

Anomaly Detection in Network Security Logs Using Machine Learning

Dr. Manjula H Nebagiri

Professor, Dept. of CSE
Shridevi Institute of Engineering and Technology,
Tumkur, Karnataka, India.
Visvesvaraya Technological University, Belagavi,
Karnataka, India.

Manasa H C

Dept. of CSE
Shridevi Institute of Engineering and Technology,
Tumkur, Karnataka, India.
Visvesvaraya Technological University, Belagavi,
Karnataka, India.

Kavyashree H T

Dept. of CSE
Shridevi Institute of Engineering and Technology,
Tumkur, Karnataka, India.
Visvesvaraya Technological University, Belagavi,
Karnataka, India.

Manya H N

Dept. of CSE
Shridevi Institute of Engineering and Technology,
Tumkur, Karnataka, India.
Visvesvaraya Technological University, Belagavi,
Karnataka, India.

Manyashree H T

Dept. of CSE
Shridevi Institute of Engineering and Technology, Tumkur, Karnataka, India.
Visvesvaraya Technological University, Belagavi, Karnataka, India.

ABSTRACT

Network devices produce large volumes of security logs, and checking them by hand is slow. Signature-based tools work well against known attacks but can miss new ones. This paper builds an anomaly detection system for network security logs around the Isolation Forest algorithm, which is trained on normal records only and does not need labelled attacks. The system covers data loading, scaling, model training, anomaly scoring, performance evaluation, and a Flask web dashboard.

Each record has three features: bytes sent, session duration, and failed logins. The experiments use a synthetic dataset of 1,000 records (950 normal and 50 anomalous). The models are trained on 760 normal records and tested on 240 records, made up of 190 normal records and all 50 anomalous ones. Isolation Forest is compared with Local Outlier Factor (LOF) and One-Class SVM using accuracy, precision, recall, and F1-score. All three models find every anomaly (recall of 1.00). Isolation Forest reaches an accuracy of 0.950 and an F1-score of 0.893 with 12 false alarms, LOF is slightly higher (F1-score 0.901), and One-Class SVM is slightly lower (0.870). The dashboard adds a sign-in page with approximate login location, and a risk score, risk

level, possible incident type, and recommended action for each analysed log entry.

Index Terms— Anomaly detection, network security, machine learning, Isolation Forest, security logs.

I. INTRODUCTION

A. Background

Firewalls, routers, intrusion detection sensors, authentication servers, and web servers all write logs about network activity. A log entry may record who connected, which protocol and port were used, how long the session lasted, how many bytes were moved, and whether the attempt succeeded. These logs help security teams notice suspicious behaviour and keep services reliable. Most entries describe normal use, and only a small share points to attacks such as port scans, brute-force logins, denial-of-service floods, or data theft. Because attacks are rare and the volume is large, manual review is slow and easy to get wrong, and logs from different systems often use different formats.

B. Problem Statement

Traditional intrusion detection depends on signatures of known attacks. These tools need constant updating, and they can miss new attacks or old attacks that were slightly modified. Supervised machine learning can learn from labelled normal and attack traffic, but building a large and reliable labelled dataset is difficult.

Attack types also change over time, so labelled examples of new attacks may not exist. In addition, normal traffic usually far outnumbers attack traffic, which makes it hard for a supervised model to learn the rare class [13]. A method that learns only what normal traffic looks like, and then flags records that differ from it, is therefore a practical alternative.

C. Motivation

Unsupervised anomaly detection suits this problem because it does not need a label for every record. Instead of matching traffic against known attacks, it marks records that differ from the normal pattern. Isolation Forest [1] is a good choice for logs because of a simple idea: unusual records are easier to separate from the rest of the data, so they need fewer random splits. This keeps the method fast and easy to apply. We therefore use Isolation Forest as the main model and compare it with two other unsupervised methods.

D. Objectives

The aim of this work is to build and test a machine-learning system that detects abnormal activity in network security logs. The specific objectives are:

1. To prepare network log records for modelling by handling missing values and scaling the numeric features.
2. To describe each record with three features: bytes sent, session duration, and failed logins.
3. To train an Isolation Forest on normal records only, so that no labelled attack data is needed for training.
4. To classify records as normal or anomalous and to give each analysed entry a risk score and risk level.
5. To evaluate the system using accuracy, precision, recall, and F1-score.
6. To compare Isolation Forest with Local Outlier Factor (LOF) and One-Class SVM under the same conditions.
7. To build a web dashboard where a user can sign in, enter a log record, and see the result with a possible incident type and a recommended action.

E. Contributions

The main contributions of this work are:

1. A pipeline from data loading and scaling to model training and scoring, implemented in Python with scikit-learn.

2. An unsupervised detector that learns from normal records only and detected all 50 anomalies in the test data.
3. A comparison of Isolation Forest, LOF, and One-Class SVM on the same data split, reported with a confusion matrix and four metrics.
4. A Flask web dashboard with sign-in, approximate login location, risk scoring, incident labelling, and charts.
5. A discussion of the limits of the experiment, mainly that the data is synthetic and that only one split was used.

II. RELATED WORK

A. Traditional Network Intrusion Detection

Early intrusion detection systems compared network traffic with a database of known attack patterns, called signatures. This works well for attacks that are already known. It fails for zero-day attacks and for variants that change just enough to avoid a signature, and the database must be updated again and again.

B. Machine Learning for Anomaly Detection

Machine learning lets a system learn patterns from data instead of relying only on hand-written rules, and anomaly detection in general is surveyed in [3]. It can react to attacks that were not seen before and adapt as traffic changes. Its success still depends on good features, careful validation, and control of false alarms.

C. Supervised Learning Approaches

Supervised classifiers such as decision trees, random forests [14], and support vector machines have been trained on labelled intrusion data and often report high accuracy. They need representative labelled data, and they may fail on attack types that were missing from the training set.

D. Unsupervised Learning Approaches

Unsupervised methods such as clustering, autoencoders, One-Class SVM [5], and LOF [4] avoid the need for labels. They look for records that do not fit the structure, spread, or density of normal traffic. They suit cases where labelled attacks are rare, although some of them become costly on large log volumes.

E. Isolation Forest-Based Detection

Isolation Forest [1] separates observations by splitting the feature space at random, again and again. Records that are rare and different are isolated after fewer splits, and the average path length is turned into an anomaly score [12]. Unlike distance-based and density-based methods, it does not compute distances between records. Extended versions of the idea have also been proposed [9].

F. Research Gap

Many studies test detection algorithms on public intrusion datasets [2], [6], [8]. Two points get less attention. The first is how the whole system is put together, from raw records to something an analyst can use. The second is a fair comparison of several unsupervised methods on identical data, scaling, and settings. This work addresses both by building a small working system with a web dashboard and by comparing three methods under the same conditions.

III. PROPOSED METHODOLOGY

A. System Overview

The system takes network log records with three numeric features. The features are scaled, and the scaled values are passed to an Isolation Forest, which gives each record an anomaly score. The score is turned into a normal or anomalous label. The results are used for evaluation and are also shown in a web dashboard. The detection part is kept separate from the data loading and the dashboard, so the model can be retrained or replaced without changing the other parts.

B. Data Collection

In real networks, log data comes from firewalls, intrusion detection sensors, and connection monitors. This work uses a synthetic dataset instead, because it gives a known label for every record and lets the experiment be repeated. The data is created by the code with a fixed random seed (42) and saved as network_traffic.csv.

C. Dataset Description

The dataset has 1,000 records: 950 normal and 50 anomalous. Every record has three numeric features: bytes_sent, session_duration (in seconds), and failed_logins. For normal records, bytes sent follow a normal distribution with a mean of 500 and a standard deviation of 50, session duration follows a normal distribution with a mean of 70 and a standard deviation of 10, and failed logins follow a Poisson distribution with a mean of 2. For anomalous records, the means are 3,000 bytes (standard deviation 500), 250 seconds (standard deviation 40), and 10 failed logins. The label is used only for evaluation. Because the data is synthetic, the results show how the methods work when abnormal activity is clearly different from normal activity, and they cannot be taken as results on real traffic. Fig. 1 shows the distribution of the three features. Normal records lie between about 320 and 660 bytes and between 40 and 100 seconds. A few records have much larger values, up to about 4,260 bytes and 315 seconds, and these are the anomalous ones.

Network Security Feature Distribution

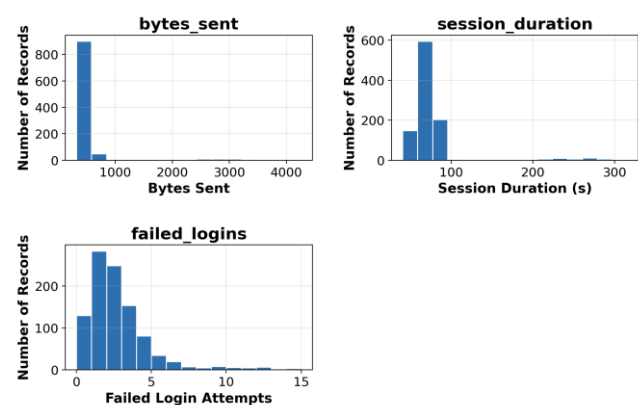


Fig. 1. Distribution of core network-log features in the dataset.

D. Data Preprocessing

The preprocessing stage is kept simple because all three features are numeric. The dataset contains no duplicate or missing values, but the code still fills any missing value with the feature median as a safeguard. The features are then standardised to zero mean and unit variance, using a scaler fitted on the training records only, so that bytes sent, which has much larger values, does not dominate the model. Fig. 2 shows a boxplot of the three features before scaling. The features have a long right tail, with many points above the upper whisker. These points are not removed, because they are the anomalous records that the system is designed to detect.

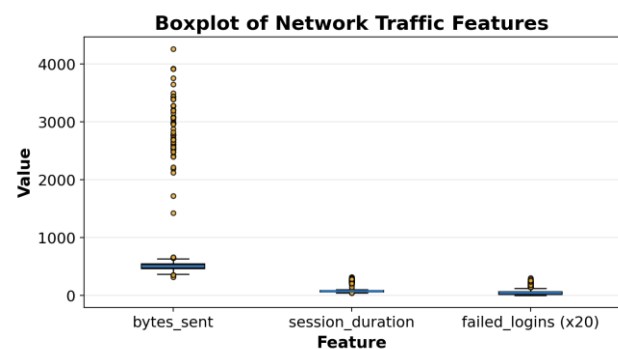


Fig. 2. Boxplot of network traffic features (bytes_sent, session_duration, failed_logins) prior to scaling. The failed_logins values are multiplied by 20 so that they are visible on the same axis.

E. Feature Set

Three features are used: bytes sent, session duration, and number of failed logins. They describe the data volume, the length, and the login behaviour of a session, which are common signs of data theft and brute-force attempts. No automatic feature-selection step is applied. Fig. 3 shows the correlation between the three features and the label. All three features correlate positively with the label (0.97 for bytes sent, 0.95 for session duration, and 0.76 for failed logins), and bytes sent and session duration are strongly correlated with each other (0.94).

The label is used only for this figure and for evaluation, not for training.

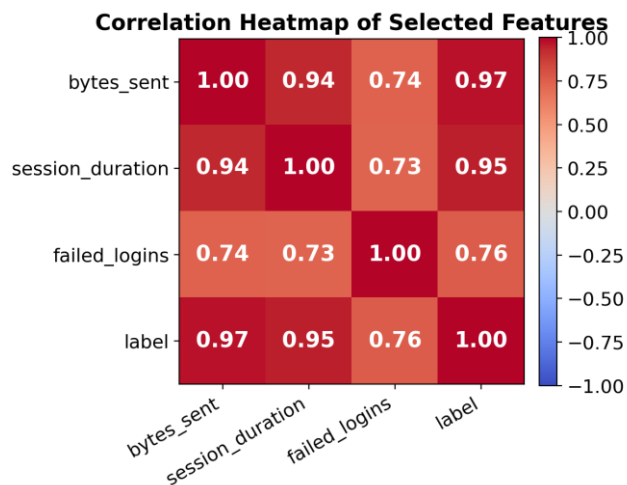


Fig. 3. Correlation heatmap of the three features and the traffic label.

F. Isolation Forest Algorithm

Isolation Forest builds many random trees. At each node of a tree, one feature is picked at random, and a split value is picked at random between the smallest and the largest value of that feature. The data is split again and again until each record is alone or a depth limit is reached.

Records that are very different from the rest are separated after only a few splits, so they have short average path lengths across the trees. The anomaly score is calculated from these path lengths, and a shorter path means a higher chance that the record is anomalous [12].

G. Anomaly Detection Process

At detection time, each scaled record goes through the trained model and receives an anomaly score. The score is compared with a threshold to give a normal or anomalous label. The threshold comes from the contamination parameter, which is the expected share of anomalies. Here it is set to 0.05, which matches the 5% share of anomalous records in the full dataset, and the model uses 150 trees. The predicted labels are then compared with the true labels to calculate the metrics.

IV. SYSTEM ARCHITECTURE

End-to-End System Architecture

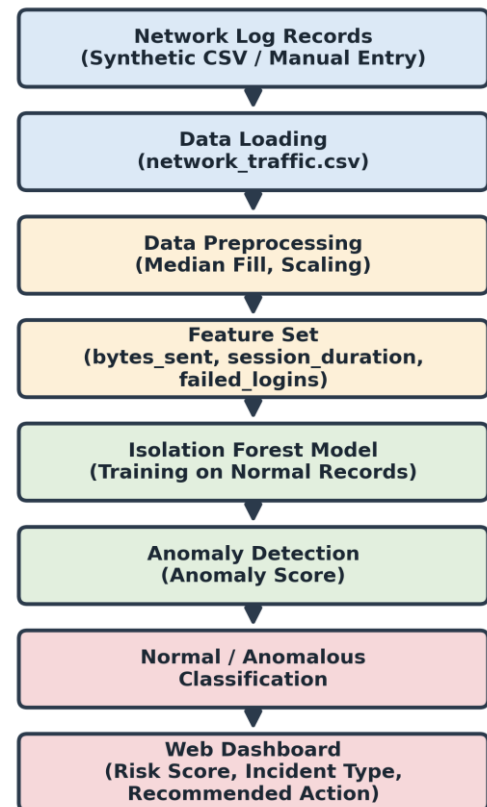


Fig. 4. End-to-end system architecture for log-based anomaly detection.

The system is built as a straight pipeline of stages (Fig. 4). Log records are loaded from the CSV file, cleaned and scaled, and then scored by the Isolation Forest. The resulting labels are shown on a Flask-based web dashboard, which also calculates a risk score, a risk level, a possible incident type, and a recommended action for every analysed entry. Presenting alerts to analysts in this way is similar to the role of SIEM tools [11].

Keeping the stages separate also makes testing easier. If the output looks wrong, the intermediate result of the scaling step or the feature step can be checked on its own to see whether the problem is in the data or in the model.

V. IMPLEMENTATION

A. Software Requirements

The following software was used:

- Python 3.11
- Scikit-learn for Isolation Forest, Local Outlier Factor (LOF), and One-Class SVM
- Pandas and NumPy for data handling and numerical work

- Matplotlib for the figures in this paper
- Flask and Flask-CORS for the web dashboard and its API, with Chart.js for the dashboard charts
- The Requests library for the approximate IP-based login location

B. Hardware Requirements

A normal computer is enough to run the system:

- A standard computer with a multi-core CPU
- A few gigabytes of RAM
- A small amount of disk space for the dataset and the code
- A web browser to open the dashboard

C. Algorithm Implementation

Three unsupervised models are used: Isolation Forest, Local Outlier Factor (LOF), and One-Class SVM. All three are trained on the same scaled normal records and tested on the same evaluation set.

1) Isolation Forest: The model builds 150 random isolation trees and uses a contamination value of 0.05 and a random seed of 42. Records that need few splits to isolate receive high anomaly scores and are labelled anomalous.

2) Local Outlier Factor: LOF compares the local density around a record with the density around its neighbours. A record in a much sparser area than its neighbours gets a high outlier score and is labelled anomalous [4]. It uses 20 neighbours and a contamination value of 0.05, in novelty mode so that it can score records that were not in the training data.

3) One-Class Support Vector Machine: One-Class SVM learns a boundary around the normal training records. Records outside this boundary are labelled anomalous [5]. It uses an RBF kernel, the default scale setting for gamma, and $\nu = 0.05$.

The predictions of the three models are compared with the true labels using accuracy, precision, recall, and F1-score.

D. Training and Testing

The three models are trained on normal records only. Of the 950 normal records, 80% (760 records) are used for training and the remaining 20% (190 records) are kept for testing. The evaluation set consists of these 190 normal records and all 50 anomalous records, so it has 240 records. Because this is an unsupervised setting, no anomalous record is seen during training, and the labels are used only afterwards to calculate the metrics. The same random seed (42) is used for the split and for the models, and the scaler is fitted on the training records only.

The prediction of each model is compared with the true label of every record in the evaluation set. Note that anomalies make up 20.8% of this evaluation set (50 of 240 records), which is higher than their 5% share of the full dataset.

E. Web Dashboard Implementation

The dashboard is a Flask application with a browser front end. On the sign-in page (Fig. 9) the user enters a name and a username. The server records the login time and finds an approximate location from the IP address using an IP geolocation service; for a local address such as 127.0.0.1 it shows a local network instead. After sign-in, the user can type the three values of a network log entry and press a button to analyse it. For this live analysis, an Isolation Forest is trained on all 950 normal records, and an entry is marked as an anomaly if the model flags it or if a simple rule is triggered (8 or more failed logins, 2,500 or more bytes sent, a session of 180 seconds or more, or a risk score above 60). The risk score is a weighted sum of the three values: bytes sent divided by 3,000 and multiplied by 35, session duration divided by 250 and multiplied by 25, and failed logins divided by 10 and multiplied by 40, limited to 100. A score up to 30 is normal, up to 60 is suspicious, up to 80 is high risk, and above 80 is critical. Each level has a recommended action, and the possible incident type, for example a brute-force attempt or data exfiltration, is chosen by rules on the feature values. The dashboard also keeps a list of recent detections and draws charts of the results.

VI. EXPERIMENTAL RESULTS

A. Dataset Results

Table I shows the record counts of the dataset used in the experiments, and Fig. 5 shows them as a chart.

The 95:5 ratio of normal to anomalous records was chosen to reflect real networks, where attacks are only a small part of all connections.

Category	Record Count
Total Records	1,000
Normal	950
Anomalous	50

Table I. Dataset composition.

Table I describes the full dataset. After the split in Section V-D, the training set has 760 normal records, and the evaluation set has 190 normal and 50 anomalous records (240 in total).

B. Number of Normal and Anomalous Records

Fig. 5 shows the number of normal and anomalous records in the full dataset: 950 normal records and 50 anomalous ones.

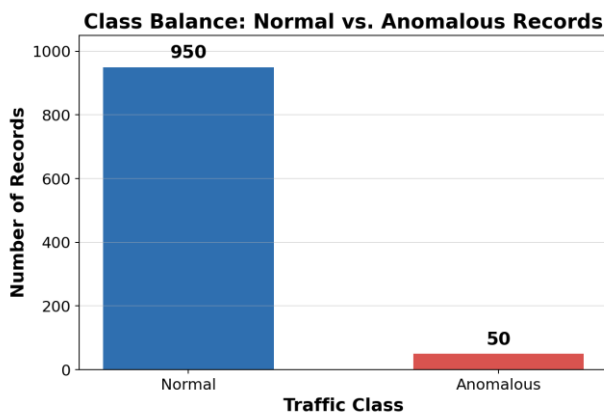


Fig. 5. Class balance of normal vs. anomalous network traffic records.

Fig. 6 breaks down the 50 anomalous records by likely incident type. The type is assigned by simple rules on the feature values; for example, 8 or more failed logins with moderate data transfer is labelled a possible brute-force attempt. Most anomalous records (40) fall under possible data exfiltration, because they combine very high byte counts with long sessions. Eight records are labelled possible brute-force attempts, and one record each is labelled a possible credential attack and suspicious network activity.

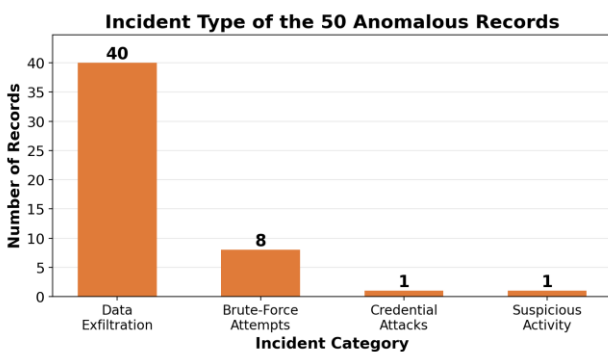


Fig. 6. Rule-based incident type of the 50 anomalous records.

C. Confusion Matrix

Table II and Fig. 7 report the confusion matrix of the Isolation Forest model on the evaluation set of 240 records. The model detected all 50 anomalous records, so there are no false negatives, and it classified 178 of the 190 normal records correctly. It wrongly flagged the other 12 normal records as anomalous (false positives).

Table II. Confusion matrix of Isolation Forest on the evaluation set.

	Pred. Normal	Pred. Anomaly
Actual Normal	178	12
Actual Anomalous	0	50

These 12 false positives are about 6.3% of the normal evaluation records. The decision threshold is set so that about 5% of the training records are flagged, so a similar share of unseen normal records is also flagged. A single split of synthetic data is not enough for strong conclusions, so the result should be checked with cross-validation and different random seeds before any real deployment.

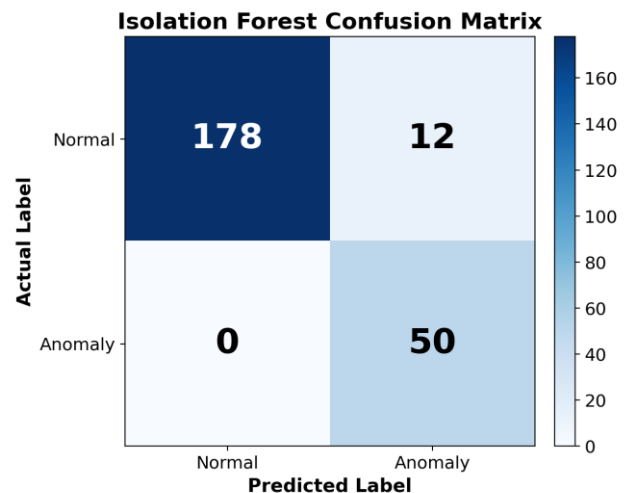


Fig. 7. Confusion matrix for the Isolation Forest model.

D. Performance Metrics: Accuracy, Precision, Recall, F1-score

Table III reports the detection scores of the three methods on the evaluation set. All three models reach a recall of 100%, so none of them missed an anomaly. They differ in precision, that is, in how many normal records they wrongly flagged. LOF has the highest accuracy (95.42%), precision (81.97%), and F1-score (90.09%). Isolation Forest is second (accuracy 95.00%, F1-score 89.29%), and One-Class SVM is third (accuracy 93.75%, F1-score 86.96%). The differences are small, because they come from only 11 to 15 false positives per model, so they should not be treated as a firm ranking.

Table III. Performance comparison across models (values in %).

Model	Acc.	Prec.	Rec.	F1
Isolation Forest	95.00	80.65	100.00	89.29
LOF	95.42	81.97	100.00	90.09
One-Class SVM	93.75	76.92	100.00	86.96

Because recall is identical for the three models, the comparison depends on precision. LOF produced one false alarm fewer than Isolation Forest (11 against 12), and One-Class SVM produced the most (15). In this dataset the anomalies lie far from the normal group in all three features, so every method finds them easily; the difference between the

methods lies only in how they treat normal records near the edge of the normal range.

E. Visualization

Fig. 8 shows the distribution of the risk score for the 1,000 records. The risk score is a weighted formula on the three features, which the dashboard uses: bytes sent contributes up to 35 points, session duration up to 25 points, and failed logins up to 40 points (each scaled against a reference value), with a maximum of 100. It is not the Isolation Forest score itself. Normal records have risk scores between about 11 and 42 (mean 20.5), while anomalous records score between about 62 and 100 (mean 93.3). The two groups do not overlap, which shows that the three features separate the two classes well in this synthetic data.

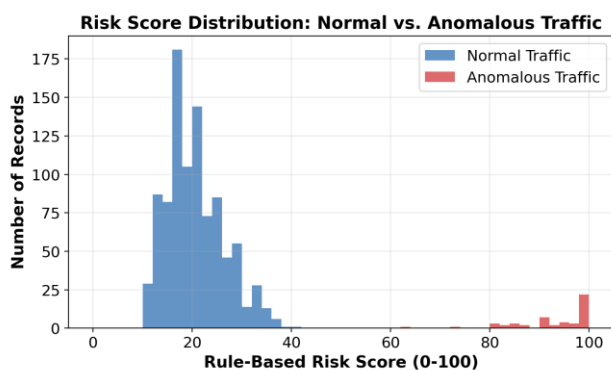


Fig. 8. Rule-based risk score distribution for normal and anomalous records.

F. Web Dashboard

The trained model is also used in a web dashboard built with Flask. Fig. 9 shows its sign-in page. After sign-in, the dashboard shows an approximate login location taken from the IP address. When the application runs on a local machine, this is shown as a local network, because a public location is not available for the address 127.0.0.1.

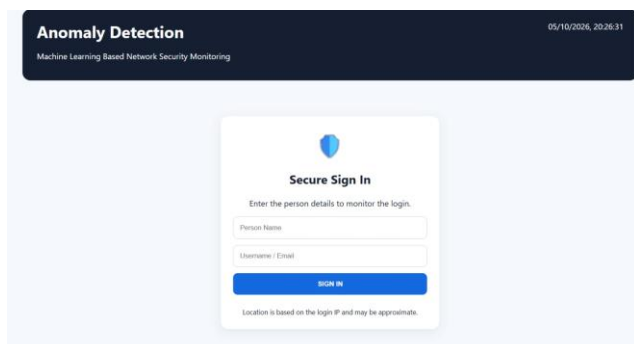


Fig. 9. Sign-in page of the web dashboard.

Fig. 10 shows the result of analysing two network log entries. The first entry (520 bytes, 72 seconds, and 2 failed logins) is classified as normal, with a risk score of 21.27. The second entry has the same values but 10 failed logins. It is

flagged as an anomaly with a risk score of 53.27 (suspicious) and a possible brute-force attempt, and the dashboard shows the reason and a recommended action.

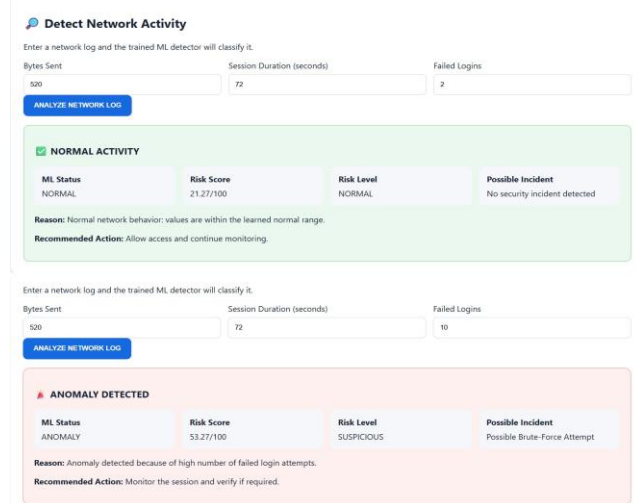


Fig. 10. Dashboard result for a normal entry (top) and an anomalous entry (bottom).

Fig. 11 shows the model comparison table of the dashboard, which matches Table III, together with a built-in demonstration that passes one normal and one anomalous record from the dataset through the model. The normal record receives a risk score of 17.44 and the anomalous record receives 95.16 (critical).

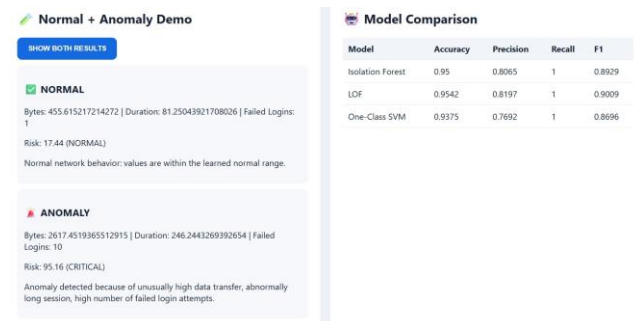


Fig. 11. Demonstration results and model comparison in the dashboard.

Fig. 12 shows the dashboard graphs. The first chart shows 950 normal and 50 anomalous records, and the chart of average features shows much higher bytes sent, session duration, and failed logins for anomalous records. The risk-level and incident charts are built from the entries analysed during the session, so in this screenshot they show only the single brute-force test entry from Fig. 10.



Fig. 12. Graphs shown in the web dashboard.

VII. RESULTS AND DISCUSSION

A. Analysis of Detection Results

Table II and Fig. 7 show that the Isolation Forest model detected all 50 anomalous records in the evaluation set, so no abnormal record was missed. However, it also flagged 12 of the 190 normal records as anomalous. In a security setting, such false alarms cost analyst time, so this number matters as much as recall. The risk-score distribution in Fig. 8 shows a clear gap between the two groups, but this is expected because the anomalies were generated with values far from the normal ones. Results on real network traffic may differ, mainly because real traffic is noisier. In real-world deployments, seasonal and time-of-day variations in legitimate traffic should also be considered. If the training data does not represent normal traffic across different operating periods, such variations may be incorrectly classified as anomalies.

B. Performance Analysis

Isolation Forest achieved performance comparable to LOF and better than One-Class SVM on the evaluation set (F1-scores of 0.893, 0.901, and 0.870). The three features were enough to separate the generated anomalies, as shown by the perfect recall of all models. The number of false positives depends on the contamination setting (0.05 here) and on the chosen features. Parameter tuning and evaluation on different datasets are therefore needed before practical use.

C. Comparison with Existing Methods

Table III shows that LOF obtained the highest accuracy, precision, and F1-score, followed by Isolation Forest and then One-Class SVM, while all three had a recall of 100%. The F1-scores were 90.09% for LOF, 89.29% for Isolation Forest, and 86.96% for One-Class SVM. The differences are small and come from a few false alarms on a single split of synthetic data, so these results do not show that one method is clearly better than the others. All models used the same data split and scaling, with contamination = 0.05 for Isolation Forest and LOF and nu = 0.05 for One-Class SVM.

D. Discussion

The experimental results indicate that Isolation Forest is a workable approach for unsupervised network anomaly detection when labelled attack data is limited, since it found all 50 anomalies without seeing any anomalous record during training. However, in this experiment it did not outperform LOF, and it produced 12 false positives. The main limitations are that the dataset is synthetic and has only three features, that a single train-test split was used, that the contamination parameter was fixed at 0.05 and not tuned, and that no cross-validation was done. Real traffic would need more features, such as ports, protocols, and packet counts, and real log data. These limitations motivate future work with real datasets and deep-learning-based methods.

In the dashboard, a new log entry is marked as an anomaly when the trained Isolation Forest flags it or when simple threshold rules are triggered (8 or more failed logins, 2,500 or more bytes sent, or a session of 180 seconds or more). The dashboard then reports a risk score, a risk level (normal, suspicious, high risk, or critical), a possible incident type, and a recommended action. For example, an entry with 520 bytes, 72 seconds, and 2 failed logins was classified as normal with a risk score of 21.27, while the same entry with 10 failed logins was flagged as a possible brute-force attempt with a risk score of 53.27 (suspicious) (Fig. 10). These are demonstration inputs and are not part of the quantitative evaluation, and the threshold rules are set by hand.

Periodic retraining on recent log data may help with concept drift [10]. Watching how the anomaly scores change over time can also show when the model or the features need updating.

VIII. CONCLUSION

This paper presented a framework for detecting anomalies in network security logs using the Isolation Forest algorithm. The system combines preprocessing with an unsupervised isolation-based model that is trained on normal records only, so it does not need labelled attack data. On a synthetic dataset of 1,000 records, Isolation Forest detected all 50 anomalies in the evaluation set, with 12 false positives among 190 normal records (F1-score 0.893). LOF performed slightly better (F1-score 0.901) and One-Class SVM slightly worse (0.870), so the three methods performed similarly on this data. A web dashboard built on the trained model shows risk scores and incident types for new log entries. Because the data is synthetic, further testing on real network logs is needed before the approach can be used in practice.

IX. FUTURE SCOPE

Future work will extend the framework with real-time network log ingestion and automated alerts, because the current dashboard works on manually entered records. The system can also be tested on real-world benchmark datasets with more

features [2], [6], [8], checked with cross-validation, and compared with deep-learning-based anomaly detection methods [7], [15]. The contamination parameter can be tuned to reduce false alarms, and extended variants of Isolation Forest can be explored [9]. Periodic model retraining can be investigated to handle changes in network traffic patterns and concept drift in practical environments.

REFERENCES

- [1] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413-422.
- [2] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA), 2009.
- [3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009.
- [4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in Proc. ACM SIGMOD Int. Conf. Management of Data, 2000, pp. 93-104.
- [5] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol. 13, no. 7, pp. 1443-1471, 2001.
- [6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. Int. Conf. Information Systems Security and Privacy (ICISSP), 2018.
- [7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
- [8] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in Proc. Military Communications and Information Systems Conf. (MilCIS), 2015.
- [9] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended Isolation Forest," IEEE Trans. Knowledge and Data Engineering, vol. 33, no. 4, pp. 1479-1489, 2021.
- [10] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys, vol. 46, no. 4, pp. 1-37, 2014.
- [11] H. Bhatt, K. Rajeswari, and V. Kumar, "Security information and event management (SIEM): Analysis, survey and open challenges," in Proc. IEEE Int. Conf. Computing, Communication and Networking Technologies (ICCCNT), 2021.
- [12] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-Based Anomaly Detection," ACM Transactions on Knowledge Discovery from Data, vol. 6, no. 1, pp. 1-39, 2012.
- [13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.
- [14] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
- [15] G. C. Fernández and S. Xu, "A Case Study on Using Deep Learning for Network Intrusion Detection," arXiv preprint arXiv:1910.02203, 2019.