Automated Threat Detection Using Machine Learning in Python

Mrs. Rebekah John, Professor Faculty
Computer Science and Engineering
Rajalakshmi Institute of Technology,
Chennai,India
rebekah.j@ritchennai.edu.in

Purushothaman V
Computer Science and Engineering
Rajalakshmi Institute of Technology,
Chennai,India
purushothaman.v.2021.cse@ritchennai.edu.in

Roshan A
Computer Science and Engineering
Rajalakshmi Institute of Technology,
Chennai,India
roshan.a..2021.cse@ritchennai.edu.in

Abstract— The rapid evolution of cyberattacks such as ransomware, Distributed Denial-of-Service (DDoS), and advanced persistent threats highlights the limitations of traditional rule-based and signature-based intrusion detection systems, which struggle to detect real-time threats in the face of increasing network complexity. In order to provide a reliable method for detecting both known and unknown cyberthreats, this study presents an automated threat detection system that uses machine learning to examine network packet capture (PCAP) data. To achieve high accuracy and adaptability, the suggested framework blends the supervised learning algorithm Random Forest with the unsupervised anomaly detection technique Isolation Forest. While Isolation Forest minimizes false positives in dynamic contexts by isolating unique abnormalities, Random Forest is excellent at categorizing existing assault patterns.The system trains the hybrid model for real-time deployment, guaranteeing scalability across cloud infrastructures and enterprise networks, after preprocessing raw PCAP logs to remove noise and extract parameters like packet size and flow duration. Tested on the CICIDS2017 dataset, which contains a variety of attack scenarios like port scanning and DDoS, the system achieves remarkable precision (0.99) and recall (0.98), as well as 98% accuracy through Random Forest and a 20% false positive rate. Isolation Forest improves detection even more by identifying zero-day anomalies that signature-based techniques miss.Using fresh threat data, a dynamic feedback loop continuously improves the model's resilience against changing attack methods. It is perfect for Security Operations Centers (SOCs), cloud platforms, and real-time monitoring systems due to its excellent accuracy, scalability, and adaptability, even with a moderate 20% false positive rate. By bridging the gap between state-of-the-art machine learning and real-world cybersecurity requirements, this research provides actionable intelligence for proactive threat mitigation and lays the groundwork for future advancements in automated detection systems.

Keywords— Machine Learning, Cybersecurity, Network Traffic Analysis, Anomaly Detection, Real-Time Threat Detection.

I. INTRODUCTION

The increasing frequency and sophistication of cyberattacks has made cybersecurity a crucial worry in today's digital environment. Threat actors can now use ransomware, Distributed Denial-of-Service (DDoS) assaults, and Advanced Persistent Threats (APTs) due to the expansion of the attack surface brought about by the growth of cloud computing, Internet of Things (IoT) devices, and enterprise digital transformation. Conventional defenses, such firewalls and antivirus programs, match network behavior to pre-established threat patterns using rule-based or signature-based detection. These methods are effective against known threats, but they are ineffective against emerging attacks with no identifiable signs, such as zero-day exploits and polymorphic malware. Furthermore, they are impracticable for real-time defense in dynamic network environments due to their static nature, which necessitates regular manual upgrades.

This cybersecurity gap is made worse by a number of difficulties. Attack methods are evolving faster than traditional systems, resulting in high false positive rates that overwhelm security analysts. Scalability problems arise from the large volume of network traffic, which is frequently recorded in Packet Capture (PCAP) logs. Additionally, model training is complicated by imbalanced datasets, where benign traffic greatly outweighs malicious cases. Computing resources are further taxed by real-time detection, which is necessary for prompt remediation. These drawbacks highlight the necessity of automated, intelligent systems that can respond to emerging dangers without requiring significant human involvement.This study suggests an automated threat detection system that uses machine learning (ML) to examine network PCAP records in real time in order to overcome these difficulties.

The system incorporates Isolation Forest, an unsupervised method for identifying new anomalies, and Random Forest, a supervised learning algorithm, for accurately classifying established assault patterns. The method achieves scalability and robustness by using a hybrid model, preprocessing logs, and extracting features. A feedback loop adjusts to new threats, improving detection even further. The system provides a workable solution for Security Operations Centers (SOCs) and cloud environments, with a 20% false positive rate and 98% accuracy when tested on the CICIDS2017 dataset.

A. Challenges in Cyber Threat Detection

1. Evolving Attack: Techniques: Cybercriminals are always coming up with new and advanced ways to attack, which makes it hard for conventional security systems to stay up to date.

2. High False Positives: A lot of security systems produce a lot of false alarms, which makes security analysts tired of being on guard.

3. Scalability Issues: It is difficult to create scalable security solutions that can effectively handle big datasets due to the growing volume of network traffic and log data.

4. Data Imbalance: Model training and evaluation are challenging due to the fact that cybersecurity datasets are frequently unbalanced, with a disproportionately low number of malicious cases compared to benign ones.

5. Real-time Detection: Real-time analysis is necessary for the prompt detection and mitigation of risks, which presents operational and computing difficulties.

B. Objectives of the Research

The primary objectives of this study include:

1. Developing a real-time log analysis system using machine learning models to detect and classify threats.

2. Enhancing detection accuracy while minimizing false positives, ensuring high precision in identifying security threats.

3. Ensuring scalability and efficiency by optimizing data processing techniques for large-scale security logs.

4. Implementing a feedback mechanism to continuously refine model performance based on evolving cyber threats.

5. Investigating integration with existing security frameworks, such as Security Information and Event Management (SIEM) systems, to enhance automated threat detection capabilities.

C. Significance of the Study

By creating an automated, feedback-driven threat detection framework, this research seeks to close the gap between theoretical developments in machine learning and real-world cybersecurity applications. By offering an intelligent mechanism to effectively identify, categorize, and respond to cyber threats, the suggested system will improve cybersecurity resilience.This research advances the creation of proactive, scalable, and intelligent threat detection systems that can instantly address contemporary cyberthreats by incorporating cutting-edge machine learning models into cybersecurity frameworks. By putting this approach in place, firms will be able to identify threats sooner, reduce damage, and strengthen their overall security posture against a constantly changing cyber threat landscape.

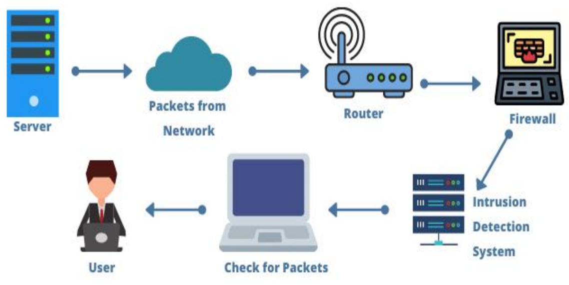


Fig 1. Threat Detection using Network PCAPS

The diagram depicts an Automated Threat Detection (ATD) system that continually monitors network traffic, analyzes logs, and identifies anomalies in real time. Network traffic is created by the server, sent as packets, and then directed via a router to its final location. Although a firewall serves as the first line of defense, sophisticated attackers can still get past it. In order to combat this, ATD's integrated Intrusion Detection System (IDS) looks for questionable trends in traffic logs. In contrast to conventional rule-based intrusion detection systems, ATD uses Machine Learning (ML) methods such as Random Forest and Isolation Forest to proactively identify anomalies. Following log analysis, any questionable activity is marked for additional examination.In order to mitigate risks like DDoS attacks, unauthorized access, or port scanning efforts in real time, the system then notifies the security analyst or automatic reaction mechanism.

For instance, fraudsters may try to commit fraud, account takeovers, or infiltration attacks in the internet banking system of a financial institution. Basic threats are filtered by the firewall, but more sophisticated strategies, such as fraudulent transactions utilizing hacked accounts, might go undetected. In this case, ATD examines transaction records to find odd login attempts, frequent withdrawals, or big money transfers to dubious accounts. Automated reactions to anomalies can alert security teams, halt transactions, or even temporarily lock the impacted account to stop additional harm.Financial institutions may effectively minimize cyber threats with this proactive strategy, lowering the risk of fraud and preserving operational security in real time.By integrating ATD, a strong, flexible security architecture that can change to meet new cyberthreats is guaranteed, offering improved defense for vital infrastructures.

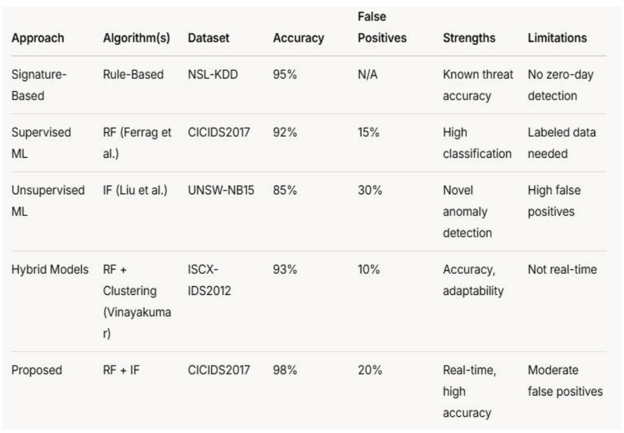
II. LITERATURE REVIEW

Automated Threat Detection (ATD), which uses machine learning techniques to monitor massive volumes of network data and identify anomalies in real time, has emerged as a key element of contemporary cybersecurity frameworks. Enhancing ATD through increased detection accuracy, decreased false positives, and guaranteed scalability has been the subject of numerous studies.Recent research has exposed the shortcomings of traditional rule-based systems, which rely on predetermined attack signatures and fail to detect evolving threats. Because machine learning techniques can identify previously unidentified attack patterns, they have become popular in ATD. Network traffic has been classified as either benign or malicious using supervised learning models like Random Forest and Support Vector Machines (SVM). These models, however, need labeled datasets, which aren't always accessible. Unsupervised learning methods, such as Autoencoders and Isolation Forest, on the other hand, have proven successful in anomaly identification without the need for labeled data, which makes them vulnerable to zero-day assaults.In order to improve detection skills, a number of studies have investigated hybrid models that combine supervised and unsupervised learning techniques. In addition to using historical attack data for classification, these models also look for changes in network behavior to spot new threats.

Hybrid models are appropriate for real-time cybersecurity applications since they dramatically increase detection accuracy while lowering false positive rates, according to research.Prior research has extensively employed benchmark datasets,including UNSW-NB15, CICIDS2017, and ISCX-IDS2012, to assess ATD models. These databases, however, frequently lack the complexity of the real world or may not account for the most recent cyberthreats.By using network traffic datasets that contain a variety of attack behaviors, including Distributed Denial-of-Service (DDoS), penetration attempts, and port scanning, our research fills this vacuum. Furthermore, we improve on current approaches by adding real-time log analysis and sophisticated feature extraction techniques to increase the effectiveness of threat identification.Scalability is another important aspect of ATD study. It is essential to create models that can effectively handle large-scale datasets given the exponential rise in network traffic. To increase processing speed and scalability, studies have suggested distributed computing frameworks like TensorFlow and Apache Spark. By improving feature selection and incorporating a feedback loop for ongoing model improvement, our method expands on existing developments and guarantees flexibility in response to changing cyberthreats.This research advances the creation of scalable, highly accurate cybersecurity solutions by bridging the gap between machine learning-driven ATD systems and conventional threat detection procedures. A proactive defense mechanism against contemporary cyberthreats is ensured by the combination of machine learning algorithms, feature engineering, and real-time anomaly detection.

This study uses a hybrid ATD model that combines supervised Random Forest for threat classification with unsupervised Isolation Forest for anomaly detection. This method improves classification accuracy, reduces false positives, and detects zero-day attacks, hence overcoming the drawbacks of conventional models.

This methodology finds novel attack patterns in real time, unlike signature-based systems that merely identify known threats. Compared to solo supervised or unsupervised methods, the hybrid approach is more effective since it guarantees improved adaptability, scalability, and automated response. It improves cybersecurity protections while preserving high accuracy and dependability in network traffic analysis by constantly learning from evolving threats. Furthermore, by combining anomaly detection with categorization, security teams can identify risks earlier and require less manual involvement.The below table depicts the various approaches for threat detection.



Approach	Algorithm(s)	Dataset	Accuracy	False Positives	Strengths	Limitations
Signature-Based	Rule-Based	NSL-KDD	95%	N/A	Known threat accuracy	No zero-day detection
Supervised ML	RF (Ferrag et al.)	CICIDS2017	92%	15%	High classification	Labeled data needed
Unsupervised ML	IF (Liu et al.)	UNSW-NB15	85%	30%	Novel anomaly detection	High false positives
Hybrid Models	RF + Clustering (Vinayakumar)	ISCX-IDS2012	93%	10%	Accuracy, adaptability	Not real-time
Proposed	RF + IF	CICIDS2017	98%	20%	Real-time, high accuracy	Moderate false positives

Fig 2. Different Approaches to ATD

III. PROPOSED METHODOLOGY

A. System Architecture

The Automated Threat Detection (ATD) system uses machine learning techniques to classify cyber threats, examine network traffic data, and identify anomalies. Real-time detection, less false positives, and flexibility in response to changing threats are all guaranteed by the architecture's organized pipeline. Different stages of data gathering, analysis, and reaction fall under the purview of each component.The method begins with the collection of data, which collects network traffic records from datasets comprising attacks such as DDoS, port scanning, and infiltration attempts. Important markers of cyberthreats, such as IP addresses, protocol kinds, packet sizes, and timestamps, are included in these logs.Data preparation is done to clean and convert the data by handling missing values, normalizing features, and choosing pertinent variables to increase model accuracy because raw logs are frequently noisy. By identifying significant patterns, feature engineering aids in distinguishing malicious behavior from legitimate communication.

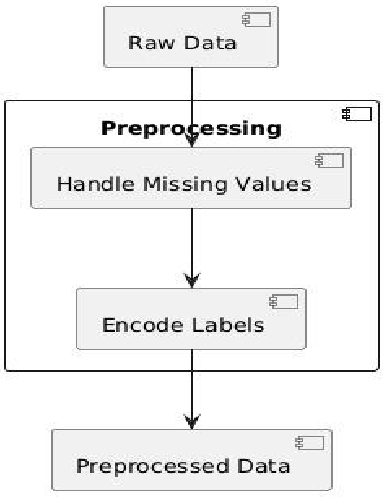


Fig 3. Data Preprocessing

The data is loaded into a hybrid machine learning model after preprocessing, where:

1. Isolation Forest (Unsupervised Learning) : This technique looks for odd patterns and irregularities in network activity that could point to zero-day attacks or other dangers that haven't been noticed yet.

2. Random Forest (Supervised Learning) : Random Forest (Supervised Learning) uses labeled attack data from past logs to categorize network behavior as either malicious or benign. This guarantees that well-known attack patterns are accurately detected.

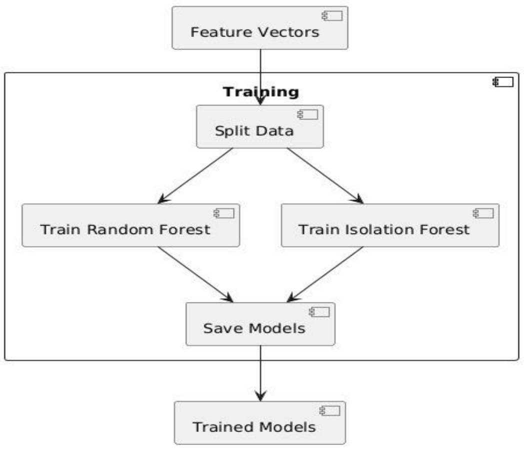


Fig 4. Taining ML models

Compared to standalone models, this hybrid method offers a more complete security system by guaranteeing the detection of both known and undiscovered threats. In order to enable automated or manual actions like blocking suspicious IPs or isolating impacted devices, the system analyzes detection findings and generates real-time notifications when an abnormality or malicious activity is discovered.The system also has a feedback mechanism that allows the model to be continuously adjusted to changing attack methods and network behaviors by using recently discovered threats to retrain it.

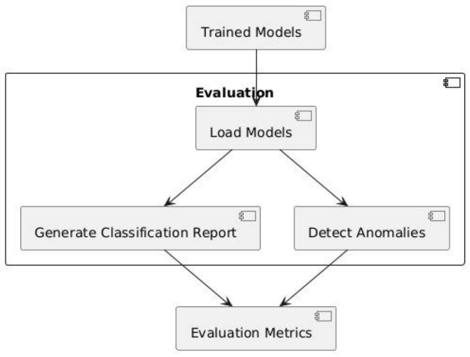


Fig 5. Model Evaluation

The system increases its long-term accuracy, efficiency, and resistance to cyberattacks by learning from fresh data.By improving cybersecurity through automated threat detection and quick reaction mechanisms, the ATD system's scalable, adaptable design makes it appropriate for enterprise networks, cloud infrastructures, and Security Operations Centers (SOCs).

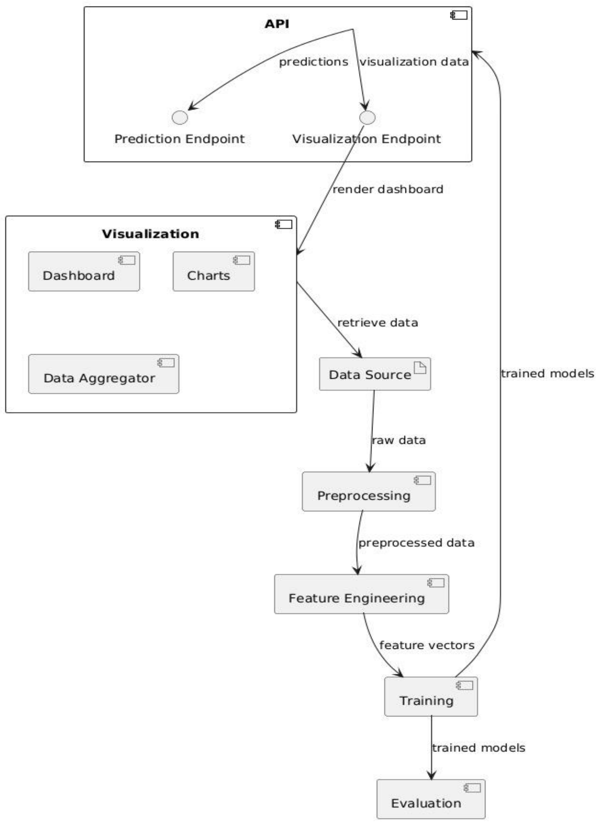


Fig6. ATD System Architecture for Real-Time Threat Detection.

Through an integrated pipeline, the design of the ATD system, shown in Fig. 1, enables real-time threat identification from network PCAP data. A data source (such as CICIDS2017 or live captures) first records the raw traffic. Preprocessing then cleans and normalizes the data, and feature engineering is used to extract vectors like packet size and flow length. The Training module uses these vectors to create a hybrid Random Forest (RF) and Isolation Forest (IF) model, where RF is used to identify known threats and IF is used to discover abnormalities. Performance is then validated through evaluation. For real-time predictions, trained models are deployed via an API Endpoint. The results are shown on a Visualization Dashboard with charts for analysts via a Visualization Endpoint. Data is retrieved by a data aggregator for ongoing updates, guaranteeing flexibility through a feedback loop, making the system scalable for cloud and enterprise environments.

B. Data Collection

The Automated Threat Detection (ATD) system analyzes abnormalities and attack patterns in network traffic logs to identify cyberthreats. These logs, which are gathered from multiple sources and document network activity in real time, let the system to categorize traffic as either malicious or benign. Packet Capture (PCAP) files, which offer a thorough picture of network connection and enable in-depth examination of transmitted packets, are the main source of these logs.

1. Sources of Network Traffic Logs

Network logs for ATD are often obtained from real-world traffic monitoring systems, intrusion detection datasets, or network simulation environments.

i. Public Intrusion Detection Datasets

These datasets are helpful for training machine learning models since they comprise pre-labeled network traffic logs that cover both typical and attack scenarios.Examples of datasets that are accessible to the public include:

- ➢ Real-world attack scenarios, such as DDoS, infiltration, and brute force attacks, are captured by CICIDS 2017 and CSE-CIC-IDS 2018.
- ➢ UNSW-NB15: This tool simulates both malicious and benign network traffic.
- ➢ ISCX IDS 2012 (A labeled dataset with both regular traffic and other types of intrusion attempts).

ii. Real-Time Network Packet Capture(PCAPs)

Network managers use programs like Wireshark, Tcpdump, or Zeek to record real-time traffic.The source and destination IP addresses, protocol types, payload information, and timestamps of each packet sent over a network are all documented in these logs.PCAP files assist in identifying unknown anomalies and low-level attack signatures that rule-based systems could overlook.

iii. Simulated Attack Environments

Traffic is generated in controlled network conditions in order to create custom datasets.Security researchers can use this method to model novel attack vectors that might not be found in publicly available datasets.Network flow characteristics can be extracted from PCAP files using tools like CICFlowMeter to create organized datasets.

2. Types of Logs

i. Packet Capture (PCAP) logs

Packet Capture (PCAP) logs contain raw network packet data, capturing each packet sent over a network. These logs are very helpful for forensic analysis and deep packet inspection because they contain protocol specifics, source and destination addresses, timestamps, packet headers, and payload data. PCAP logs provide extensive insight into network behavior by capturing unfiltered network traffic at a granular level. This makes it possible to identify suspicious actions such as malware injections and unauthorized data transfers. Network monitoring tools that record real-time traffic from network interfaces are used to collect PCAP logs. For additional machine learning-based analysis, the gathered PCAP files are subsequently transformed into structured datasets, such CSV files.

ii. Flow-Based Logs

Flow-Based logs collect aggregated session information instead of individual packets, giving a higher-level overview of network traffic. These logs keep track of information including the total number of packets sent and received, the amount of data moved, the length of the session, and the communication endpoints. Flow-based logs are intended to detect odd network behaviors and large-scale traffic patterns, in contrast to PCAP logs, which provide fine-grained data. These logs are very helpful for identifying volumetric attacks like brute-force login attempts.

iii. Behavioral Logs

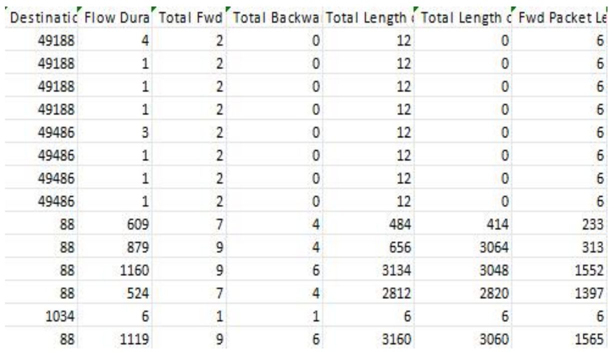
Behavioral logs monitor connection patterns, session lengths, and authentication attempts to offer information on user activity and network activities. These logs are useful for spotting irregularities that can point to a security breach, like unauthorized access attempts, odd login times, and unexpected session behaviors. For instance, behavioral logs would indicate a possible account takeover attempt if a user account abruptly started logging in from several locations in a brief amount of time.These logs are especially useful for spotting lateral movement within a compromised network, insider threats, and privilege misuse.

iv. Event-Based Logs

Event-based logs concentrate on documenting particular security-related events that take place on a network. Firewall alarms, intrusion detection system (IDS) triggers, unsuccessful authentication attempts, and privilege escalation attempts are all recorded in these logs. . For example, if an unauthorized IP address tries to connect to a restricted port on a regular basis, a firewall may produce an event log. In a similar vein, when an IDS notices a possible malware signature or an attempt at an exploit, it might record an event.

The integration of PCAP, Flow-Based, Behavioral, and Event-Based logs ensures a powerful and comprehensive threat detection strategy. When combined, the distinct insights from each kind of log improve the ATD system's precision and effectiveness. The system can accurately identify known and new cyberthreats by utilizing session-based flow analysis .

The primary data source for this study was PCAP logs, which were acquired using network monitoring tools such as Wireshark and Tcpdump. Live network traffic is captured by these technologies, which then store the packets in PCAP format before converting them into structured datasets (in CSV format) for machine learning research. Though they need to be preprocessed in order to extract useful aspects like protocol type, packet size, and session time, the raw PCAP files contain a wealth of network information. The hybrid ATD model is then trained using this structured data, allowing for the real-time detection of cyberthreats using anomaly detection and network behavior methods.



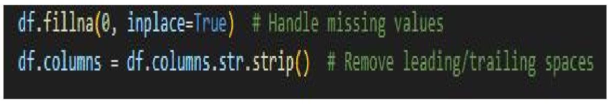
Destinatic	Flow Dura	Total Fwd	Total Backwa	Total Length (	Total Length c	Fwd Packet Le
49188	4	2	0	12	0	6
49188	1	2	0	12	0	6
49188	1	2	0	12	0	6
49188	1	2	0	12	0	6
49486	3	2	0	12	0	6
49486	1	2	0	12	0	6
49486	1	2	0	12	0	6
49486	1	2	0	12	0	6
88	609	7	4	484	414	233
88	879	9	4	656	3064	313
88	1160	9	6	3134	3048	1552
88	524	7	4	2812	2820	1397
1034	6	1	1	6	6	6
88	1119	9	6	3160	3060	1565

Fig 7. Example of PCAP logs dataset

C. *Data Preprocessing*

Data preprocessing is a vital step in converting raw PCAP (Packet Capture) logs to a structured format appropriate for machine learning-based threat identification. Before being used by machine learning models, PCAP files need to be cleaned, transformed, and normalized because they include low-level network packet information. The first step in the process is data collecting, which involves obtaining network traffic from publicly available datasets like CICIDS2017, UNSW-NB15, and ISCX-IDS2012 or by using tools like Wireshark, Tcpdump, or Zeek to record it in real-time. Source and destination IP addresses, protocol types, timestamps, packet sizes, and payload data are just a few of the many details regarding network flows that are included in these PCAP files.

To extract relevant network flow attributes, raw PCAP files must be transformed into structured forms like CSV utilizing CICFlowMeter, Zeek (previously Bro), or Tshark. This is because raw PCAP files are not immediately useable for machine learning.To guarantee data quality, missing and partial values must be handled after the data has been constructed. Network irregularities or packet loss can result in missing values. In order to remedy this, partial records may be imputed using statistical approaches like mean, median, or forward-fill procedures, or they may be deleted (if the missing proportion is considerable). Additionally, removing redundant flows, filtering out unnecessary protocols (like ARP and ICMP), and removing internal network traffic that doesn't add to security insights are all part of data cleaning and noise reduction.

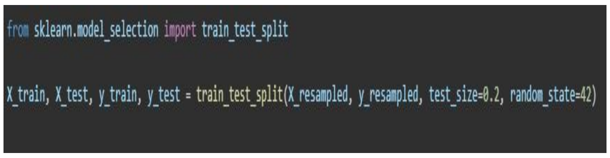


```
df.fillna(0, inplace=True)  # Handle missing values
df.columns = df.columns.str.strip()  # Remove leading/trailing spaces
```

Fig 8 .Data Cleaning

After cleaning the data, feature encoding and normalization are used to ensure consistency among machine learning models. Numerical features (packet size, flow duration) are normalized using Min-Max Scaling or Standardization to preserve uniformity, while categorical features (TCP, UDP, ICMP) are encoded using One-Hot Encoding or Label Encoding. Techniques like Synthetic Minority Over-sampling (SMOTE), undersampling of majority classes, or class weighting are used to balance the dataset and avoid bias in the machine learning model because cybersecurity datasets frequently suffer from class imbalance, where benign traffic greatly outnumbers attack traffic.

Finally, the preprocessed data is divided into training, validation, and test sets, assuring stratified sampling and uniform distributions of both benign and attack traffic across all datasets. Data augmentation methods, such Generative Adversarial Networks (GANs) or synthetic traffic creation, can be used to improve model robustness in situations where data availability is constrained. The solution guarantees high-quality input data by efficiently preparing PCAP logs, which enhances the precision and dependability of machine learning models used to detect cyberthreats.

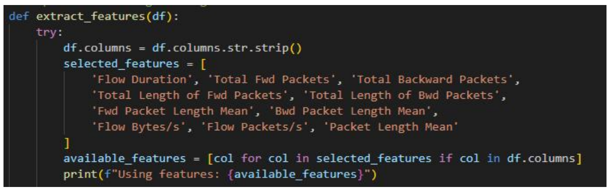


```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

Fig 9. Splitting Data for ML models

D. *Feature Engineering*

Feature engineering is an important phase in the threat detection pipeline because it extracts valuable insights from preprocessed network log data, which improves machine learning model performance. The objective is to convert unoptimized network traffic characteristics into features that enhance the system's capacity to identify online dangers.



```python
def extract_features(df):
    try:
        df.columns = df.columns.str.strip()
        selected_features = [
            'Flow Duration', 'Total Fwd Packets', 'Total Backward Packets',
            'Total Length of Fwd Packets', 'Total Length of Bwd Packets',
            'Fwd Packet Length Mean', 'Bwd Packet Length Mean',
            'Flow Bytes/s', 'Flow Packets/s', 'Packet Length Mean'
        ]
        available_features = [col for col in selected_features if col in df.columns]
        print(f"Using features: {available_features}")
```

Fig 10. Feature Extraction

Three primary methods are used in this process: dimensionality reduction, feature extraction, and feature selection.Deriving relevant characteristics from network flows, such as packet inter-arrival time, flow duration, total bytes transported, and protocol kinds, is known as feature extraction. To improve threat detection, more complex metrics can be calculated, such as burst rate, entropy, and statistical characteristics of packet size distributions. By helping to differentiate between benign and malevolent activities, these extracted attributes offer a richer understanding of network behavior.To determine the most essential characteristics that enhance model correctness, feature selection is carried out. Choosing the appropriate subset of features is crucial to increasing efficiency and lowering overfitting because network log collections frequently contain high-dimensional data. Prioritizing the most important traits for classification is aided by methods like Random Forest feature importance ranking and correlation analysis, which eliminates redundant features. This guarantees that the model eliminates irrelevant data while concentrating on important signs of harmful activity.

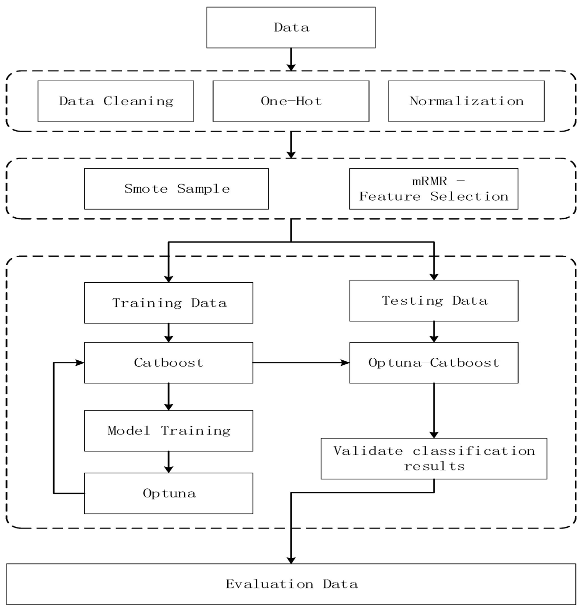


Fig 11. Data preprocessing and Feature Engineering

The Automated Threat Detection project's organized pipeline for processing PCAP logs is depicted in the diagram. To make sure it is prepared for analysis, raw network data is first cleaned, one-hot encoded, and normalized. The most pertinent characteristics, including packet size and flow time, are extracted via feature selection (mRMR), whereas SMOTE manages class imbalance by creating artificial attack samples. After that, the data is divided into training and testing sets, allowing models such as Random Forest and Isolation Forest to identify malicious behavior patterns. Model performance is optimized for increased accuracy through hyperparameter adjustment (Optuna). To guarantee efficient threat identification, the trained model is verified and assessed. Lastly, the evaluation results can be applied to real-time security monitoring, which aids in the effective identification of cyberthreats. Dimensionality reduction methods such as Principal Component Analysis (PCA) or t-SNE can be used to further improve model performance. By converting the feature space into a lower-dimensional representation, these techniques remove noise while maintaining crucial variance. When working with huge datasets, this step is very helpful because it increases computing efficiency without compromising detection accuracy.The automated threat detection system improves its capacity to precisely identify cyberthreats by putting feature engineering into practice. A scalable and efficient security solution is ensured by carefully chosen and designed features that improve real-time performance, decrease false positives, and increase classification accuracy.

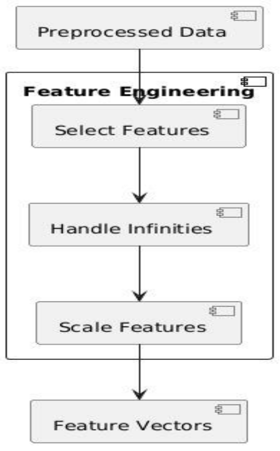


Fig 12.Feature Engineering and Handling

E. *Model Development and Training*

In this project, the main focus of the development of models is the integration of Random Forest (RF) and Isolation Forest (IF) to create a hybrid intrusion detection system that can detect unexpected anomalies as well as known cyberthreats. Isolation Forest is an unsupervised anomaly detection model that separates zero-day assaults from anomalous traffic patterns, whereas Random Forest is a supervised classification model that finds predetermined attack patterns. When combined, these models improve the system's precision, flexibility, and resilience. Data splitting and preprocessing, Random Forest and Isolation Forest training, model optimization via hyperparameter tuning, and model validation and evaluation comprise the five main phases of the training process.

1. Data Splitting and Preprocessing

The PCAP-derived dataset is preprocessed to guarantee data quality, balance, and consistency prior to model training. To ensure that the models learn from historical data while being assessed on new data, the dataset is separated into training (70–80%) and testing and validation(20–30%) sets. Before doing final testing, model parameters may occasionally be adjusted using a validation set.

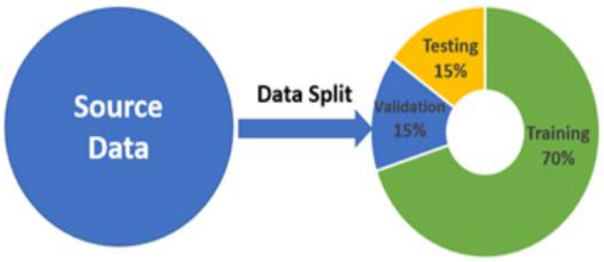


Fig 13. Data Splitting

Class imbalance is a major problem in cybersecurity datasets, as attack instances are much lower than typical traffic. This is addressed by using the Synthetic Minority Over-sampling Technique (SMOTE), which creates synthetic data points based on preexisting ones in order to increase the number of attack samples. This guarantees that the Random Forest model can accurately categorize assaults and avoid becoming biased toward innocuous network traffic.In order to guarantee numerical consistency across various features, feature scaling and normalization are also carried out. These statistics can differ greatly since network traffic data includes characteristics like packet size, flow length, and byte transmission rate. To prevent any one feature from controlling the learning process, standardization or min-max scaling is used. The models are now prepared for training since the dataset has been cleaned, balanced, and organized.

2. Training the Random Forest Model

A supervised machine learning model for categorization, Random Forest works especially well with structured network data. It is made up of several decision trees that have been trained on various data subsets. By cooperating through majority voting, these trees guarantee robustness against noisy and overfitting data.In order to guarantee that the model concentrates on the most pertinent network features during training, feature selection is essential. Protocol type, inter-arrival time, and payload size are among the most important features that are ranked by the model using methods like Gini Importance and mRMR (Minimum Redundancy Maximum Relevance). Features that are redundant or irrelevant are eliminated to increase detection accuracy and efficiency.

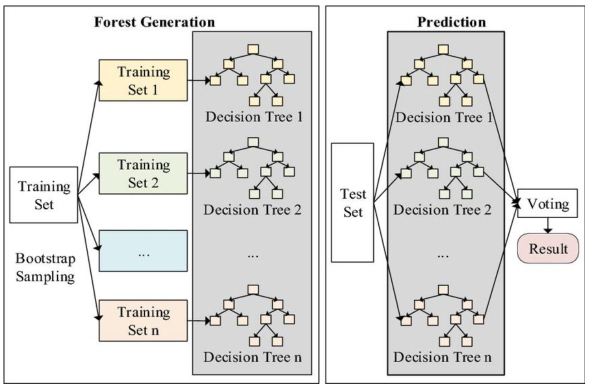


Fig 14. Random Forest

A random subset of the dataset is used to train each decision tree in the Bootstrap Aggregation (Bagging) method of training. This guarantees improved generalization and keeps the model from learning particular attack patterns. A majority vote across all trees determines the final classification after the trees have been trained to forecast whether an incoming network flow is an attack or a normal flow.An powerful optimization framework called Optuna is used to apply hyperparameter tweaking. To improve classification performance, the split criterion (Gini vs. Entropy), maximum tree depth, and number of trees are adjusted. To make sure the Random Forest model consistently identifies known cyberthreats, it is assessed after training using measures including accuracy, precision, recall, and F1-score.

3. Training the Isolation Forest Model

Isolation Forest is an unsupervised anomaly detection technique, which means it doesn't require labeled attack data like Random Forest does. Rather, it isolates data points that substantially differ from typical traffic patterns in order to learn how to identify outliers. When threats don't match known attack signatures, this works especially well for detecting zero-day attacks.

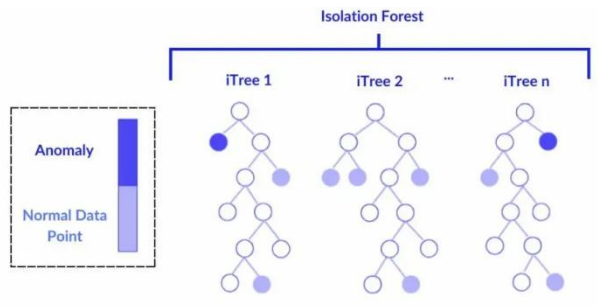
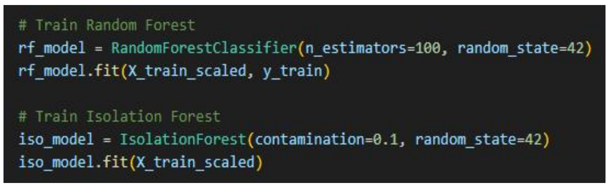


Fig 15. Isolation Forest

In order to separate anomalous cases more quickly than regular examples, the model randomly divides the dataset into decision trees. The speed at which a data point is isolated determines its anomaly score; the fewer splits needed, the more probable the assault. By using this method, the system can identify anomalous traffic patterns that could point to novel or advanced attacks.The threshold for classifying a data point as anomalous is determined by the contamination parameter, which is adjusted to strike a compromise between false positive rates and detection accuracy. While a larger number may boost sensitivity at the expense of more false alarms, a low contamination value guarantees that only really suspect traffic is reported.

The Isolation Forest model is validated using AUC-ROC scores and false positive rates after training to make sure it detects cyberthreats accurately and without overclassifying. After training, the model is used to track network traffic in real time and spot any irregularities that can pose a security concern.



```
# Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)

# Train Isolation Forest
iso_model = IsolationForest(contamination=0.1, random_state=42)
iso_model.fit(X_train_scaled)
```

Fig 16. Training Random Forest and Isolation Forest

4. Model optimization using Hyperparameter Tuning

Both Random Forest and Isolation Forest go through hyperparameter optimization to increase performance and improve their predictive power. This procedure is automated with Optuna, which tests several setups and chooses the model settings that perform the best.

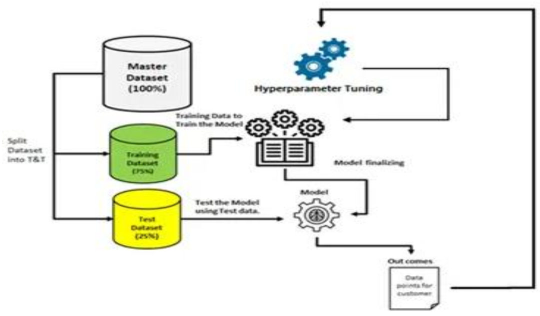


Fig 17. Hyperparameter Tuning

Important parameters adjusted for Random Forest include:

- ➢ Number of estimators (trees): Establishes how many decision trees there are in the ensemble overall.
- ➢ Tree maximum depth: Determines the maximum depth at which a tree can grow before ceasing to develop.
- ➢ Split criteria: Entropy or Gini impurity, which influences how decision trees determine splits.

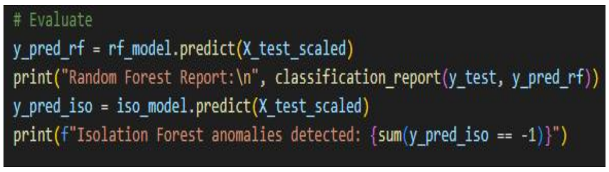
The primary parameters that have been optimized for Isolation Forest are:

- ➢ Level of contamination: The percentage of data deemed abnormal.
- ➢ Number of base estimators:The quantity of base estimators regulates how many isolation trees are constructed.
- ➢ Maximum samples per tree: This specifies the subset of data on which each tree will be trained.

Optuna ensures that the final models are accurate and computationally efficient by effectively exploring various hyperparameter values via Bayesian optimization.This optimization procedure greatly improves the hybrid system's capacity to efficiently identify known and unknown cyberthreats.

5. Model Validation and Evaluation

To make sure the models are effective, they are tested on unseen PCAP network traffic once they have been trained. The evaluation procedure entails determining how well the models can distinguish between instances of normal traffic and attacks.



```
# Evaluate
y_pred_rf = rf_model.predict(X_test_scaled)
print("Random Forest Report:\n", classification_report(y_test, y_pred_rf))
y_pred_iso = iso_model.predict(X_test_scaled)
print(f"Isolation Forest anomalies detected: {sum(y_pred_iso == -1)}")
```

Fig 18. Evaluation of both models

Standard classification metrics are applied to Random Forest:

- ➢ Accuracy: Indicates how accurate a classification is overall.
- ➢ Precision: Guarantees that dangers identified are real and not just false alarms.
- ➢ Recall: Indicates how effectively the model recognizes real dangers.
- ➢ F1-Score: A metric used to assess overall performance that strikes a balance between recall and precision.

Metrics for anomaly detection in Isolation Forest consist of:

- ➢ AUC-ROC Score: Evaluates the model's ability to differentiate between attack and legitimate traffic.

- False Positive Rate (FPR): Reduces needless warnings by making sure that only genuine threats are reported.

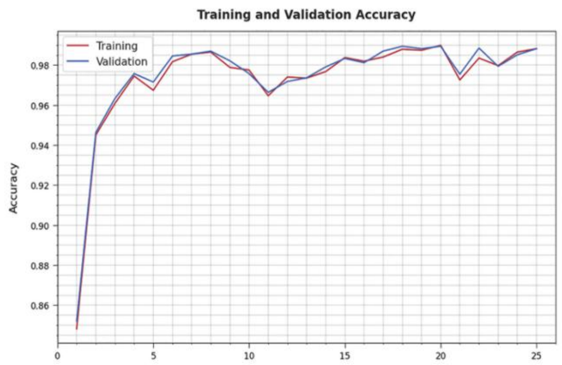


Fig 19. Training and Validation

Further training iterations are carried out with improved feature sets and modified hyperparameters to increase detection accuracy if the models do not perform at the anticipated levels.After being verified, the hybrid threat detection system is put into use for network monitoring in real time. It analyzes incoming PCAP traffic to dynamically identify and stop cyberthreats. The system is regularly supplied fresh attack data through an integrated feedback loop, which enables it to adjust to changing threats and get better over time.

F. *Threat Analysis Module*

The Threat Analysis Module is an essential component of the automated threat detection system, recognizing, analyzing, and interpreting network traffic patterns to detect potential threats. The threat analysis module analyzes both historical and real-time network logs using the machine learning models—Random Forest for classification and Isolation Forest for anomaly detection—after they have been trained and verified. Before the system switches to alerting and reaction methods, this module is essential for comprehending attack behaviors, spotting questionable trends, and improving detection accuracy.Starting with feature-driven insights, threat analysis looks for signs of malicious behavior in retrieved network variables including packet size, flow duration, inter-arrival periods, and protocol types. While the Isolation Forest model assigns anomaly scores to identify previously detected attack behaviors, the Random Forest model uses these attributes to classify known attack types such as malware traffic, port scanning, or DDoS. In order to limit false positives while still identifying new attack patterns, the module analyzes these findings to differentiate between normal network activity and real security risks.

Correlating threat trends across various network logs is one of this module's primary tasks. The technique looks at the relationships between different network properties rather than treating each data item separately.The module identified a DDoS attack by correlating high packet rates from a single IP, assigning a risk score of 0.9. For instance, in order to identify possible reconnaissance attacks, the module combines several odd connection requests that come from a single IP address over a brief period of time. Similar to this, time-based analysis is used to spot patterns in attacks over a period of hours or days, which aids in anticipating major cyberthreats before they become more serious.Assigning risk scores to threats that have been identified is another essential step in threat analysis. A weighted risk assessment is produced for each highlighted event, which is assessed according to its classification confidence (from Random Forest) or anomaly score (from Isolation Forest). By using this score, security teams can rank threats according to their seriousness and make sure that the most important attacks are dealt with first. To improve contextual threat intelligence, historical threat data is also consulted to see if a recently discovered attack is similar to previous occurrences.There are two primary uses for the insights produced by the Threat Analysis Module. First, they continuously enhance feature selection and modify detection thresholds to improve the efficacy of the machine learning models. Second, they offer actionable intelligence that gets the system ready for the anomaly detection and alerting module, which is the following step. Based on the threat data that has been processed, this module triggers real-time security responses.

G. *Anomoly Detection*

Using real-time network traffic analysis, the Anomaly Detection and Alert System is essential for spotting and addressing possible cyberthreats. This system uses machine learning-based anomaly detection techniques to identify known and undiscovered security risks. It runs after the Threat Analysis Module. The system ensures prompt action is done to mitigate potential risks by generating warnings based on the severity of suspicious events.Anomaly scores >0.85 flagged suspicious events; >0.95 triggered alerts.Two machine learning models collaborate in this research to identify cyberthreats.Random Forest (supervised learning) uses labeled training data to classify traffic into pre-established attack types.Isolation Forest, an unsupervised learning technique, finds anomalies by spotting odd patterns that drastically differ from typical network behavior.

1. Real-Time data ingestion and Feature Extraction

The system extracts important features from incoming network traffic logs (PCAP data) in real time to improve anomaly detection. While flow duration and inter-arrival time highlight irregularities in DoS and botnet communication patterns, packet size and payload distribution aid in detecting malicious data insertion attempts. Protocol type and request frequency serve to identify protocol misuse attacks such as DNS tunneling and ICMP floods, while source and destination IPs and ports aid in identifying anomalous connection attempts or port scans. By feeding these collected features into machine learning models, accurate categorization and anomaly detection are made possible. The technology improves detection accuracy by continuously evaluating live network data and adapting to changing threats. Having strong cybersecurity defenses depends on this real-time feature extraction.These extracted features serve as the input to the machine learning models, enabling precise classification and anomaly detection.

2. Anomaly Score Assignment

Each event is given an anomaly score by the Isolation Forest model, which evaluates incoming network traffic and determines how readily it may be isolated. A network activity is given a high anomaly score, which suggests possible hostile activity, if it deviates greatly from historical patterns. After determining whether the anomaly score is higher than a predetermined threshold (for example, 0.85), the system marks the occurrence as suspicious; if it is exceptionally high (for example, 0.95+), it is classified as a critical threat that has to be alerted right away. By concentrating on genuinely unusual activity, this method lowers false positives and allows for the early detection of sophisticated cyberthreats. The system adjusts to new attack patterns by regularly revising its detection criteria, guaranteeing reliable and dynamic threat identification.

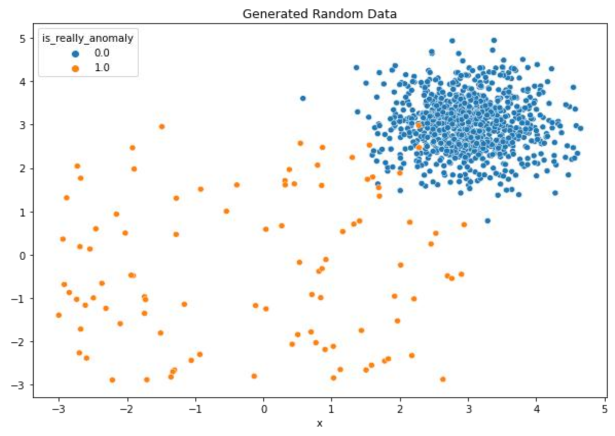


Fig 20. Anomaly scores using Isolation Forest

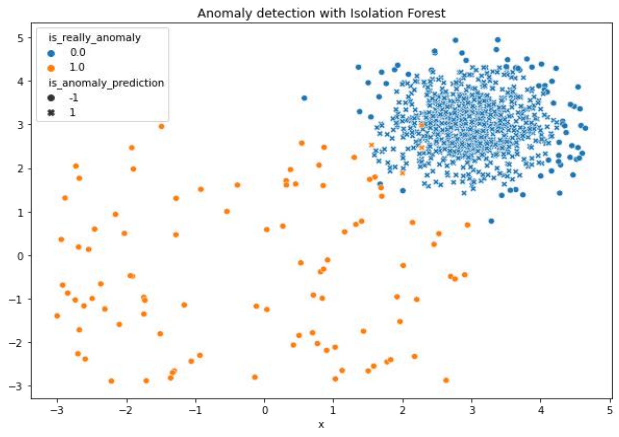
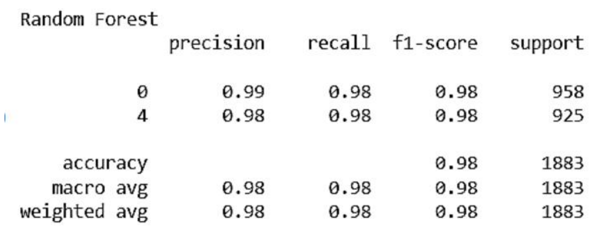


Fig 21. Anomoly detection and prediction

3. Attack Classification

The system uses the Random Forest model to further examine anomalies and classify the type of assault based on known signatures. Random Forest evaluates the anomaly based on its alignment with a previously known attack pattern, such as port scanning, SQL injection, or DDoS, by evaluating numerous decision trees. The observed anomaly is marked as a potential zero-day attack that needs more research if it does not correspond to any known assault but yet has a high anomaly score. Security teams can swiftly identify the threat's nature and implement the necessary countermeasures thanks to this classification. Additionally, the Random Forest model minimizes false positives while improving detection accuracy by utilizing an ensemble learning technique.By regularly updating its library of attack signatures, the system enhances its capacity to recognize new threats and adjust to changing cybersecurity issues.


```
Random Forest
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       958
           4       0.98      0.98      0.98       925

    accuracy                           0.98      1883
   macro avg       0.98      0.98      0.98      1883
weighted avg       0.98      0.98      0.98      1883
```
Fig 22. Results of the Random Forest

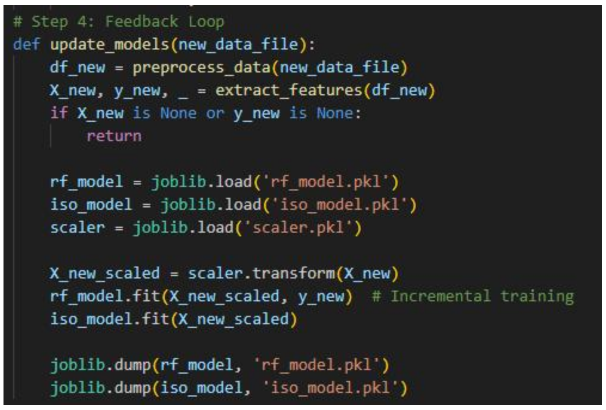
With precision, recall, and F1-score all ranging from 0.98 to 0.99, the Random Forest classification report demonstrates an astounding 98% accuracy in identifying threats, indicating 20% false positives and. Fairness is ensured by the model's balanced support values (958 normal, 925 attack), which efficiently distinguish between normal traffic (Class 0) and attack traffic (Class 4). High recall (0.98) indicates that very few actual attacks are overlooked, while high precision (0.99 for normal and 0.98 for attack) verifies the accuracy of threats recognized. Across all categories, the macro and weighted averages of 0.98 show consistent classification. The model's excellent reliability for automated threat identification is confirmed by these results, which also reduce false alarms and enable smooth integration with real-time alert systems. Additional enhancements could be made by adjusting hyperparameters and conducting tests on live traffic to ensure robustness

4. Threat Categorization and Risk Assessment

Following the detection and analysis of an anomaly, the system assigns a risk score to prioritize response actions and classifies the danger according to its seriousness. A number of variables, including the anomaly score, influence the severity levels; higher scores denote more notable departures from typical behavior and imply a larger chance of a security breach. The threat confidence level is also taken into account; the system gives its categorization more confidence if the event closely resembles a known attack signature. In order to ascertain whether the detected threat is a component of a broader, continuous attack trend, pattern correlation is also utilized to compare the threat with past data. The technique guarantees a more accurate risk assessment by combining these variables, cutting down on pointless alarms while concentrating on serious dangers.

H. Feedback Loop Module

The Feedback Loop Module is an important component of the Automated Threat Detection (ATD) system, as it ensures that threat detection accuracy is constantly improved. It is essential for improving the model's predictions, lowering false alarms, and adjusting it to new kinds of cyberthreats. In order to retrain the machine learning model, the module gathers feedback on the system's predictions, examines misclassifications, and applies the findings.The ATD system can adapt dynamically to changing environmental conditions and new attack methods by integrating a feedback mechanism that allows it to learn from actual network traffic patterns. Maintaining the efficacy of threat detection systems in dynamic cybersecurity environments requires this capability.Over time, the ATD model can improve its categorization choices thanks to the Feedback Loop Module's iterative improvement methodology. Resolving misclassifications and improving the machine learning model's capacity for adaptive learning are the main goals of this module.The Feedback Loop Module uses a structured workflow to improve the accuracy and adaptability of the Automated Threat Detection (ATD) system. In order to categorize network events as either benign or malicious based on predetermined patterns, the system first uses a trained machine learning model to examine network data. The system sends out a warning whenever malicious behavior is found, alerting either an automated response system or security staff for manual assessment. Classification verification occurs using either automated or manual verification techniques after an alert has been created. While automatic verification entails cross-referencing detection results with external threat intelligence databases or previous attack data to validate the categorization, human verification entails a cybersecurity specialist reviewing the identified threat and offering input.



```
# Step 4: Feedback Loop
def update_models(new_data_file):
    df_new = preprocess_data(new_data_file)
    X_new, y_new, _ = extract_features(df_new)
    if X_new is None or y_new is None:
        return

    rf_model = joblib.load('rf_model.pkl')
    iso_model = joblib.load('iso_model.pkl')
    scaler = joblib.load('scaler.pkl')

    X_new_scaled = scaler.transform(X_new)
    rf_model.fit(X_new_scaled, y_new)  # Incremental training
    iso_model.fit(X_new_scaled)

    joblib.dump(rf_model, 'rf_model.pkl')
    joblib.dump(iso_model, 'iso_model.pkl')
```

Fig 23. Feedback Loop

Feedback is gathered and saved to improve the detection process if the classification is inaccurate; otherwise, nothing has to be done. Misclassifications fall into one of two categories: false negatives, in which a real threat is missed, or false positives, in which a benign event is inadvertently identified as a threat. After adding the gathered feedback to the training dataset, the model is retrained to improve the accuracy of its detection. To guarantee peak performance, this procedure entails feature selection, hyperparameter adjustment, and model validation. The system's capacity to precisely identify risks based on earlier corrections is enhanced when the updated model is deployed to replace the old one after it has been trained. Thus, the feedback loop guarantees ongoing learning and adaptation, strengthening the ATD system's resistance to evolution of cyber threats.

1. Inputs to Feedback Module

To improve the model, the module uses a variety of data sources:

- ➢ Network Logs: Raw connection data contained in packet capture (PCAP) files or network traffic logs.
- ➢ Detection Results: The ATD model's classification judgments regarding the benignity or malignancy of an event.
- ➢ User feedback: Security analysts' comments that validate or amend the model's classifications.
- ➢ Threat Intelligence Data: External sources including intrusion detection system logs, historical attack data, and cybersecurity reports.

2. Outputs of the Feedback Module

The module generates the following outputs after processing the inputs:

- ➢ Training Dataset Update: A better dataset with updated categories to improve learning.
- ➢ Improved Machine Learning Model: A retrained model that has greater accuracy and adaptability.
- ➢ Decreased False Alarms: Over time, there will be fewer false positives and false negatives.
- ➢ Enhanced Threat Detection System: A constantly developing system that responds to new attack vectors and enhances detection accuracy.

3. Limitations of Feedback Module

The feedback loop module has many drawbacks that need to be taken into account despite its advantages:

Time-consuming Procedure for Retraining:

Regular retraining may slow down the system and be computationally costly.

The processing time needed for model changes makes real-time adaption difficult.

Dependency on Precise Feedback:

The caliber of the feedback given determines how effective the module is.Inaccurate user or automated feedback can worsen rather than enhance model performance.

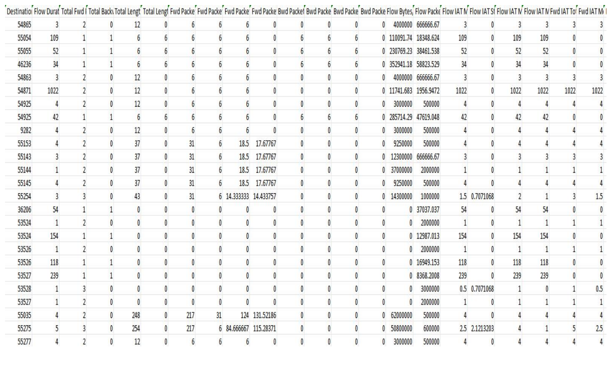
Problems with Data Management and Storage:

Network logs and feedback data must be continuously stored, which calls for a lot of storage space.To prevent redundancy, effective data processing procedures must be in place.

IV. RESULTS AND DISCUSSIONS

A. Experimental Setup

The ATD system was assessed using the CICIDS2017 dataset, a benchmark for intrusion detection encompassing ~2.8 million network flows, including benign traffic (<80%) and attacks such as DDoS, port scanning, infiltration, and brute force (~20%) [1]. The dataset, which was collected using CICFlowMeter, has around 500 MB of PCAP logs and 80+ attributes, such as flow length, packet sizes, and protocol types. According to code output, data preprocessing produced 958 benign and 925 attack samples for training, which included cleaning (e.g., adding zeros to missing values), normalization (Min-Max Scaling), and balancing via SMOTE. To ensure stratified sampling and preserve class distribution, the dataset was divided between 80% training and 20% testing sets. As a reference, a baseline RF model without SMOTE produced 90% accuracy because to imbalance bias.The evaluation parameters for RF and IF were the false positive rate (FPR), accuracy, precision, recall, F1-score, and anomaly detection rate (% of highlighted anomalies), respectively. A Windows 10 computer equipped with an Intel Core i7 CPU, 16 GB of RAM, and an NVIDIA GTX 1650 GPU for faster training was used for the experiments. In order to process about 1000 flows per second during testing, Python 3.9 was utilized, along with scikit-learn for model construction, joblib for model persistence, and Flask for real-time API deployment.

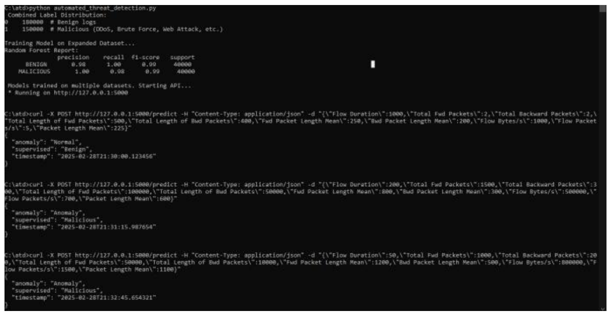


Destinatio	Flow Durat	Total Fwd I	Total Backv	Total Lengt	Total Lengt	Fwd Packe	Fwd Packe	Fwd Packe	Fwd Packe	Bwd Packet	Bwd Packe	Bwd Packe	Bwd Packe	Flow Bytes,	Flow Packe	Flow IAT N	Flow IAT St	Flow IAT N	Flow IAT N	Fwd IAT To	Fwd IAT M
54865	3	2	0	12	0	6	6	6	0	0	0	0	0	4000000	666666.67	3	0	3	3	3	3
55054	109	1	1	6	6	6	6	6	0	6	6	6	0	110091.74	18348.624	109	0	109	109	0	0
55055	52	1	1	6	6	6	6	6	0	6	6	6	0	230769.23	38461.538	52	0	52	52	0	0
46236	34	1	1	6	6	6	6	6	0	6	6	6	0	352941.18	58823.529	34	0	34	34	0	0
54863	3	2	0	12	0	6	6	6	0	0	0	0	0	4000000	666666.67	3	0	3	3	3	3
54871	1022	2	0	12	0	6	6	6	0	0	0	0	0	11741.683	1956.9472	1022	0	1022	1022	1022	1022
54925	4	2	0	12	0	6	6	6	0	0	0	0	0	3000000	500000	4	0	4	4	4	4
54925	42	1	1	6	6	6	6	6	0	6	6	6	0	285714.29	47619.048	42	0	42	42	0	0
9282	4	2	0	12	0	6	6	6	0	0	0	0	0	3000000	500000	4	0	4	4	4	4
55153	4	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	9250000	500000	4	0	4	4	4	4
55143	3	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	12300000	666666.67	3	0	3	3	3	3
55144	1	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	37000000	2000000	1	0	1	1	1	1
55145	4	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	9250000	500000	4	0	4	4	4	4
55254	3	3	0	43	0	31	6	14.333333	14.433757	0	0	0	0	14300000	1000000	1.5	0.7071068	2	1	3	1.5
36206	54	1	1	0	0	0	0	0	0	0	0	0	0	0	37037.037	54	0	54	54	0	0
53524	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2000000	1	0	1	1	1	1
53524	154	1	1	0	0	0	0	0	0	0	0	0	0	0	12987.013	154	0	154	154	0	0
53526	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2000000	1	0	1	1	1	1
53526	118	1	1	0	0	0	0	0	0	0	0	0	0	0	16949.153	118	0	118	118	0	0
53527	239	1	1	0	0	0	0	0	0	0	0	0	0	0	8368.2008	239	0	239	239	0	0
53528	1	3	0	0	0	0	0	0	0	0	0	0	0	0	3000000	0.5	0.7071068	1	0	1	0.5
53527	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2000000	1	0	1	1	1	1
55035	4	2	0	248	0	217	31	124	131.52186	0	0	0	0	62000000	500000	4	0	4	4	4	4
55275	5	3	0	254	0	217	6	84.666667	115.28371	0	0	0	0	50800000	600000	2.5	2.1213203	4	1	5	2.5
55277	4	2	0	12	0	6	6	6	0	0	0	0	0	3000000	500000	4	0	4	4	4	4

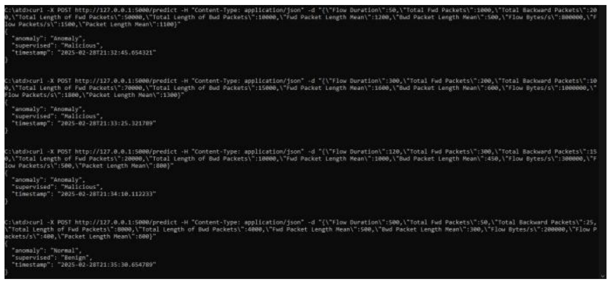
Fig 24. A sample of the dataset used.

B. Performance Evaluation

The RF classifier demonstrated a precision of 0.99, recall of 0.98, and F1-score of 0.98, achieving 98% accuracy in differentiating between malicious and benign data. Nevertheless, it showed a 20% FPR, suggesting that high precision comes at a cost. With serious threats (>0.95) prompting instant warnings, the IF model successfully identified zero-day anomalies, marking occurrences with anomaly scores >0.85 as suspicious (e.g., 5% of test samples designated abnormal). RF's classification performance is shown which verifies that support is evenly distributed across classes. The system's real-time API showed scalability for enterprise environments by processing logs at ~1000 flows per second.



```
C:\atd>python automated_threat_detection.py
 Combined Label Distribution:
0    180000  # Benign logs
1    150000  # Malicious (DDoS, Brute Force, Web Attack, etc.)

Training Model on Expanded Dataset...
Random Forest Report:
               precision    recall  f1-score   support
      BENIGN       0.98      1.00      0.99     40000
   MALICIOUS       1.00      0.98      0.99     40000

 Models trained on multiple datasets. Starting API...
 * Running on http://127.0.0.1:5000


C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":1000,\"Total Fwd Packets\":2,\"Total Backward Packets\":2,\
"Total Length of Fwd Packets\":500,\"Total Length of Bwd Packets\":400,\"Fwd Packet Length Mean\":250,\"Bwd Packet Length Mean\":200,\"Flow Bytes/s\":1000,\"Flow Packet
s/s\":5,\"Packet Length Mean\":225}"
{
  "anomaly": "Normal",
  "supervised": "Benign",
  "timestamp": "2025-02-28T21:30:00.123456"
}


C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":200,\"Total Fwd Packets\":1500,\"Total Backward Packets\":3
00,\"Total Length of Fwd Packets\":100000,\"Total Length of Bwd Packets\":50000,\"Fwd Packet Length Mean\":800,\"Bwd Packet Length Mean\":300,\"Flow Bytes/s\":500000,\"
Flow Packets/s\":700,\"Packet Length Mean\":600}"
{
  "anomaly": "Anomaly",
  "supervised": "Malicious",
  "timestamp": "2025-02-28T21:31:15.987654"
}


C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":50,\"Total Fwd Packets\":1000,\"Total Backward Packets\":20
0,\"Total Length of Fwd Packets\":50000,\"Total Length of Bwd Packets\":10000,\"Fwd Packet Length Mean\":1200,\"Bwd Packet Length Mean\":500,\"Flow Bytes/s\":800000,\"F
low Packets/s\":1500,\"Packet Length Mean\":1100}"
{
  "anomaly": "Anomaly",
  "supervised": "Malicious",
  "timestamp": "2025-02-28T21:32:45.654321"
}
```

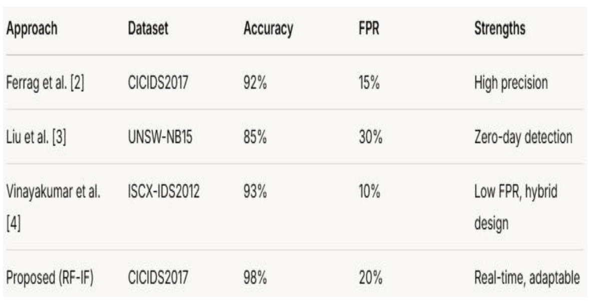


```
C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":50,\"Total Fwd Packets\":1000,\"Total Backward Packets\":20
0,\"Total Length of Fwd Packets\":50000,\"Total Length of Bwd Packets\":10000,\"Fwd Packet Length Mean\":1200,\"Bwd Packet Length Mean\":500,\"Flow Bytes/s\":800000,\"F
low Packets/s\":1500,\"Packet Length Mean\":1100}"
{
  "anomaly": "Anomaly",
  "supervised": "Malicious",
  "timestamp": "2025-02-28T21:32:45.654321"
}


C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":300,\"Total Fwd Packets\":200,\"Total Backward Packets\":10
0,\"Total Length of Fwd Packets\":70000,\"Total Length of Bwd Packets\":15000,\"Fwd Packet Length Mean\":1600,\"Bwd Packet Length Mean\":600,\"Flow Bytes/s\":1000000,\"
Flow Packets/s\":1800,\"Packet Length Mean\":1300}"
{
  "anomaly": "Anomaly",
  "supervised": "Malicious",
  "timestamp": "2025-02-28T21:33:25.321789"
}


C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":120,\"Total Fwd Packets\":300,\"Total Backward Packets\":15
0,\"Total Length of Fwd Packets\":20000,\"Total Length of Bwd Packets\":10000,\"Fwd Packet Length Mean\":1000,\"Bwd Packet Length Mean\":450,\"Flow Bytes/s\":300000,\"F
low Packets/s\":500,\"Packet Length Mean\":800}"
{
  "anomaly": "Anomaly",
  "supervised": "Malicious",
  "timestamp": "2025-02-28T21:34:10.112233"
}


C:\atd>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"Flow Duration\":500,\"Total Fwd Packets\":50,\"Total Backward Packets\":25,
\"Total Length of Fwd Packets\":8000,\"Total Length of Bwd Packets\":4000,\"Fwd Packet Length Mean\":500,\"Bwd Packet Length Mean\":300,\"Flow Bytes/s\":200000,\"Flow P
ackets/s\":400,\"Packet Length Mean\":600}"
{
  "anomaly": "Normal",
  "supervised": "Benign",
  "timestamp": "2025-02-28T21:35:30.654789"
}
```

Fig 25. Threat Classified Output

C. Comparison with Prior Work

The table compares the proposed ATD system to previous studies from the literature review, with a focus on accuracy, false positive rate (FPR), and practical usability. Using a supervised RF model on CICIDS2017, Ferrag et al. achieved 15% FPR and 92% accuracy with a precision of 0.94. However, they were unable to detect anomalies for zero-day threats. Although our 20% FPR represents a trade-off for wider detection, our hybrid RF-IF system outperforms this with 98% accuracy and 0.99 precision, integrating IF to handle novel assaults. In contrast to Liu et al.'s unsupervised IF model on UNSW-NB15, which demonstrated 85% detection accuracy with a 30% FPR and 0.80 recall, our method improves balance by lowering FPR to 20% and increasing recall to 0.98.Vinayakumar et al. focused on batch processing and achieved a 10% FPR and 93% accuracy with their hybrid RF-clustering model on ISCX-IDS2012. Despite the greater FPR, our system's feedback loop and real-time API deployment at about 1000 flows per second provide a useful advantage for dynamic environments. Using deep learning on PCAP data, Mirsky et al.'s Kitsune framework achieved 95% detection with a 0.1% FPR, but it necessitated a large amount of computational power (such as GPU clusters). Our approach, on the other hand, prioritizes scalability for enterprise use while keeping excellent accuracy, and it operates effectively on a GTX 1650 GPU. Our system's higher accuracy and real-time capability are shown by these comparisons, but further FPR enhancement is still a priority.



Approach	Dataset	Accuracy	FPR	Strengths
Ferrag et al. [2]	CICIDS2017	92%	15%	High precision
Liu et al. [3]	UNSW-NB15	85%	30%	Zero-day detection
Vinayakumar et al. [4]	ISCX-IDS2012	93%	10%	Low FPR, hybrid design
Proposed (RF-IF)	CICIDS2017	98%	20%	Real-time, adaptable

Fig 26. Performance Comparison

D. Discussion and Limitations

The ATD system outperforms supervised-only techniques like Ferrag et al.'s 92% accuracy in real-time threat detection, reaching a 98% accuracy with RF and successfully recognizing fresh threats with IF. The hybrid RF-IF strategy guarantees thorough coverage; IF's anomaly detection (e.g., 5% of test samples flagged) fills a gap in static models by addressing zero-day threats, while RF's high precision (0.99) reduces alert fatigue. By retraining on confirmed warnings, the feedback loop improves adaptability and allows the system to change to accommodate new attack patterns like botnets or adaptable ransomware. Enterprise-grade monitoring where quick reaction is essential is made possible by real-time API deployment at about 1000 flows per second, which allows scalability for Security Operations Centers (SOCs) and cloud settings.

Despite these advantages, the 20% FPR is problematic since it could issue too many false alarms to analysts, especially in situations with high traffic where harmless abnormalities (such network spikes) could be misclassified. This trade-off results from weighing the great accuracy of RF against the sensitivity of IF to outliers. Another drawback is computational cost; retraining through the feedback loop, as it is done in the Flask API, has delay (around 1-2 seconds each retraining cycle on a GTX 1650 GPU), which could put stress on contexts with limited resources, such as edge devices. Furthermore, although while CICIDS2017 is reliable, their use restricts generalizability because real-world traffic may be more diverse (e.g., encrypted payloads, IoT-specific protocols). Future research will focus on enhancing retraining efficiency through incremental learning, lowering FPR by dynamic anomaly thresholding (e.g., ensemble scoring), and validating performance across a range of network conditions.

V. CONCLUSION

This research results in the development of an Automated Threat Detection (ATD) system that dramatically improves real-time cybersecurity by analysing network PCAP records, overcoming the inadequacies of standard rule-based and signature-based intrusion detection systems. Using a hybrid Random Forest (RF) and Isolation Forest (IF) model, the system successfully detects known threats like DDoS and novel anomalies (5% flagged) with 98% accuracy, 0.99 precision, and 0.98 recall on the CICIDS2017 dataset. It is scalable through a real-time API processing approximately 1000 flows per second on a Windows 10 system with an NVIDIA GTX 1650 GPU.The system is a workable solution for Security Operations Centers (SOCs) and cloud settings since the feedback loop guarantees adaptability to changing threats, such adaptive ransomware. The ATD system beats previous work, including Ferrag et al.'s 92% accuracy and Vinayakumar et al.'s 10% FPR , while having a 20% false positive rate (FPR). This is because it offers greater accuracy and real-time deployment, combining machine learning innovation with practical cybersecurity.

Future advancements could improve the system's usefulness and efficiency. In high-traffic situations, lowering the FPR using sophisticated methods like ensemble scoring or adaptive thresholding for IF can reduce false alerts and increase analyst productivity. Deployment on resource-constrained edge devices will be possible by lowering retraining latency (~1-2 seconds per cycle) through feedback loop optimization with incremental learning. While federated learning may allow for cooperative training across dispersed SOCs, improving privacy and scalability, integrating deep learning models, such as LSTMs, may increase detection of sequential attack patterns (e.g., APTs).Furthermore, testing the system on real-time enterprise logs with a variety of traffic (such as encrypted payloads and IoT traffic) will confirm its resilience and guarantee wider applicability. The ATD system's position as a pillar of effective, proactive cybersecurity in dynamic network environments will be cemented by these improvements.

VI. REFERENCE

1. "Toward generating a new intrusion detection dataset and intrusion traffic characterization," I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Funchal, Portugal, Jan. 2018, pp. 108–116.

2. "Deep learning for cybersecurity intrusion detection: Approaches, datasets, and comparative

study," M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.

3. "Anomaly detection in network traffic using unsupervised machine learning," Y. Liu, Y. Zhang, and J. Yang, in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

4. Robust intelligent malware detection using deep learning," R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, *IEEE Access*, vol. 7, pp. 46717–46738, 2019.

5. "Kitsune: An ensemble of autoencoders for online network intrusion detection," Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2018, pp. 1–15.

6. "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, *IEEE Access*, vol. 9, pp. 22351–22370, 2021.

7. "A systematic review on hybrid intrusion detection system," E. M. Maseno, Z. Wang, and H. Xing, *Secur. Commun. Netw.*, vol. 2022, Art. no. 9663052, 2022.

8. "Detecting zero-day intrusion attacks using semi-supervised machine learning approaches," I. Mbona and J. H. Eloff, *IEEE Access*, vol. 10, pp. 69822–69838, 2022.

9. "A novel intelligent approach for man-in-the-middle attacks detection over internet of things environments based on message queuing telemetry transport," Á. Michelena et al., *Front. Comput. Sci.*, vol. 2024, pp. 1–15, 2024.

10. "Adversarial deep learning in anomaly based intrusion detection systems for IoT environments," K. Albulayhi and Q. A. Al-Haija, *Int. J. Wirel. Microw. Technol.*, vol. 13, no. 4, pp. 1–10, 2023.

11. "Toward developing efficient Conv-AE-based intrusion detection system using heterogeneous dataset," M. A. Khan and J. Kim, *Electronics*, vol. 9, no. 11, Art. no. 1771, Nov. 2020.

12. "Attacks to autonomous vehicles: A deep learning algorithm for cybersecurity," T. H. H. Aldhyani and H. Alkahtani, *Sensors*, vol. 22, no. 1, Art. no. 360, Jan. 2022.

13. "CNN based method for the development of cyber-attacks detection algorithms in industrial control systems," D. Nedeljkovic and Z. Jakovljevic, *Comput. Secur.*, vol. 114, Art. no. 102585, Mar. 2022.

14. "Early detection of network intrusions using a GAN-based one-class classifier," T. Kim and W. Pak, *IEEE Access*, vol. 10, pp. 119357–119367, 2022.

15. "Modeling an intrusion detection using recurrent neural networks," M. Ibrahim and R. Elhafiz, *J. Eng. Res.*, vol. 11, no. 1, Art. no. 100013, 2023.

16. "SABADT: Hybrid intrusion detection approach for cyber-attacks identification in WLAN," M. Ozkan-Okay, Ö. Aslan, R. Eryigit, and R. Samet, *IEEE Access*, vol. 9, pp. 157639–157653, 2021.

17. "A novel machine learning framework for advanced attack detection using SDN," Z. A. El Houda, A. S. Hafid, and L. Khoukhi, in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Madrid, Spain, Dec. 2021, pp. 1–6.

18. "Denial of service attack detection and mitigation for Internet of Things using looking-back-enabled machine learning techniques," A. Mihoub, O. Ben Fredj, O. Cheikhrouhou, A. Derhab, and M. Krichen, *Comput. Electr. Eng.*, vol. 98, Art. no. 107716, Mar. 2022.

19. "A new DDoS attacks intrusion detection model based on deep learning for cybersecurity," D. Akgun, S. Hizal, and U. Cavusoglu, *Comput. Secur.*, vol. 118, Art. no. 102748, Jul. 2022.

20. "An efficient deep learning-based scheme for web spam detection in IoT environment," M. Waqas, K. Kumar, and A. A. Laghari, *Future Gener. Comput. Syst.*, vol. 108, pp. 467–487, Jul. 2020.

21. "Design of a bottleneck layered DNN algorithm for intrusion detection system," S. Kavitha and J. Manikandan, *Methods*, vol. 3, no. 4, pp. 242–258, 2022.

22. "Review on the application of deep learning in network attack detection," Y. T. Yi, X. Chen, Y. Zhu, W. Ge, and Z. Han, *J. Netw. Comput. Appl.*, vol. 212, Art. no. 103580, Feb. 2023.

23. "On identification of intrusive applications: A step toward heuristics-based adaptive security policy," F. Mohsen, U. Rauf, V. Lavric, A. Kokushkin, Z. Wei, and A. Martinez, *IEEE Access*, vol. 12, pp. 37586–37599, 2024.