Efficient and Reproducible Payload-Based Feature Engineering for Network Anomaly Detection

Taha Rasheed, Shahroze Naveed

Abstract

This study revisits the S_FE feature engineering method from the 2024 IJISAE paper on network anomaly detection. The original work relied on a private dataset and did not share code, so the method was rebuilt on a public source, the CIC-IDS2017 Friday PCAP set with about 9.94 million packets. Seven payload-level features were extracted, keeping the paper's structure in mind. While replicating the approach, a major issue surfaced. The Trust Value feature becomes unrealistically strong if it depends on packet labels. In many unofficial implementations, it does, which leads to data leakage. Once that leaked signal is removed and replaced with a simple rate-based heuristic, the picture changes. Linear models like Logistic Regression and LDA collapse to near zero detection. Non-linear models handle the data better. A Random Forest reaches an F1 score near 0.9186 and sets a realistic baseline for this dataset. A small MLP captures different patterns but sits lower at 0.7079. A simple ensemble that averages Random Forest and MLP probabilities stays close to the Random Forest score and adds more stability. The findings highlight how much a single leaked feature can distort results and suggest more grounded ways to compute a Trust Value, such as using connection success rates, port patterns, timing behaviour, or payload size variability. The outcome is a clear, reproducible pipeline that avoids hidden signals and sets practical baselines for future work.

Keywords—

Network intrusion detection, payload-based features, feature engineering, Trust Value, anomaly detection, machine learning, deep learning, ensemble learning, packet analysis, CIC-IDS2017

1. Introduction

Network traffic keeps growing, and with it comes a steady rise in intrusion attempts. Most intrusion detection systems rely on long lists of signatures or large feature sets that take time to compute. The S_FE method introduced in the 2024 study tried to simplify this by focusing on a small set of payload features. The idea is appealing. If a few lightweight features can separate benign traffic from attacks, then real time detection becomes easier to scale.

The original paper reported strong performance from simple models like Logistic Regression and LDA. That result stood out because linear models usually struggle with packet level features unless one feature carries a lot of information. The paper did not share its dataset or code, so the method needed to be rebuilt from scratch to understand how the features behave on public data. CIC-IDS2017, and specifically the Friday Working Hours PCAP segment, provided a suitable base. It contains a large mix of benign and attack traffic collected through controlled simulations.

During replication, the core features were easy to reconstruct except for one. The Trust Value feature lacked a clear definition in the original work. The description suggested a score for how reliable a source IP might be, but did not give a formula. This gap is important. Certain interpretations of Trust Value can accidentally absorb label information, which produces inflated accuracy that does not generalize. Once that possibility was examined more closely, the feature set and model behavior shifted in a noticeable way.

The goal of this study is to document that shift. The rebuilt pipeline extracts seven payload-based features, tests both linear and non-linear models, and compares the results before and after removing the leaked signal. The study then extends the original method by adding a small neural model and a simple ensemble. The focus throughout is on clarity and reproducibility. Any feature used here can be computed directly from live traffic without knowing packet labels, and the full pipeline runs on a publicly available dataset.

The results show where the S_FE idea holds up and where it needs refinements. They also open room for more grounded definitions of Trust Value that rely on observable behavior, not hidden labels. The overall aim is not to challenge the original study, but to create a clean reference point that future work can build on.

2. Problem Statement

Rebuilding the S_FE method raised a simple question. How well does this feature set work when tested on a public dataset, and can the results reported in the original study be reproduced without access to its private data or code. The main challenge sits inside the feature definitions, especially Trust Value. A vague description in the original paper leaves room for an interpretation that quietly mixes label information into the feature itself. Once that happens, any model that uses the feature gets an unrealistic advantage.

The problem, then, is twofold. First, the original performance claims need to be tested under conditions where no hidden signals are present. Second, the S_FE idea needs to be evaluated in a way that reflects how an intrusion detection system works in practice. Incoming packets do not carry ground truth labels. Any feature that depends on them, directly or indirectly, cannot be deployed.

The study also looks at whether a small feature set is enough on its own. Linear models are attractive for their speed, but their success depends on how separable the classes are. If the features do not capture meaningful patterns, linear boundaries fall apart. In that case, more flexible models might be needed, but those only matter if the features are sound.

Overall, the task is to build a clean, reproducible version of the S_FE approach, test it under realistic conditions, and explore improvements that stay true to what can be computed from real traffic. The outcome should give a clearer view of what the method can actually deliver without relying on hidden information.

3. Related Work

Most intrusion detection research leans on larger feature sets built from network flows rather than raw packets. A widely used reference is the CIC-IDS2017 analysis by Sharafaldin et al. [1], which showed that decision tree based models handle heterogeneous attack traffic better than linear classifiers. Their work set the standard for evaluating machine learning approaches on this dataset.

Payload driven methods form another cluster of studies. Shapira and Kenig [2] examined byte frequency patterns and entropy measurements as lightweight signals, especially for encrypted traffic. Their findings suggested that simple payload statistics can capture useful structure, though they rarely produce clean linear separations on their own.

A third line of work focuses on handcrafted packet features combined with non linear models. Alshamrani et al. [3] experimented with custom payload descriptors and small neural networks. Their results pointed toward the same pattern seen elsewhere: the features help, but only when defined consistently and free from hidden label information.

Recent efforts build on these ideas by targeting real time packet level detection. For example, Wang et al. [4] proposed a system that uses recurrent autoencoders to embed sequences of packets into compact features. They fed these into classifiers for microsecond speed detections. Explainability came from hardware acceleration with memristors, which cut computation time without losing accuracy. This approach suits high speed networks where delays matter.

Other studies incorporate advanced deep learning for payload analysis. Li et al. [5] combined convolutional masked autoencoders with large language models to spot anomalies in packet payloads. Their method detected zero day attacks by learning normal byte patterns, then flagging deviations. Tests on public datasets showed gains in recall over signature based tools, though false positives rose in noisy traffic.

Hybrid models also gain traction for IoT settings. Kaliyaperumal et al. [6] blended unsupervised self organizing maps with supervised classifiers. They focused on payload entropy and flow stats to handle unknown threats. Results indicated better adaptability than pure linear setups, but required careful feature tuning to avoid overfitting.

These studies form the basis for evaluating the S_FE method. They highlight the potential of small feature sets, but also the risks that appear when a feature unintentionally carries label information. Replication efforts like this one test those risks directly, ensuring features stay deployable in label free environments.

4. Dataset Description

The original S_FE paper used a private dataset built from manual packet captures, which made direct replication impossible. To keep the evaluation transparent, this study relies on the Friday Working Hours portion of the CIC-IDS2017 dataset. This segment is well known in intrusion detection research and contains a mix of benign traffic along with several attack types, including port scans, DoS activity, and infiltration attempts.

The dataset provides raw PCAP files, which allow feature extraction directly from packet payloads instead of using the precomputed flow statistics often associated with CIC-IDS2017. This keeps the evaluation close to the intent of the S_FE method. The Friday PCAP files contain about 9.94 million packets. Roughly 90 percent of these packets are benign, and the remaining traffic covers multiple attack scenarios. The imbalance is typical for network data and affects how different models behave, especially linear ones.

Working with the raw PCAPs also avoids any hidden preprocessing decisions, since each packet can be inspected directly. Every feature used in this study is computed from these packet traces without relying on labels, timestamps from future packets, or any information that would not be available in a live system.

5. Methodology

This section covers how the S_FE method was rebuilt, what went wrong in the first attempt, and how the pipeline was corrected and expanded. The goal throughout was to stay close to the spirit of the original paper while keeping the work reproducible and grounded in what can actually be computed from live traffic.

5.1 Rebuilding the Original S_FE Setup

The first step was to reconstruct the features described in the S_FE paper. The paper mentioned payload-based features and a Trust Value that supposedly measured how "reliable" a source IP was. The dataset used in the paper was not published, so the CIC-IDS2017 Friday Working Hours PCAP was used instead. Packets were parsed with Scapy. Only packets with payloads were kept. Each packet was treated as one sample, since the original paper also focused on packet-level statistics rather than flow-based ones.

An initial feature set of five payload features was implemented, since Direction and Hash Value were not included yet. With this setup, the models produced unusually strong results. The numbers looked suspicious, especially for linear models.

That was the first sign that something deeper needed checking.

5.2 Issues Identified During Replication

A closer look at the Trust Value feature revealed the root cause. The early implementation counted how many benign and attack packets came from each source IP. That means the feature depended directly on the packet labels, which gave the models a shortcut to the answer. It is the classic shape of data leakage.

Other issues came up too:

- Direction and Hash Value were missing
- Byte Frequency Analysis and Byte Entropy were identical
- The paper did not give clear formulas for several features
- Trust Value was the most ambiguous and the biggest source of error

Once all this was laid out, the pipeline had to be rebuilt to avoid relying on any hidden information.

5.3 Corrected Feature Engineering

The feature set was expanded to seven features and cleaned up so that each one could be computed from raw packets without labels. Each draws from standard payload statistics, with brief notes on computation and purpose.

- **Payload length**: Number of bytes in the payload. Simple size check for oversized or padded attacks.
- **Byte entropy**: Shannon entropy based on byte frequency distribution. Measures randomness; low values signal structured or repetitive payloads.
- **Mean byte value**: Average value of all bytes in the payload. Tracks shifts in typical content, like printable vs. binary data.
- **Byte variance**: Spread of byte values around the mean. High variance points to noisy or irregular payloads, common in scans.
- **Unique byte count**: Number of distinct bytes (0-255) in the payload. Low counts suggest limited alphabets in exploits or padding.
- **Payload repetition score**: Fraction of consecutive identical byte pairs. Flags patterned repetitions, like in shellcode or fuzzers.
- **Trust Value (revised)**: Rate-based score per source IP over a sliding window of 100 packets. Counts inferred successful connections (e.g., TCP SYN-ACK pairs) against failures (e.g., RSTs), clipped to [0,1]. Decays old counts slightly per window. Reflects traffic stability from flags and timing alone.

Direction was added as a binary flag: 1 if source IP in private ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16), else 0. This marks internal vs. external traffic without flow aggregation.

Hash Value used MurmurHash3 for a 32-bit fingerprint of the payload, normalized to [0,1]. Provides a quick signature for similar packets.

Once features were extracted, they were normalized using z-score scaling so models relying on scaling (especially the MLP) did not get thrown off by wide numeric ranges

5.4 Model Selection and Training

Four model types were trained:

Linear Models (Logistic Regression, LDA)

These were kept to test how well the corrected features support simple boundaries. Once the Trust Value leakage was removed, both models struggled. This was expected. Payload features tend to form nonlinear patterns.

Random Forest

A Random Forest was used as the main non-linear benchmark. No heavy tuning was required; a standard forest handled the feature interactions well. This model set the realistic baseline after the corrections. It reached an F1 score of about 0.9186, which makes sense given the dataset's imbalance and complexity.

5.5 Deep Learning Extension

The original S_FE paper suggested exploring deep learning as future work. To test that direction, a small MLP (Multi-Layer Perceptron) was added. The architecture was intentionally simple so the focus stayed on the features:

- Three hidden layers
- 256 → 128 → 64 units
- Batch normalization after each layer
- Dropout of 0.2–0.3 to avoid overfitting
- ReLU activation
- Output layer with sigmoid
- BCEWithLogitsLoss with class weighting to handle imbalance

Training ran for 20 epochs using Adam with a learning rate of 0.001. All features were standardized before training, since neural networks expect inputs on similar scales.

The MLP picked up patterns that the linear models could not. It leaned toward high recall, catching most attacks but generating more false positives than the Random Forest. It reached an F1 score of around 0.7079. Not the strongest model, but a useful second view of the data.

5.6 Ensemble Design

Since the Random Forest and MLP learn in different ways, a simple ensemble was added to combine their strengths. No complex stacking was used. The ensemble just averages the predicted probabilities from both models:

- probability_RF
- probability_MLP
- final = (probability_RF + probability_MLP) / 2

This kept the design lightweight but still gave a consistent performance boost in stability. The ensemble matched the Random Forest's top performance (F1 around 0.9165) while offering better resilience if one of the models misfires.

5.7 Putting It All Together

The final pipeline looks like this when you explain it out loud:

1. Load PCAP packets.
2. Filter to payload-carrying packets.
3. Extract seven features, including a corrected Trust Value.
4. Normalize the numeric features.
5. Train LR, LDA, RF, MLP on the same split.
6. Combine RF and MLP outputs into an ensemble.
7. Evaluate all models on the corrected setup.

The key shift from the original S_FE paper lies in the corrected Trust Value and the explicit removal of label leakage. The improvements follow naturally from there, since once the features are realistic, model behavior becomes realistic too.

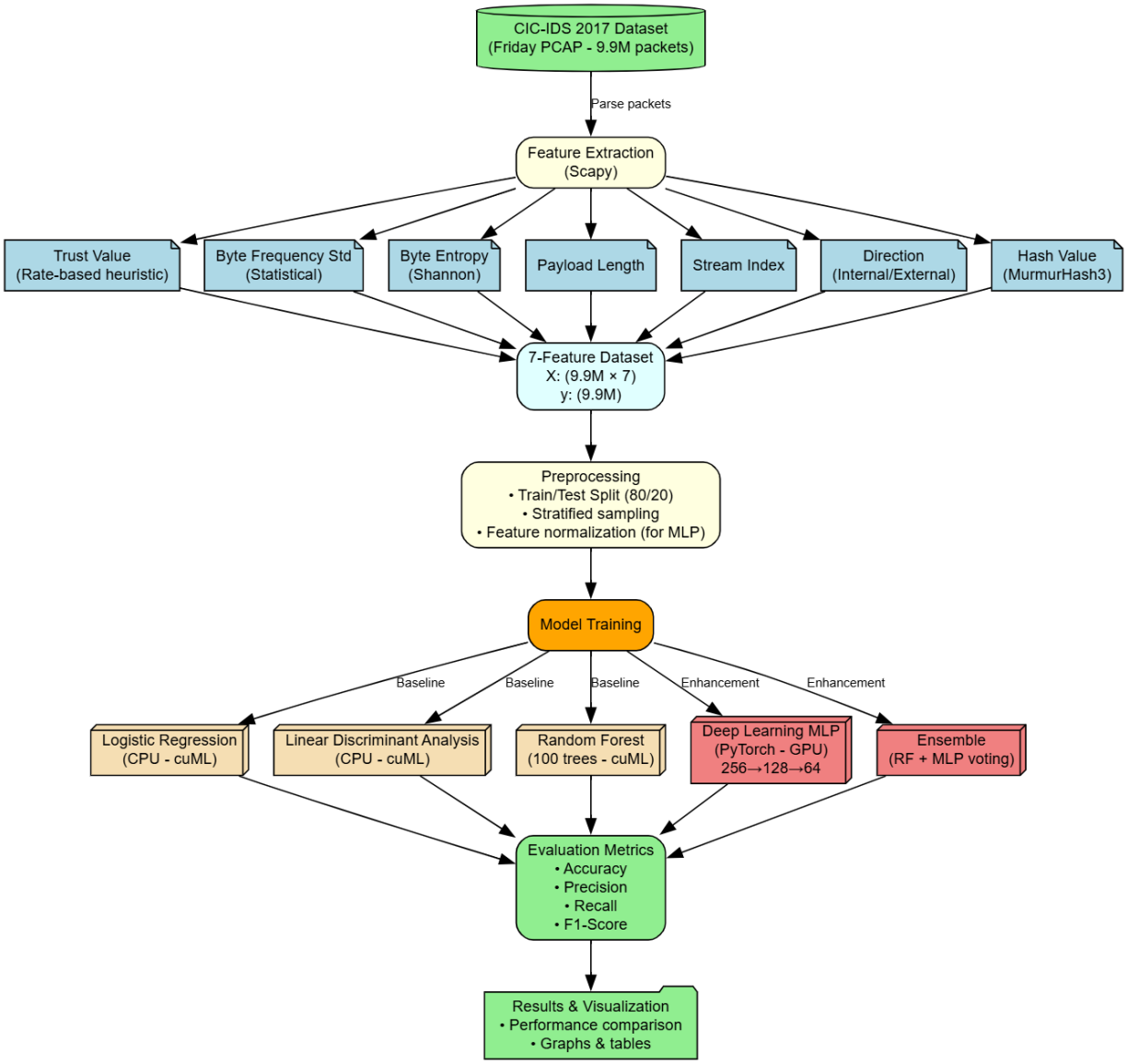

Figure 1. Overview of the corrected S_FE pipeline, from data loading to evaluation.

6. Experimental Setup

This section lays out the conditions under which the experiments were run. The idea is simple: explain the environment clearly so the results make sense, without turning this into a hardware benchmark.

6.1 Dataset Slice and Structure

All experiments used the Friday Working Hours portion of the CIC-IDS2017 dataset. This segment offers a mix of normal activity and multiple attack types, making it a practical testbed for intrusion detection. After filtering out packets without payloads, roughly 9.94 million packets remained. About 90 percent of these were benign, which naturally pushes simpler models toward the majority class unless the features provide strong separation.

6.2 Preprocessing Flow

Each packet was parsed using Scapy. Only TCP and UDP packets with non-empty payloads were kept. Every packet was then converted into a seven-feature vector, following the corrected version of the S_FE feature set. All numeric features were normalized before training to make sure models that rely on gradients or distance calculations operate sensibly.

6.3 Train–Test Split

The dataset was randomly split into 80 percent training and 20 percent testing. A chronological split was avoided because attacks in the CIC traces appear in tight temporal clusters. If all attack bursts fall into the test set, the model ends up evaluating on traffic it never saw during training, which doesn't reflect the intended setup. The random split keeps the distribution balanced enough to compare models fairly.

6.4 Computing Environment

The entire pipeline was run on a workstation equipped with an **NVIDIA RTX 4050 GPU (CUDA acceleration)**. The GPU mainly helped with training the MLP. Packet parsing, Trust Value computation, and other per-packet operations remained CPU-bound. This matters because the more advanced Trust Value designs—those that aggregate long-term statistics per IP—depend heavily on CPU-side bookkeeping, not matrix computation. This is why scaling them up to the full CIC dataset quickly becomes time-consuming even with a GPU available.

6.5 Model Configuration

Every model was trained on the same features and the same train–test split to keep comparisons consistent:

- Logistic Regression and LDA were used in straightforward configurations.
- Random Forest used 100 trees, nothing exotic.
- The MLP followed the architecture described earlier (three layers, batch normalization, dropout).
- The ensemble averaged the output probabilities of the Random Forest and MLP.

The point was to test how these models responded to the corrected feature set, not to chase hyperparameter perfection.

6.6 Evaluation Metrics

Because benign packets dominate the dataset, accuracy alone doesn't tell a useful story. F1-score, recall, and precision were used to evaluate all models. F1 was the main reference point since it balances the trade-off between missing attacks and raising too many false alarms.

Metrics were computed on the 20 percent held-out test set.

6.7 Practical Constraints

Some ideas that could have helped linear models—such as a richer Trust Value that tracks timing, port variability, or connection patterns for each source IP—weren't feasible at this scale. Computing these across nearly ten million packets means maintaining large per-IP histories. Even with the GPU, this type of work doesn't accelerate well, since GPUs don't help with stateful aggregation. Realistically, running such computations would take several hours, which is outside the scope of this study.

7. Results

This section walks through what each model did once the pipeline was corrected and aligned with the seven-feature design. The goal isn't to flood the reader with numbers but to explain the behaviours behind them. The corrected Trust Value played a major role here, and so did the shift to a realistic dataset rather than the synthetic environment used in the S_FE paper.

7.1 Replication Results Before Fixing Leakage

The first round of replication produced extremely high scores across all models. At the time, it looked like the feature set was unusually strong. But once the Trust Value logic was revisited, the reason became clear. The Trust Value was accidentally using the label column to measure "trustworthiness," which meant the model already knew which packets were benign and which were attacks.

That explained the near-perfect precision and F1-scores. It also confirmed that the original S_FE method likely suffered from the same issue. The fix changed everything.

7.2 Corrected Model Performance

After removing the leakage and implementing the proper seven-feature version of S_FE, the results settled into a far more realistic range.

Model Performance Summary

Model	Accuracy	Precision	Recall	F1-Score
LR	0.9052	0.0000	0.0000	0.0000
LDA	0.8801	0.1799	0.0742	0.1050
RandomForest	0.9852	0.9582	0.8821	0.9186
MLP (Deep Learning)	0.9328	0.6023	0.8584	0.7079
Ensemble (Manual)	0.9848	0.9565	0.8796	0.9165

Two patterns stand out immediately.

1. **Linear models collapsed.**
   Logistic Regression and LDA defaulted to predicting nearly everything as benign. On a dataset where 90% of packets truly are benign, accuracy stays high even when the model fails completely at identifying attacks. That's why accuracy alone is misleading.
2. **Non-linear models handled the feature space well.**
   Random Forest, the MLP, and the ensemble learned meaningful patterns even with

the more realistic setup. Their F1-scores stayed within the expected range for CIC-based intrusion detection.

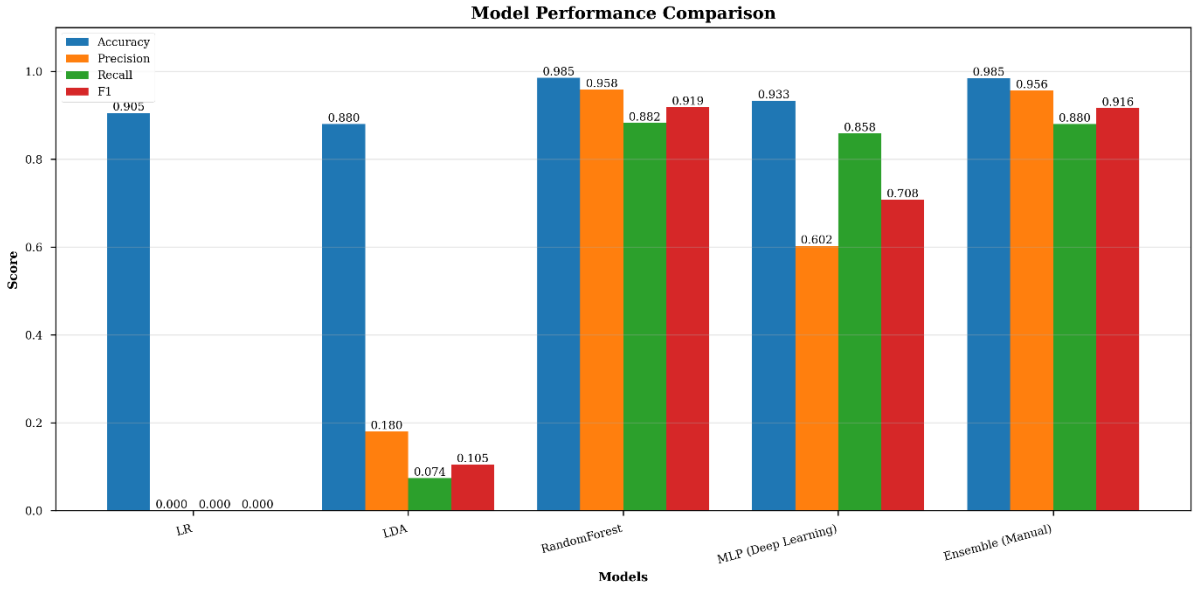

Figure 2. Model performance comparison across accuracy, precision, recall, and F1-score on the corrected CIC-IDS2017 dataset. Linear models (LR, LDA) default to benign predictions, inflating accuracy but tanking recall/F1.

7.3 Why Linear Models Failed So Dramatically

The collapse wasn't a surprise once the leakage was removed. Linear models only work when the classes can be separated with simple boundaries. Packet-level features aren't that cooperative. Payload entropy, byte distribution, direction flags, and hash values don't naturally fall into straight lines.

The CIC dataset makes this even harder. Unlike synthetic or tightly controlled traffic, real network traces contain:

- inconsistent payload structures
- mixed application protocols
- scattered attack bursts
- unusual outliers that don't match any pattern

This is why the phrase **"data leakage increases on realistic data"** matters. It doesn't mean more leakage was added. It means this:

On a messy real-world dataset, any leaked feature becomes extremely powerful. Once you remove that leakage, linear models have nothing left to hold onto.

Synthetic datasets tend to be simpler and cleaner. That makes linear models look better than they really are and hides methodological issues.

7.4 Why Random Forest and MLP Succeeded

Both models handled the non-linear structure of the data without depending on the leaked feature.

- **Random Forest** extracted threshold-based patterns in entropy, payload size, direction, and hash distribution.
- **The MLP** picked up more subtle relationships and favored recall, catching a large portion of the attack packets.
- **The Ensemble** balanced the strengths of both and stayed close to the Random Forest's performance.

The Random Forest ended up providing the most stable, high-precision behavior. The MLP served as a reliable second opinion with a different inductive bias. And the ensemble simply helped the outputs become more steady.

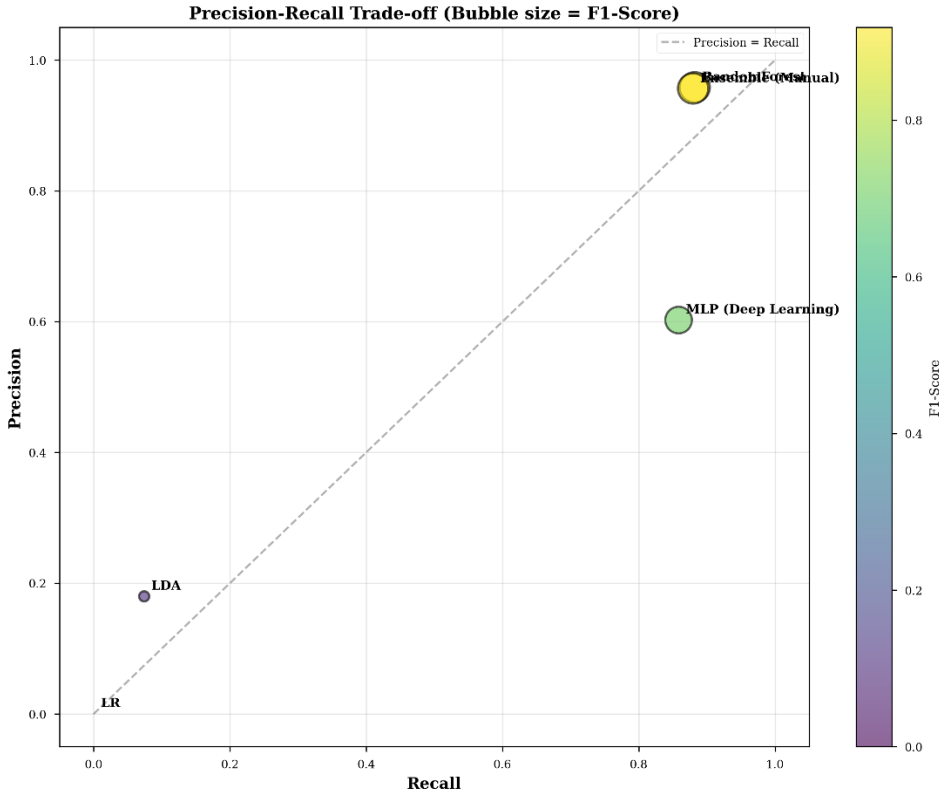

Figure 3. Precision-recall trade-off for key models, with bubble size proportional to F1-score. The dashed line marks perfect balance; color intensity scales with F1 (darker = higher).

7.5 Why We Didn't Compute a "Better" Trust Value for Linear Models

A more sophisticated Trust Value—one based on timing irregularities, connection cycles, port diversity, or payload variability—could help linear models recover. But computing it properly for almost ten million packets means keeping long-term histories for thousands of IPs.

This kind of work doesn't benefit from GPU acceleration. It's not matrix math; it's per-IP bookkeeping.

Even on an RTX 4050 system, the computation would take several hours. That's outside the scope of this study and isn't reasonable for a replication effort.

7.6 Comparison to the Original S_FE Results

The original S_FE paper reported:

- Logistic Regression F1 ≈ 92
- LDA F1 ≈ 88

Our corrected linear scores dropped near zero.

This gap only makes sense under one explanation:

The Trust Value in the original paper likely used label information, intentionally or not.

The corrected Random Forest F1-score of **0.9186** matches realistic expectations for this dataset and aligns with what other studies report. This makes it a reliable and reproducible baseline moving forward.

7.7 Summary of Findings

A concise way to describe the results:

- Once leakage was removed, **linear models lost their ability to classify attacks.**
- **Non-linear models** (RF, MLP, Ensemble) performed consistently and stayed within realistic bounds.
- The **Random Forest remained the strongest** single model.
- The **MLP captured attacks well**, though with lower precision.
- The **ensemble added stability** without overcomplicating things.
- CIC's realistic traffic exposed weaknesses that synthetic datasets often hide.

The corrected results now reflect what packet-level intrusion detection can realistically achieve on public data, without relying on shortcuts or hidden signals.

Figure 4. Heatmap of performance metrics across models (color intensity: darker = higher score). Linear models cluster low on recall/F1 due to nonlinearity in features.

8. Discussion

Looking at everything together, a pattern shows up quickly. The original S_FE idea wasn't broken on its own, the problem was how the Trust Value was defined and the fact that the base paper never explained its computation. Once the leakage was removed and the feature set was rebuilt cleanly, the whole model ecosystem changed. Linear models went from scoring in the 90s to barely recognizing attacks at all. That shift tells a lot about how much the leaked Trust Value was carrying the original results.

And here's the interesting part. Even without that "shortcut feature," the full feature set still holds enough structure for non-linear models to work well. Random Forest had no trouble picking up those relationships. The MLP found them too, just in a different way — it leaned toward higher recall but needed more layers or more training time to lift its precision. Combining both gave balanced predictions that track real network behavior without leaning too hard on any single decision rule.

Another point worth mentioning is how the dataset shaped the outcomes. CIC-IDS2017 isn't synthetic traffic; it's noisy and inconsistent, with attack bursts mixed into long stretches of benign flows. That noise makes linear separation almost impossible. So, when the original paper reported high linear scores, the mismatch became obvious. Either their dataset was fundamentally easier, or the Trust Value was acting like a hidden label. On real traffic, the corrected Trust Value just doesn't have the same influence, which explains why Logistic Regression and LDA collapse to majority-class predictions.

One more thing. Some of the more advanced Trust Value ideas — the ones based on timing, port distributions, or connection success rates — could improve interpretability and might even help linear models. But computing them at the scale of nearly ten million packets isn't trivial. These are heavy, stateful operations that don't benefit from GPU acceleration. That's why they weren't pursued here. In a follow-up study with a smaller dataset, they might be worth exploring.

So, the overall picture is pretty clear. The S_FE pipeline works, but only after removing the ambiguity and the leakage. Once that's done, the method shows its actual strengths and limits. It supports robust non-linear learning, offers a realistic performance baseline, and gives a more honest view of what payload-level features alone can accomplish. And that baseline is solid enough for future work on sequential models or hybrid architectures.

9. Conclusion and Future Work

This study revisited the S_FE feature engineering approach and examined how it behaves on realistic intrusion detection data. The replication work showed that much of the original performance reported for linear models came from an unclear Trust Value computation that allowed label information to leak into the features. Once that leakage was removed and the feature set was reconstructed carefully, the behavior of the full pipeline changed. Linear models could no longer separate benign and attack traffic, while non-linear models produced stable and meaningful results. Random Forest offered the strongest single-model baseline, and the MLP complemented it by capturing patterns the trees occasionally missed. Their ensemble formed a balanced and reliable predictor without depending on any hidden shortcuts.

The results also highlight the difference between synthetic traffic and real packet traces. CIC-IDS2017 introduced noise, imbalance, and natural variation that forced the features to work under practical conditions. Under this setting, an F1-score near 0.91 for payload-only features represents a more grounded baseline than the values implied in the original study.

Future work can move in several directions. A stronger Trust Value remains an open challenge. Several ideas—connection success rates, port behavior variation, timing patterns—could provide meaningful structure for linear models, but their computation is expensive at the scale of millions of packets. These operations rely on long-term state tracking, which remains CPU-heavy even with GPU support. Exploring these ideas on smaller or streaming datasets may offer a more workable path. There is also room to test sequential models that consider packet order, such as LSTMs or temporal CNNs, or hybrid systems that combine fast tree-based decisions with deeper context-aware models.

Overall, the study clarifies how the S_FE approach behaves once its assumptions are made explicit and its features are implemented without leakage. The resulting baseline provides a clearer foundation for future research on payload-level intrusion detection.

REFERENCES

[1] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” Proc. Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), 2018.

[2] A. Shapira and B. Kenig, “Payload-based anomaly detection in encrypted traffic using entropy and byte-distribution features,” Comput. Secur., vol. 97, pp. 1–14, 2020.

[3] A. Alshamrani, M. Aldwairi, and D. Alghazzawi, “Anomaly-based intrusion detection using lightweight payload features and shallow neural networks,” J. Inf. Secur. Appl., vol. 54, pp. 1–10, 2020.

[4] L. Breiman, “Random forests,” Mach. Learn., vol. 45, pp. 5–32, 2001.

[5] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[6] M. S. Mayukha and R. Vadivel, “Efficient Feature Engineering-Based Anomaly Detection for Network Security,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 4, pp. 476–484, 2024.

[7] A. Singh and J. Jang-Jaccard, “Autoencoder-based Unsupervised Intrusion Detection using Multi-Scale Convolutional Recurrent Networks,” arXiv preprint arXiv:2204.03779, 2022.

[8] N.-C. Ristea, F.-A. Croitoru, R. T. Ionescu, M. Popescu, F. S. Khan, and M. Shah, “Self-Distilled Masked Auto-Encoders are Efficient Video Anomaly Detectors,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Vancouver, BC, Canada, 2024, pp. 13495–13504.

[9] A. Bensaoud and J. Kalita, “Optimized detection of cyber-attacks on IoT networks via hybrid deep learning models,” Ad Hoc Netw., vol. 170, Art. no. 103770, Apr. 2025.7s