, NOADDR sessions exhibit distinct temporal and structural characteristics, which later contributed to the clear separability of SJ traffic in dimensionality-reduction analyses.

C. Timeout and Cap Rules

To preserve temporal coherence while preventing excessive session growth, protocol-aware timeout rules were applied during session-based segmentation, as shown in Table II. Short

TABLE II: Protocol-aware timeout values for Session-based Segmentation.

Traffic Type	Timeout (s)
Broadcast / Multicast	0.2
Fragmentation	0.5
Unicast	3.0
DTLS-secured traffic	10.0

TABLE III: Session-based Segmentation outcomes.

Metric	Value
Retained packets	130,323
Sessions	5,161
Flows	5,405
Label purity	100%

timeouts capture bursty and transient traffic, while longer thresholds accommodate legitimate delays in unicast and DTLS-secured communication. Table III provides a summary after applying these rules to the Thread dataset (451,212 packets). Packet reduction results from filtering incomplete or addressless traffic, except where explicitly preserved (e.g., NOADDR sessions for `attack_SJ`), yielding coherent, label-consistent communication units.

D. Segmentation and Feature Extraction

Session traffic was aggregated into fixed-length temporal segments to produce machine-learning-ready inputs that capture both traffic volume and temporal dynamics. Segmentation was performed per (`session_key`, `source_file`) to preserve session integrity and label consistency. Only packets with valid session keys, labels, and numeric timestamps were retained; empty columns and non-essential metadata were removed.

Addressless packets (e.g., all-zero, broadcast, or fragmented addresses) were excluded by default but explicitly preserved for attack types that rely on such behaviour, including `attack_SJ` and `attack_PANID`. Packets whose session keys contained NOADDR, FRAG, or broadcast identifiers were therefore retained.

A sliding-window approach with a fixed length of 15 s and 50% overlap (7.5 s step) was applied. For windows truncated at session boundaries, rate-based features were normalised using the actual window duration to avoid bias. This design preserves temporal structure without mixing unrelated traffic while maintaining observability of address-agnostic attacks.

1) *Feature Extraction*: Each session-aware sliding window (15 s duration with 50% overlap) was transformed into a fixed-length feature vector derived exclusively from packets within the window bounds. For windows truncated at session boundaries, all rate-based and time-normalised features were computed using the actual window duration to avoid bias introduced by incomplete windows.

Feature extraction was designed to capture complementary aspects of network behaviour relevant to anomaly detection, including traffic intensity, packet size and timing variability, address diversity, protocol composition, and contextual information inherited from the parent session and flow. The

extracted features were grouped by category, as summarised in Table IV.

Address entropy was computed using Shannon entropy:

$$H(X) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i), \quad (1)$$

where $p(x_i)$ denotes the probability of observing address x_i within a given window.

In total, 33 features were extracted per window. The final dataset consisted of 5,303 windows derived from 5,161 scoped sessions, providing a consistent representation of both benign and attack traffic. This window-level feature matrix forms the input to the anomaly detection models described in the subsequent section.

E. Dimensionality Reduction and Feature Correlation

Prior to model training, exploratory analyses were conducted to assess the structure, redundancy, and compressibility of the window-level feature space. This stage aimed to evaluate feature variance distribution, identify correlated features, and inform the selection of anomaly detection models.

1) *Principal Component Analysis*: Principal Component Analysis (PCA) was applied to the 33 window-level features to examine the intrinsic dimensionality of the dataset. By projecting the original features onto orthogonal components ordered by explained variance, PCA provided insight into how much information could be preserved under dimensionality reduction and whether the feature space exhibits strong compressibility.

2) *Feature Correlation Analysis*: Feature redundancy was assessed using Pearson correlation coefficients computed for all feature pairs. Strong correlations were identified using a threshold of $|r| > 0.8$, and the resulting correlation matrix was used to visualise dependencies among features. This analysis highlighted groups of overlapping features and confirmed the presence of correlated inputs.

Together, the PCA and correlation analyses indicated that the feature space is both redundant and partially compressible. Based on these observations, two anomaly detection models were selected: an OCSVM as a classical baseline sensitive to feature correlations, and an autoencoder as the primary model capable of learning compact nonlinear representations. This complementary setup supports robust evaluation of anomaly detection performance on Thread network traffic.

F. Machine Learning Models for Anomaly Detection

Two anomaly detection models were implemented to evaluate classical and deep learning approaches on Thread network traffic: an OCSVM as a baseline, and a deep autoencoder as the primary model. Both models were trained exclusively on normal traffic and evaluated on datasets containing both benign and attack windows.

1) *One-Class SVM*: The OCSVM was used as a baseline method suitable for one-class learning scenarios. Training was performed using only normal windows, and anomalies were identified as samples falling outside the learned decision

TABLE IV: Summary of features extracted at the window level, grouped by category.

Category	Features
Traffic Volume	packet_count, total_bytes, packet_rate, byte_rate
Packet Length & Timing	pkt_length_mean, pkt_length_std, pkt_length_max, pkt_length_min, iat_mean, iat_std, iat_p95
Address Diversity & Entropy	unique_src_addrs, unique_dst_addrs, src_addr_entropy, dst_addr_entropy
Protocol & Flow	mle_packets, thread_packets, ack_requests, fragments, unicast_packets, broadcast_packets, mle_ratio, thread_ratio, fragment_ratio, ack_ratio, unicast_ratio, broadcast_ratio, unique_flows_in_window, flow_starts_in_window
Session/Flow Context	session_duration, session_packet_count, flow_duration, flow_packet_count
Targets/Labels	attack_fraction, y_binary

boundary. An RBF kernel was employed to capture nonlinear feature relationships, and the ν parameter controlled the expected fraction of outliers in the training data. The model outputs a binary normal/anomalous classification and serves as a reference point for comparison.

2) *Autoencoder-Based Detection*: A deep autoencoder was implemented to learn a compact representation of normal traffic and detect anomalies via reconstruction error. The network takes the 33 window-level features as input and follows a symmetric encoder-decoder architecture:

$$33 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 33.$$

Dropout layers (rate 0.2) were applied throughout to improve generalisation. The model was trained using MSE loss with the Adam optimiser, with MAE tracked as an auxiliary metric. Input features were standardised prior to training, and only normal samples were used during optimisation.

Anomaly detection was performed by thresholding reconstruction error. The threshold was set to the 95th percentile of reconstruction errors observed on the normal training set, enabling data-driven separation between normal and anomalous behaviour without manual tuning.

3) *Comparative Evaluation*: Both models were trained and evaluated under identical conditions, enabling a direct comparison between a kernel-based classical approach and a nonlinear representation-learning method. Model performance was assessed using standard binary classification metrics, including precision, recall, F1-score, ROC-AUC, and PR-AUC. The F1-score was used as the primary evaluation metric due to class imbalance.

IV. EVALUATION AND RESULTS

This section evaluates the anomaly detection pipeline, examining model performance, the role of key features in shaping detection outcomes, and cases where the system struggled.

A. Functional Performance and Failure Cases

The functional evaluation of the anomaly detection models was conducted at both an overall and per-attack level (for autoencoder). This section critically examines their performance, identifies strengths and weaknesses, and highlights failure cases using quantitative results and supporting visualisations.

1) *Overall Model Comparison*: The baseline OCSVM and the deep autoencoder were first compared in terms of overall anomaly detection performance. As reported in Table V, the OCSVM achieved an overall F1-score of 60%, whereas the autoencoder achieved an F1-score of 87%. The substantial improvement demonstrates the autoencoder's superior ability to learn compact latent representations of normal traffic and to exploit non-linear feature interactions for anomaly detection.

This result established the autoencoder as the primary model for subsequent analysis, while the OCSVM served as a benchmark to highlight the benefits of adopting representation-based methods.

2) *Per-Attack Performance of the Autoencoder*: A more detailed breakdown of autoencoder performance across individual attack types is provided in Table VI. The results reveal significant variation, with some attack classes detected almost perfectly, while others remained highly challenging.

a) *Energy Depletion Attack (attack_ED)*: The autoencoder achieved an F1-score of 85% for attack_ED, the highest across all attack types. Both precision and recall were strong, indicating that attack_ED was consistently and reliably detected. This outcome aligns with the reconstruction error distributions (Figure 3), where attack_ED windows mainly produced errors well above those of normal traffic, creating an almost clear separation. The detection timeline (Figure 2) similarly shows that the majority of attack_ED windows exceeded the anomaly threshold, with only a small number of misses.

b) *Session Jamming Attack (attack_SJ)*: It recorded an F1-score of 43%, with recall = 100% (all 81 attack windows detected) but precision = 27%. In Figure 3, attack_SJ errors are shifted far from normal traffic, explaining the perfect recall. However, Figure 2 shows that some normal windows also spiked above the threshold, producing false positives and lowering precision. This outcome is consistent with the sessionisation strategy in Section III-B2, where addressless bursts were explicitly preserved as NOADDR sessions rather than being discarded. This methodological choice ensured that jamming traffic remained visible in the dataset and produced distinct statistical signatures, which explains its clear separability in PCA and the autoencoder's perfect recall. At the same time, retaining burst-driven traffic patterns increased sensitivity to superficially similar normal retransmissions, resulting in false positives and reduced precision.

TABLE V: Overall model performance comparison between OCSVM and Autoencoder.

Model(overall performance)	Precision	Recall	F1-Score	ROC-AUC	PR-AUC
OCSVM	55.71%	66.56%	60.65%	66.66%	65.41%
Autoencoder	81.69%	93.17%	87.06%	98.93%	96.83%

TABLE VI: Per-attack performance metrics of the Autoencoder model.

Label	Precision	Recall	F1-Score	ROC-AUC	PR-AUC
Overall	81.69%	93.17%	87.06%	98.93%	96.83%
Attack_panID	3.17%	53.85%	5.98%	94.66%	13.16%
Attack_ED	79.20%	93.04%	85.56%	98.97%	96.85%
Attack_PG	19.55%	94.55%	32.40%	98.92%	37.79%

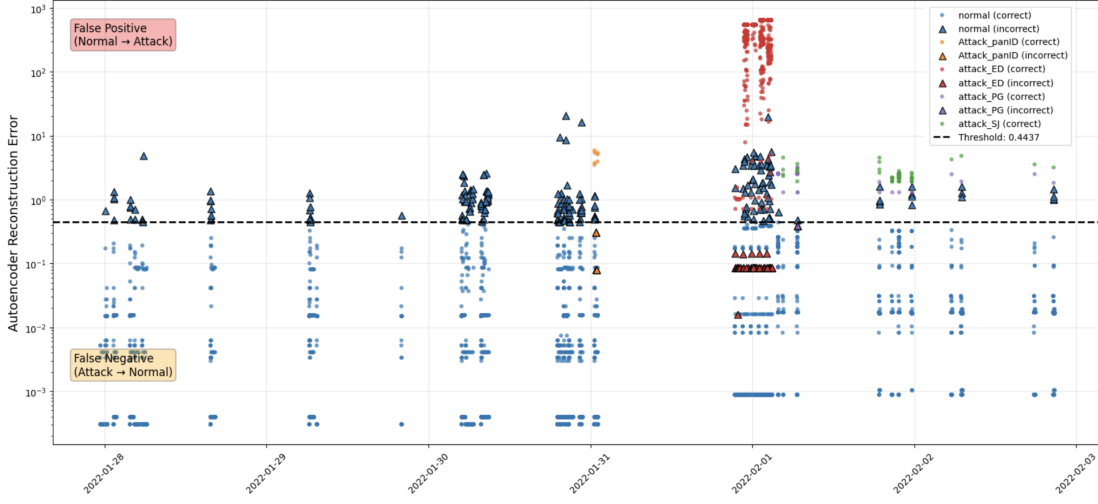


Fig. 2: Time-based reconstruction errors produced by the autoencoder. The dashed line indicates the anomaly detection threshold (0.443). Correctly classified samples cluster on their respective sides of the threshold, while false positives (normal misclassified as attack) and false negatives (attack misclassified as normal) are explicitly highlighted.

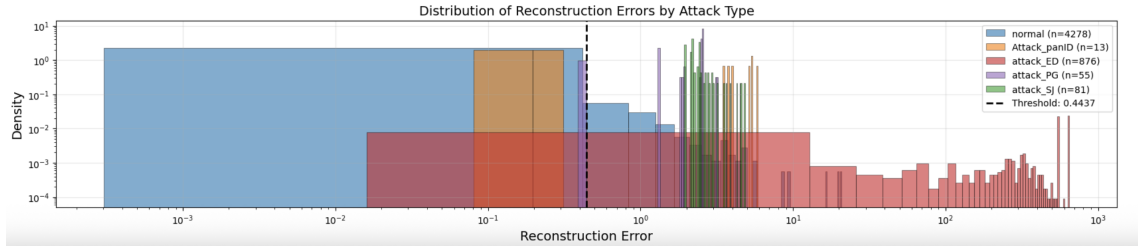


Fig. 3: Distribution of reconstruction errors by traffic class. Normal traffic is concentrated below the threshold, while attack distributions show varying degrees of separation. Clear separation is visible for Energy Depletion (ED) and Session Jamming (SJ), while Password Guessing (PG) and PAN-ID traffic overlap with normal, explaining their lower detection scores.

3) *Password Guessing Attack (attack_PG)*: `attack_PG` recorded an F1-score of 32%, characterised by high recall but very low precision. While nearly all `attack_PG` windows were detected (52 out of 55), a large number of false positives arose from normal DTLS retransmissions. The low precision of 19% indicates a high false positive rate, with the model flagging approximately two false positives per true positive. This significant misclassification rate arises because normal retransmissions produce patterns statistically similar to the attack traffic, causing the autoencoder to struggle with separation.

4) *PAN-ID Attack (attack_panID)*: This was the most challenging attack, with an F1-score of only 6%. As shown in Figure 3, `attack_panID` samples almost completely overlapped with the normal error distribution. In the detection timeline (Figure 2), fewer than half the `attack_panID` windows were flagged (7/13 detected). The reason is that PAN-ID attacks rely on beacon manipulation, which produces traffic statistically similar to normal beacon exchanges. The inherent similarity in feature values between the two classes presents a fundamental bottleneck, as the model cannot learn what the features fail to provide.

The evaluation shows both the strengths and limitations of the two models. The autoencoder outperformed the OCSVM baseline, which, despite using a non-linear RBF kernel, struggled with the high correlation and redundancy of the feature space. By compressing these features into a latent representation, the autoencoder captured complex patterns and achieved higher F1 scores, especially for attacks such as Energy Depletion `attack_ED` and Session Jamming `attack_SJ`. False positives from DTLS retransmissions and bursty normal traffic, along with poor detection of subtle attacks like Password Guessing `attack_PG` and PAN-ID `attack_panID`, highlight the pipeline’s constraints when attack traffic resembles normal behaviour. To better understand these outcomes, the next subsection examines the underlying feature behaviours and their role in shaping detection performance.

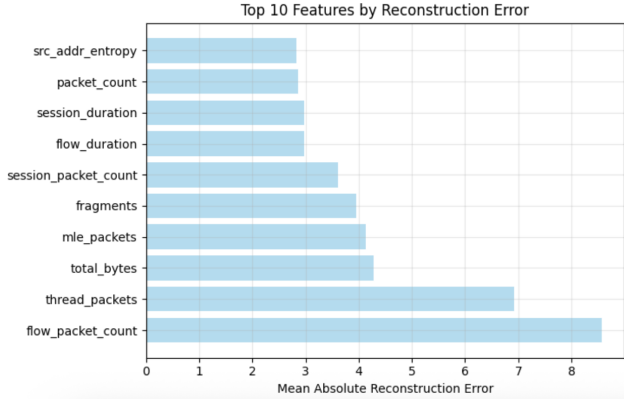


Fig. 4: Top 10 features by mean absolute reconstruction error.

B. Behavioural Testing and Insights

This subsection moves beyond raw metrics to explore why the models, particularly the autoencoder, behaved as they did. The analysis is not intended as definitive proof but as a set of informed hypotheses, grounded in (i) packet-level annotation conducted earlier in the project, (ii) the exploratory PCA and correlation analyses described in Section III, and (iii) feature-level reconstruction error patterns. Together, these perspectives help explain the behavioural tendencies observed across different attack types and highlight the interpretative value of the chosen feature set.

1) *Feature-Level Observations:* Figure 4 presents the ten features with the highest mean absolute reconstruction errors. These are interpreted as the features that most strongly influenced the model’s separation of normal and anomalous traffic. The ranking is dominated by flow- and packet-related metrics such as `flow_packet_count`, `thread_packets`, and `total_bytes`, suggesting that traffic intensity and flow dynamics were the most informative indicators of anomalies.

This hypothesis is consistent with packet-level annotation, where Energy Depletion Attacks (`attack_ED`) were characterised by inflated packet counts and repeated acknowledgements, behaviours that directly influence these features. It also aligns with the PCA analysis in Section III, which showed that

much of the dataset’s variance could be explained by packet- and flow-based dimensions.

By contrast, features capturing address diversity and entropy contributed relatively little to the reconstruction error ranking. A plausible interpretation is that these features do not change significantly under several attack scenarios. For example, `attack_panID` manipulates beacon frames without altering traffic intensity or address entropy in ways that sharply distinguish them from normal beacon traffic. This observation is reinforced by the correlation analysis, which showed that entropy features were only weakly related to the dominant packet- and flow-based variables.

2) *Hypotheses for Attack-Specific Outcomes:* The feature-level observations provide a useful lens for interpreting why the autoencoder’s per-attack detection varied so markedly in Section IV-A. These hypotheses are grounded in the earlier packet-level annotation, the PCA/correlation analysis in Section III, and the global feature reconstruction error rankings shown in Figure 4.

Energy Depletion Attack (ED): The high F1 score of 85% can be attributed to the attack’s strong effect on packet- and flow-level features. `attack_ED` inflated packet counts and acknowledgement activity, creating traffic volumes well outside the normal range. Features such as `flow_packet_count`, `thread_packets`, and `total_bytes` (all ranked among the top contributors in Figure 4) consistently reconstructed poorly, generating large errors that enabled the autoencoder to separate `attack_ED` traffic from normal with high reliability.

Session Jamming (SJ): `attack_SJ` achieved perfect recall but low precision (F1 score 43%). The addressless, bursty frames that define jamming traffic strongly deviated from normal flows, producing high reconstruction errors and guaranteeing detection. However, volume-driven features such as `flow_packet_count` and `session_packet_count` occasionally spiked in normal traffic due to bursty retransmissions, mimicking the jamming signature and leading to false positives. The prominence of these features helps explain why recall was maximised, but precision was compromised.

These outcomes were also shaped by the methodological decision in Section III-B2 to preserve addressless packets as NOADDR sessions. By explicitly retaining this traffic rather than discarding it, the pipeline ensured that jamming behaviour was visible at the session level and produced unique feature distributions. This explains the distinct separability observed in PCA and the autoencoder’s perfect recall, while also highlighting the trade-off of increased false positives from superficially similar normal retransmissions.

Password Guessing (PG): `attack_PG` recorded an F1 score of 32%, reflecting high recall but very low precision. This outcome is consistent with the similarity between PG traffic and normal DTLS retransmissions, both of which produce repeated handshake messages. Features such as `mle_packets` and `fragments`, which appeared in the top reconstruction error rankings, were disproportionately activated by these sequences. While they helped ensure that

most PG attempts were flagged (high recall), their redundancy with normal retransmission patterns caused frequent misclassifications, driving down precision.

PAN-ID Conflict: The weakest detection performance (F1 score 6%) was observed for `attack_panID`. These attacks rely on beacon frame manipulation, which preserves traffic volumes similar to normal beacon exchanges. As the dominant features highlighted in Figure 4 were volume- and flow-oriented (e.g., `total_bytes`, `packet_count`), they did not provide meaningful separation between PAN-ID and normal traffic. Features related to address entropy or diversity, though included, contributed minimally to reconstruction error, leaving the model unable to capture the subtle protocol-level irregularities characteristic of this attack.

3) *Interpretative Value:* The evaluation shows that model performance was shaped primarily by the discriminative capacity of the features rather than the modelling technique itself. Strong results for Energy Depletion reflected the clear impact of inflated packet and flow activity (e.g., `flow_packet_count`, `thread_packets`), which the autoencoder consistently reconstructed poorly compared to normal traffic. In contrast, the effectiveness against Session Jamming was shaped not only by feature behaviour but also by the methodological decision in Section III-B2 to preserve addressless bursts as NOADDR sessions. This design ensured that jamming traffic was retained, producing distinct feature profiles that guaranteed detection and yielded perfect recall. However, this choice heightened sensitivity to superficially similar bursty retransmissions, reducing precision.

By comparison, subtle attacks such as Password Guessing and PAN-ID conflicts produced feature patterns closely resembling normal behaviour, limiting their separation. This outcome is consistent with the PCA and correlation analyses, which indicated that variance was concentrated in packet- and flow-related dimensions, while entropy- and address-based features contributed less. The limited discriminative power of these latter features explains the shortcomings observed in detecting beacon-based anomalies such as PAN-ID conflicts and retransmission-heavy patterns such as Password Guessing.

C. Evaluation of Design Decisions

This part evaluates design choices and their impact on outcomes in Sections IV-A and IV-B.

1) *Sessionisation:* Address-based sessionisation ensured interpretability and resulted in 100% label purity, meaning no session contained mixed traffic types. However, this came at the cost of discarding almost 70% of the original packets, reducing the dataset from 451,212 to 130,323. This filtering improved data quality but reduced overall coverage, especially for attacks that relied heavily on addressless or fragmented packets. Importantly, though, `attack_panID` traffic was not affected by this step, as its packets carried valid addresses and were retained in sessions. The challenges with `attack_panID` detection therefore stemmed not from sessionisation but from the features extracted later in the pipeline.

A notable exception was introduced for Session Jamming (attack_SJ). Unlike Energy Depletion, which contained valid addressing information and could be processed under the standard rules, Session Jamming traffic consisted largely of addressless bursts that would otherwise have been discarded. To preserve the visibility of this attack type, the pipeline designated such traffic as NOADDR sessions. This methodological adjustment ensured that SJ remained observable at the session level and contributed to its clear separability in PCA and perfect recall during detection. However, the trade-offs are: preserving burst-driven traffic patterns increased sensitivity to similar normal retransmissions, resulting in false positives and lower precision. This illustrates how attack-specific exceptions in sessionisation can directly shape downstream model behaviour, balancing detectability against precision.

2) *Sliding Windows:* The use of a 15-second sliding window with 50% overlap was intended to balance temporal resolution with stability in feature extraction. This configuration worked well for high-frequency attacks such as Energy Depletion and Session Jamming, where the frequent packet transmissions ensured that anomalous patterns were consistently represented within the windows. At the same time, the fixed duration sometimes diluted short-lived bursts of abnormal traffic by blending them with neighbouring normal activity, thereby reducing precision. Although shorter windows might have captured these transient bursts more effectively, they would also have increased noise and false alarms. Overall, the chosen design proved robust across most scenarios, though it was not equally well suited to all attack types.

3) *Feature Set:* A total of 33 features were engineered to capture traffic volume, timing statistics, address diversity, flow activity, and protocol composition. These provided a strong basis for detecting attacks that induce major statistical shifts, such as Energy Depletion and Session Jamming. However, feature correlation analysis revealed substantial redundancy, with 48 pairs of highly correlated variables (e.g., `packet_count`, `ack_requests`, `unicast_packets`). While this redundancy did not prevent the autoencoder from achieving high accuracy, it hindered the OCSVM baseline, which relies heavily on feature independence. Additionally, features such as entropy and address diversity contributed little discriminative power in practice, explaining the model's failure to separate `attack_panID` from normal beacons. This underlines that the usefulness of features was uneven and attack-dependent.

4) *Thresholding Strategy:* The anomaly detection threshold was first set at the 95th percentile of reconstruction errors from normal training data (0.44). This provided a clear and statistically grounded cutoff, yielding an overall F1-score of 87%. Further threshold analysis showed that moving the cutoff to the 99th percentile (1.73) increased the F1-score to 92%, an improvement of 5%. This outcome highlights the trade-off in threshold choice: the 95th percentile favoured sensitivity and helped capture subtle anomalies, but also led to additional false alarms, as normal traffic windows occasionally exceeded the threshold. By contrast, the stricter 99th percentile reduced

false alarms and improved balance across metrics, though at the expense of potentially missing marginal anomalies. These results highlight the importance of careful threshold calibration and suggest that adaptive or context-specific thresholds may offer a more reliable approach.

5) *Model Selection*: The OCSVM, even with a non-linear RBF kernel, was hindered by strong feature redundancy and achieved an overall F1-score of 60%. In contrast, the autoencoder compressed these correlated features into a low-dimensional latent space and achieved a higher overall F1-score of 87%, demonstrating clear advantages against attacks that produced distinct statistical deviations. The main drawback was interpretability. The OCSVM defines explicit decision margins that are relatively easy to examine, whereas the autoencoder depends on reconstruction errors, which are less straightforward to interpret. On balance, the autoencoder was selected as the stronger model, with its superior detection performance outweighing the reduced interpretability.

6) *Overview of Design Decisions*: The pipeline's design choices collectively enabled strong detection performance for attacks that induced clear statistical deviations (`attack_ED`, `attack_SJ`) but limited its ability to detect subtle anomalies (`attack_PG`, `attack_panID`). Sessionisation prioritised purity at the cost of coverage, sliding windows balanced continuity and granularity, and the feature set provided breadth but included redundancy and weak indicators for beacon-level anomalies. Thresholding successfully captured anomalies but at the cost of precision, while model selection favoured accuracy and robustness over interpretability.

A key distinction emerged in the handling of Energy Depletion and Session Jamming traffic. Energy Depletion packets carried valid addresses and could be retained under the standard address-based sessionisation framework, ensuring label purity and consistency with the overall design. In contrast, Session Jamming consisted largely of addressless bursts, which would have been discarded under conventional rules. To preserve the visibility of this attack type, a `NOADDR` exception was introduced, allowing SJ traffic to form its own dedicated session class. This ensured that jamming behaviour remained detectable, resulting in high separability in PCA and perfect recall during detection, albeit at the cost of reduced precision due to false positives from bursty normal traffic. This reflects how attack-specific characteristics guided methodological decisions, balancing analytical clarity with protocol-specific adaptation.

V. LIMITATIONS AND CONCLUSION

A limitation of this study was the reliance on the CMU Thread dataset, limiting the assessment of model generalisability across diverse scenarios. Future work will consider other datasets and also include more recent baseline comparisons, such as embeddings for anomaly detection. The paper addressed a gap in Thread network security by developing a structured pipeline for anomaly detection, trained solely on normal traffic and evaluated across multiple attack scenarios. By combining packet-level annotation, sessionisation,

windowing, feature extraction, and model training, the work provides both a methodological contribution and an empirical benchmark for future studies in this area.

VI. ACKNOWLEDGMENT

This work was supported by the UKRI Engineering and Physical Sciences Research Council (EPSRC), Grant Numbers EP/X036871/1 and X036707/1.

REFERENCES

- [1] H.-S. Kim, S. Kumar, and D. E. Culler, "Thread/openthread: A compromise in low-power wireless multihop network architecture for the internet of things," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 55–61, 2019.
- [2] Connectivity Standards Alliance, "Matter (formerly project chip) is creating more connections between more objects, simplifying development for manufacturers and increasing compatibility for consumers," <https://github.com/project-chip/connectedhomeip>, n.d.
- [3] D.-G. Akestoridis, V. Sekar, and P. Tague, "On the security of thread networks: Experimentation with openthread-enabled devices," *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22)*, pp. 233–244, 2022.
- [4] —, "Captured thread packets from openthread-enabled development boards," <https://iee-dataport.org/open-access/crawdad-cmuthread-d-boards>, 2022, submitted by CRAWDAD Team; last updated 20 June 2023. Data format: .zip.
- [5] Silicon Laboratories Inc., "Ug103.11: Thread fundamentals," Silicon Laboratories Inc., Tech. Rep. Rev. 1.7, 2025. [Online]. Available: <https://www.silabs.com/documents/public/user-guides/ug103-11-fundamentals-thread.pdf>
- [6] D.-G. Akestoridis, M. Harishankar, M. Weber, and P. Tague, "Zigator: Analyzing the security of zigbee-enabled smart homes," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20)*. ACM, 2020, pp. 77–88.
- [7] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, 2016.
- [8] J. Veeramreddy, V. V. R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011. [Online]. Available: <https://ijcaonline.org/archives/volume28/number7/3399-4730>
- [9] R. Panigrahi, S. Borah, A. K. Bhoi, and P. K. Mallick, "Intrusion detection systems (ids)—an overview with a generalized framework," in *Cognitive Informatics and Soft Computing*, ser. Advances in Intelligent Systems and Computing, P. K. Mallick, V. E. Balas, A. K. Bhoi, and G.-S. Chae, Eds. Singapore: Springer, 2020, vol. 1040, pp. 107–117.
- [10] A. Binbusayis and T. Vaiyapuri, "Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class svm," *Applied Intelligence*, vol. 51, pp. 7094–7108, 2021.
- [11] C.-W. Tien, T.-Y. Huang, P.-C. Chen, and J.-H. Wang, "Using autoencoders for anomaly detection and transfer learning in iot," *Computers*, vol. 10, no. 7, p. 88, 2021.
- [12] A. G. Ayad, M. M. El-Gayar, N. A. Hikail, and N. A. Sakr, "Efficient real-time anomaly detection in iot networks using one-class autoencoder and deep neural network," *Electronics*, vol. 14, no. 1, p. art.104, 2025.
- [13] H. Kangale, "Unsw-nb15 complete dataset," <https://www.kaggle.com/datasets/harshwardhanbhangale/unswn-complete-dataset>, 2025.
- [14] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.
- [15] B. Boswell, S. Barrett, S. Rajaganapathy, and G. Dorai, "Flare: Feature-based lightweight aggregation for robust evaluation of iot intrusion detection," <https://doi.org/10.48550/arXiv.2504.15375>, 2025, arXiv preprint arXiv:2504.15375.
- [16] K. Kostas, M. Just, and M. A. Lones, "Individual packet features are a risk to model generalisation in ml-based intrusion detection," <https://doi.org/10.48550/arXiv.2406.07578>, 2024, arXiv preprint arXiv:2406.07578.
- [17] A. Sperotto, "Flow-based intrusion detection," Ph.D. dissertation, University of Twente, Zutphen, 2010.