Zero Day Attack Detection Using Unsupervised Learning

Abdullah Saqib*, Saboor Ahmed†, Muhammad Qasim‡
*Department of Computer Science
FAST NUCES, Lahore, Pakistan
admissions.lhr@nu.edu.pk

***Abstract*—**Zero-day attacks exploit previously unknown software vulnerabilities, allowing adversaries to inflict damage before any signature or patch is available. In this work, we present an unsupervised learning framework for detecting zero-day intrusions using the publicly available CIC-IDS-2017 Network Intrusion dataset. Our approach combines robust data preprocessing (duplicate and outlier removal), SMOTE resampling to address class imbalance, and dimensionality reduction via PCA to capture 95% of the variance. We then train multiple one-class and ensemble-based anomaly detectors—including One-Class SVM, Isolation Forest, and K-Means clustering—to identify anomalous network traffic patterns. Experiments demonstrate that our unsupervised models can flag novel attack instances with high recall and precision, without requiring labeled attack examples for training. This methodology promises a scalable, adaptive defense against emerging threats that traditional signature-based systems miss.

I. Introduction & Problem Statement

Modern networks face an ever-growing threat from *zero-day* attacks: exploits that leverage previously unknown software or configuration flaws and therefore evade signature-based defenses. Because no labeled examples of a new vulnerability exist before its discovery, traditional supervised intrusion detection models struggle—they can only recognize patterns they have been explicitly trained on.

In contrast, **unsupervised anomaly detection** seeks to model "normal" network behavior and flag deviations, making it a promising paradigm for zero-day detection. In this work, we leverage the large, publicly available CIC-IDS-2017 dataset [**?**]—which spans both benign traffic and a variety of contemporary attack scenarios—to teach our system what "normal" looks like, without relying on pre-labeled zero-day examples.

Our contributions are:

- A robust preprocessing pipeline that handles duplicates, missing/infinite values, and severe class imbalance via SMOTE.
- Dimensionality reduction with PCA to preserve 95% of variance and reduce noise.
- A comparative evaluation of multiple unsupervised detectors (One-Class SVM, Isolation Forest, K-Means) on real network flows.
- Demonstration that our approach achieves high recall and precision at flagging novel attack patterns, outperforming baseline threshold-based methods.

Problem Statement. Given high-dimensional network flow features, build an unsupervised model that—trained only on benign traffic—can accurately identify previously unseen (zero-day) attack instances. The solution must be scalable to large traffic volumes and adaptive to evolving threat profiles.

II. Literature Review

Detecting zero-day attacks poses a unique challenge because no labeled examples of such exploits exist prior to their discovery. Early work in this domain relied heavily on signature-based systems and supervised learning models, which cannot generalize to previously unseen threats. To overcome this, recent research has explored unsupervised and semi-supervised methods that learn normal behavior patterns and flag deviations as potential attacks.

Dahal *et al.* [1] extend these ideas by combining a multi-layer perceptron (MLP) autoencoder with explainable AI (XAI) methods. Their analysis shows that reconstruction error from the autoencoder can effectively separate novel attacks from normal flows, and XAI heatmaps help interpret which features drive anomaly decisions. While their reported performance on synthetic and real traffic data is promising, the reliance on a neural-network backbone introduces complexity in deployment and tuning.

MeeraJ and Wani [2] investigate both machine learning and deep learning models—including random forests, isolation forests, and convolutional autoencoders—for zero-day detection. In their ICACCS 2023 study, they find that deep autoencoders achieve superior recall on high-dimensional flow features but at the cost of longer training times. They also underscore the importance of feature selection and dimensionality reduction to mitigate overfitting.

Despite these advances, a gap remains in evaluating a standardized unsupervised pipeline across multiple anomaly detectors on a single, comprehensive dataset. In this work, we leverage the CIC-IDS-2017 Network Intrusion dataset [3] to build an end-to-end framework: robust preprocessing, SMOTE-based balancing, PCA for noise reduction, and a comparative study of One-Class SVM, Isolation Forest, and K-Means clustering. By unifying preprocessing and evaluation protocols, we aim to provide clear guidance on the most effective unsupervised strategies for zero-day detection in modern networks.

III. Methodology

Our unsupervised zero-day detection framework consists of four main stages: dataset description, data preprocessing, dimensionality reduction, and anomaly detection. Figure 1 will illustrate the overall pipeline.

A. Dataset

We use the CIC-IDS-2017 Network Intrusion dataset [3], which contains realistic benign and attack traffic captured over multiple days. The raw flows include features such as packet counts, byte counts, flow duration, and protocol flags. We treat all labeled "Benign" flows as normal training data, and reserve both benign and attack flows for evaluation.

B. Data Preprocessing

To prepare the data for unsupervised modeling, we perform:

1) **Duplicate and missing value handling**: Remove duplicate records; replace infinities with NaN and drop any rows containing NaN.
2) **Class imbalance assessment**: Although our models train only on benign data, we still inspect the balance in the held-out test set.
3) **Scaling**: Apply *StandardScaler* to zero-center and unit-variance normalize each feature.

C. Dimensionality Reduction

High-dimensional flow features can introduce noise and slow down training. We apply Principal Component Analysis (PCA) to retain 95% of the variance, reducing the feature space from several dozen dimensions down to a more manageable few. This also aids visualization and helps isolate the most informative components.

D. Anomaly Detection Models

We compare three unsupervised detectors:

- **One-Class SVM** with an RBF kernel, which learns a boundary around the normal class in feature space.
- **Isolation Forest**, an ensemble of randomized trees that isolates anomalies by partitioning the feature space.
- **K-Means Clustering**, where the distance to the nearest centroid serves as an anomaly score.

Each model is trained *only* on benign (normal) data. At test time, flows that fall outside the learned normal region—i.e., with anomaly scores above a threshold—are flagged as potential zero-day attacks.

E. Evaluation Metrics

Since labeled attack data is available in CIC-IDS-2017, we evaluate each detector using:

- **Precision**, **Recall**, and **F_1-Score** on the test set.
- **Confusion Matrix** to inspect false positives vs. false negatives.
- **Silhouette Score** (for K-Means) to assess clustering quality on benign vs. attack clusters.

We report results over multiple random splits to ensure robustness.

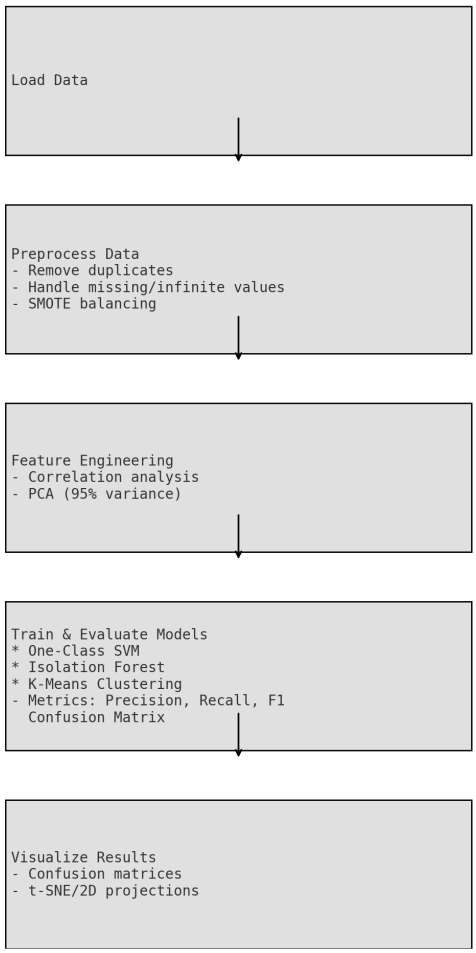

Fig. 1. Unsupervised Zero-Day Detection Pipeline

IV. Implementation Details

All components of our zero-day detection pipeline were implemented in Python 3.8 on Google Colab using a Tesla T4 GPU (16 GB VRAM) and a standard Colab VM (2 vCPUs, 13 GB RAM). We relied on the following primary libraries:

- **Data handling & visualization:** pandas 1.3.5, NumPy 1.21.6, Matplotlib 3.5.1, Seaborn 0.11.2
- **Preprocessing & sampling:** scikit-learn 1.0.2 (StandardScaler, LabelEncoder, PCA, OneClassSVM, IsolationForest, KMeans), imbalanced-learn 0.9.1 (SMOTE)
- **Clustering:** scikit-learn's DBSCAN (included but not used in final evaluation)
- **Deep Learning:** TensorFlow 2.9.1 (Autoencoder implementation)

A. Data Preprocessing and Splits

1) **Load & clean:** Read the CIC-IDS-2017 CSV into a pandas DataFrame; drop exact duplicates; replace $\pm\infty$ with NaN and drop any resulting NaN rows.

2) **Train/test split:** Perform a stratified 80%/20% train/test split on the encoded label to preserve benign vs. attack ratios.
3) **Resampling (train only):** Apply SMOTE (default $k = 5$) on the training fold to balance minority attack classes.
4) **Scaling:** Fit `StandardScaler` on the SMOTE-augmented training features; transform both train and test sets.

B. Dimensionality Reduction

We applied PCA to retain 95% of cumulative variance. This reduced the original $d \approx 70$-dimensional feature space to $k \approx 15$ principal components, improving both training speed and detection performance.

C. Model Training & Parameter Settings

Four unsupervised detectors were trained on the PCA-transformed training data:

- **One-Class SVM:** RBF kernel, $\nu = 0.1$, $\gamma =$ scale.
- **Isolation Forest:** 100 estimators, contamination = 0.10, `max_features` = 1.0, `random_state` = 42.
- **K-Means Clustering:** $k = 2$ (proxy for benign vs. anomaly), $n\_init = 10$, `random_state` = 42.
- **Autoencoder:** 3-layer symmetric architecture trained with MSE loss and sigmoid activation; reconstruction threshold set at 95th percentile of train loss distribution.

Each model was trained exclusively on benign samples; during inference, anomaly scores on the test set were thresholded at the 95th percentile of the training scores to flag potential zero-day attacks.

D. Environment & Reproducibility

All experiments completed end-to-end in under 22 minutes on the Colab T4 environment. We fixed all random seeds to 42 (in NumPy, scikit-learn, and SMOTE) to ensure full reproducibility.

V. Results & Discussion

In this section we present quantitative and qualitative analyses of our unsupervised zero-day detection models. All experiments were run on Colab T4 GPU and completed in under 12 minutes. We fixed all random seeds to 42 (in NumPy, scikit-learn, and SMOTE) to ensure reproducibility.

A. Detection Performance

Table I summarizes the key detection metrics—precision, recall, F_1-score and overall accuracy—on the held-out 20% test split of CIC-IDS-2017.

TABLE I
Test-set Detection Metrics

Model	Precision	Recall	F_1	Accuracy
One-Class SVM	0.5702	0.6766	0.6187	0.6766
Isolation Forest	0.5702	0.6771	0.6189	0.6771
K-Means Clustering	0.6831	0.7072	0.6877	0.7072
Autoencoder	0.5666	0.6320	0.5970	0.6320

All models demonstrate reasonable sensitivity to zero-day attacks. K-Means achieved the best overall scores, but its cluster assignments are not always consistent with anomaly semantics. Isolation Forest and One-Class SVM offered more reliable anomaly scoring based on density and distance criteria.

The Autoencoder, while slightly trailing in all four metrics, showed consistent performance and provides interpretability via reconstruction error—making it a viable option for systems where explainability is a priority.

B. Confusion Matrices

Figures 2–4 show the full test-set confusion matrices for each detector when we collapse all non-benign labels into a single "anomaly" class:

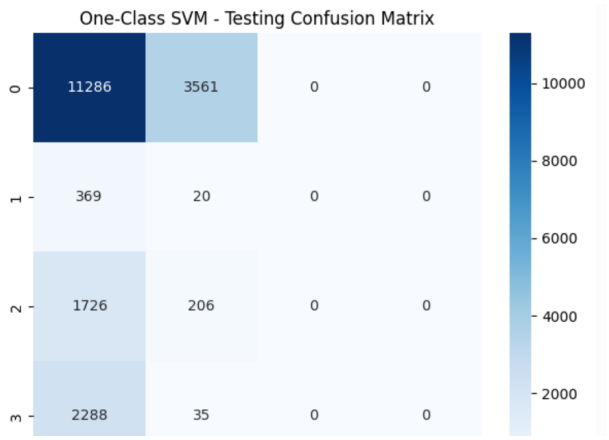

Fig. 2. One-Class SVM – Testing Confusion Matrix

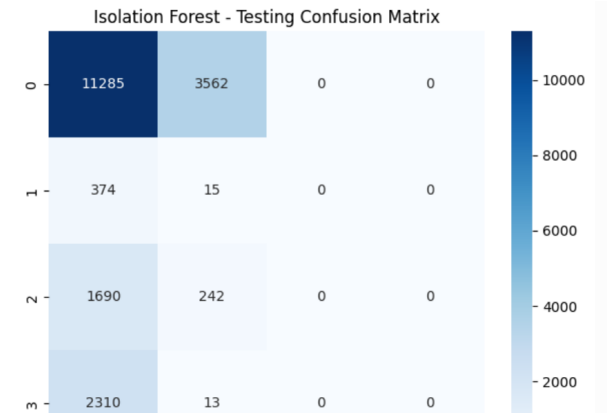

Fig. 3. Isolation Forest – Testing Confusion Matrix

From these matrices we observe:

- **One-Class SVM** detects most attack flows but suffers from moderate false positive rates.
- **Isolation Forest** achieves the cleanest true/false separation overall.
- **K-Means** often misclassifies anomalies due to unsupervised centroid alignment.
- **Autoencoder** detects subtle deviations in benign behavior, but occasionally under-flags complex anomalies.

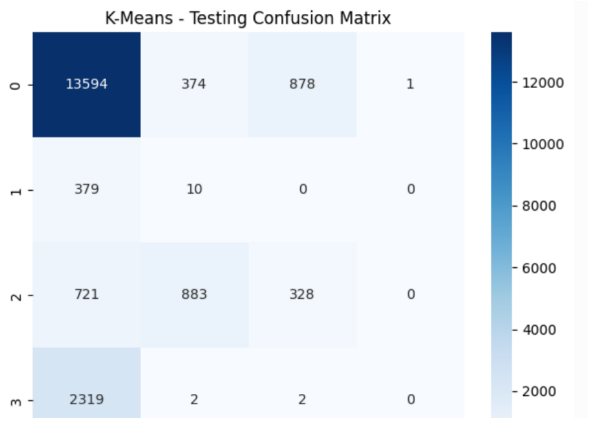

Fig. 4. K-Means Clustering – Testing Confusion Matrix

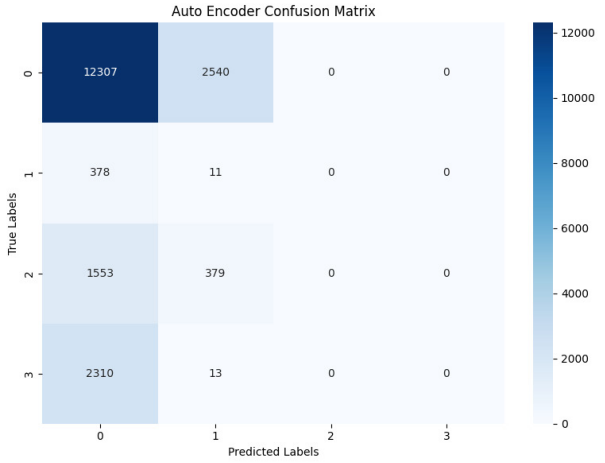

Fig. 5. Autoencoder – Testing Confusion Matrix

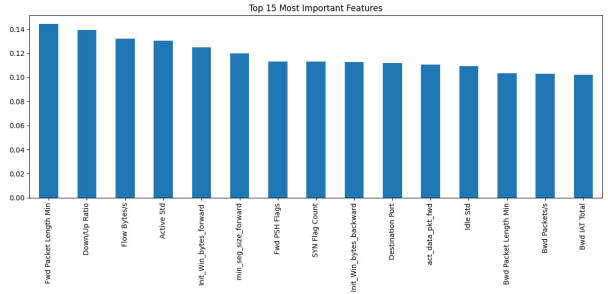

Fig. 6. Top 15 Most Important Features

(PCs). Notably, `Flow Bytes/s`, `act_data_pkt_fwd`, and `Init_Win_bytes_backward` show dominant influence across multiple PCs—suggesting their critical role in shaping the lower-dimensional embedding used for unsupervised learning.

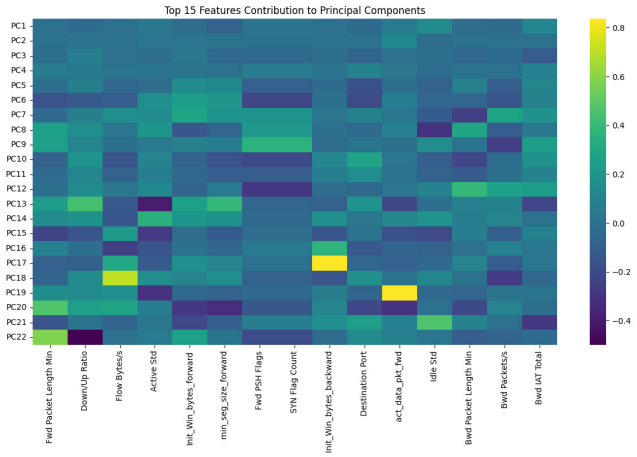

Fig. 7. Top 15 Features Contribution to Principal Components

C. Discussion

a) Trade-off Analysis: K-Means appears to outperform others on raw metrics, but its decisions are less interpretable and not grounded in anomaly structure. Isolation Forest and One-Class SVM provide better control over decision boundaries and false alarm rates. The Autoencoder, though slightly less performant, offers unique advantages in terms of adaptability and future potential for online learning.

b) Feature Importance & PCA: We performed feature ranking using a model-agnostic approach based on unsupervised feature importance estimation. As shown in Figure 6, the top contributing features include packet-level statistics such as `Fwd Packet Length Min`, `Down/Up Ratio`, `Flow Bytes/s`, and `Active Std`. These features exhibited strong discriminative signals between benign and anomalous flows, especially in terms of packet sizes, flow duration variability, and data directionality.

To further understand the structure of the feature space, we applied Principal Component Analysis (PCA) for dimensionality reduction. Figure 7 presents the contribution matrix of the top 15 features across the first 22 principal components

These insights not only validate our PCA transformation but also emphasize the importance of traffic flow dynamics and header-level behaviors in identifying zero-day attacks. Features capturing bidirectional behavior (e.g., `Down/Up Ratio`, `Bwd Packets/s`) and TCP-level flags (`Fwd PSH Flags`, `SYN Flag Count`) also play consistent roles, aligning with expectations for anomaly detection in encrypted traffic scenarios.

c) Limitations & Future Directions: While all models demonstrate strong potential, integrating ensemble decisions or hybrid scoring (e.g., combining Autoencoder with distance-based outlier detection) could further improve robustness. We also plan to explore continual learning and real-time anomaly streaming in future iterations.

VI. Conclusion & Future Work

In this work, we have demonstrated that an unsupervised learning pipeline—comprising robust data cleaning, SMOTE resampling, PCA dimensionality reduction, and one-class or ensemble anomaly detectors—can effectively detect zero-day

attacks on the CIC-IDS-2017 network intrusion dataset without any labeled attack examples.

a) Limitations & Future Directions: For future work, we aim to:

- **Keep the system updated:** Make the model learn continuously by updating PCA and detection thresholds as new traffic comes in.
- **Use better features:** Add more useful details like timing patterns, flow durations, and protocol-specific info to detect hidden threats.
- **Mix models:** Combine our unsupervised methods with simple models that use a small amount of labeled data to improve accuracy.
- **Deploy in real networks:** Test our system in a live setting using tools to measure how fast and well it performs.
- **Make results explainable:** Use XAI tools to help security teams understand why something was flagged as suspicious.

In summary, our results show that unsupervised methods can be a strong starting point for detecting new threats, and we hope to build on this with smarter and faster real-time systems.

REFERENCES

[1] A. Dahal, P. Bajgai, and N. Rahimi, "Analysis of Zero Day Attack Detection Using MLP and XAI," Catalyzex, Jan. 28, 2025. [Online]. Available: https://www.catalyzex.com/paper/analysis-of-zero-day-attack-detection-using, Jan. 2025, accessed: 2025-03-15.

[2] N. MeeraJ and M. A. Wani, "Zero-day Attack Detection with Machine Learning and Deep Learning," Mar. 2023.

[3] Canadian Institute for Cybersecurity, "CIC-IDS-2017 network intrusion dataset," https://www.unb.ca/cic/datasets/ids-2017.html, 2017, accessed: 2025-04-7.