

Intelligent Network Intrusion Detection System

Nikhil | Shivam

Roll No: 220BTCCSE062 | 220BTCCSE056

B.Tech VIII Semester | Course Code: 15BCS-0MP42J

Guide: Ms. Shiksha

Abstract— With the rapid expansion of digital networks and the increasing sophistication of cyber threats, ensuring robust network security has become a critical challenge. Traditional Intrusion Detection Systems (IDS) primarily rely on signature-based or rule-based techniques, which are effective for known attack patterns but often fail to detect unknown or evolving threats. This limitation leads to issues such as high false positive rates and reduced detection accuracy in dynamic network environments.

This research presents a comprehensive and lightweight Intrusion Detection and Prevention System (IDPS) developed using Python, integrating both rule-based detection and machine learning-based anomaly detection techniques. The proposed system captures live network traffic using tools such as Scapy and PyShark, performs protocol analysis for ICMP, HTTP, and DNS traffic, and applies a detection engine to identify malicious activities. The system is further enhanced by incorporating learning algorithms including Multilayer Perceptron (MLP), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression, and Autoencoder models. These models utilize concepts such as hidden layers, activation functions, Mean Square Error (MSE), and optimization techniques like the Adam optimizer to improve detection performance.

In addition, anomaly detection techniques such as Isolation Forest are employed to identify deviations from normal network behavior based on metrics like average path length. The system evaluates performance using key metrics including Accuracy, Precision, Recall, False Positive Rate (FPR), and F1 Score. Experimental results demonstrate that the hybrid approach significantly improves detection capability while reducing false positives compared to traditional systems.

Furthermore, the system includes a real-time alert mechanism, automated prevention module for blocking malicious IP addresses, and a Flask-based dashboard for monitoring system activity and visualizing results. The lightweight design ensures low resource consumption, making it suitable for students, small organizations, and learning environments.

Overall, this research contributes to the development of an efficient, scalable, and intelligent Network Intrusion Detection System by combining rule-based logic with advanced machine learning techniques, providing a foundation for future enhancements in adaptive cybersecurity solutions.

Keywords— *Intrusion Detection System (IDS); Intrusion Prevention System (IPS); Network Intrusion Detection Systems (NIDS); Anomaly Detection; Machine Learning; Artificial Neural Network (ANN); Multilayer Perceptron (MLP); Autoencoder; Isolation Forest; Support Vector Machine (SVM); K-Nearest Neighbor (KNN); Logistic Regression; Mean Square Error (MSE); Adam Optimizer; Radial Basis Function (RBF); Performance Metrics; Accuracy; False Positive Rate (FPR); F1 Score; Denial of Service (DoS); Attack Patterns; Network Security; Real-Time Detection; Cybersecurity.*

I. INTRODUCTION

In today's digital era, protecting network systems has become more challenging than ever. With the rapid adoption of cloud technologies, online services, and interconnected devices, networks are constantly exposed to a wide range of cyber threats. Network Intrusion Detection Systems (NIDS) have become an important part of cybersecurity frameworks, helping organizations monitor traffic and identify potential attacks. However, traditional security solutions such as firewalls and signature-based intrusion detection systems are no longer sufficient on their own. These systems mainly depend on known attack patterns, which makes them less effective when dealing with new or unknown threats such as zero-day attacks and constantly evolving malware.[1],[2]

To tackle these limitations, researchers and practitioners have increasingly turned toward anomaly detection techniques. Instead of relying on predefined signatures, anomaly-based systems focus on learning what "normal" network behavior looks like and then flagging any unusual activity. This approach allows the system to detect

previously unseen attacks, making it more adaptable in dynamic environments. In this context, machine learning has played a major role, offering tools that can analyze large volumes of network data and uncover hidden patterns that may not be obvious through traditional methods.

Among various machine learning techniques, Artificial Neural Networks (ANN) have shown strong potential due to their ability to model complex relationships in data. Models such as Multilayer Perceptron (MLP), Support Vector Machines (SVM), Logistic Regression, and Isolation Forest are commonly used to classify network traffic and identify anomalies. While these methods improve detection capabilities, they also introduce certain challenges. High computational cost, dependency on large datasets, and the possibility of generating excessive false alarms are some of the key issues that need to be addressed.

Considering these challenges, this research proposes a lightweight and efficient Intrusion Detection and Prevention System (IDPS) implemented using Python. The idea is to combine rule-based detection with machine learning techniques to create a balanced and practical solution. Rule-based methods are fast and effective for detecting known threats, while machine learning models

help identify unknown or suspicious activities. By integrating both approaches, the system aims to improve detection accuracy while keeping false positives under control.

In addition, special attention is given to optimizing performance so that the system can operate efficiently even in environments with limited resources. The overall goal is to design a solution that is not only intelligent but also practical for real-world use. This combination of adaptability, efficiency, and accuracy makes the proposed approach a meaningful contribution to modern network security systems.

II. BACKGROUND AND EXISTING APPROACHES

2. Literature Review

Intrusion Detection Systems (IDS) have undergone significant transformation over the years, evolving from simple rule-based mechanisms to more advanced and intelligent systems powered by machine learning. In the early stages, IDS primarily relied on predefined rules and signatures to identify malicious activities. While these systems were effective against known threats, they struggled to cope with the growing sophistication and diversity of modern cyber attacks. As attackers began to develop more adaptive and stealthy techniques, the limitations of traditional IDS became increasingly apparent, prompting researchers to explore more flexible and intelligent approaches.

With the advancement of data-driven technologies, anomaly detection emerged as a promising alternative. Unlike traditional systems, anomaly-based IDS focus on understanding normal network behavior and identifying deviations from it. This shift marked an important step toward detecting previously unseen or zero-day attacks. Over time, various machine learning algorithms have been proposed and implemented to improve the performance of intrusion detection systems. These approaches aim to enhance detection accuracy, reduce false alarms, and ensure scalability in real-world environments.[1],[2]

2.1 Existing Approaches

One of the earliest and most widely used approaches in IDS is the rule-based system. These systems operate by comparing network activity against a database of known attack signatures. Their primary strength lies in their speed and reliability when dealing with previously identified threats. However, their dependency on predefined rules makes them ineffective against new or evolving attacks. Maintaining and updating the rule database is also a continuous challenge, requiring constant monitoring and manual intervention.

Anomaly-based IDS address this limitation by focusing on deviations from normal behavior rather than relying on known signatures. These systems typically use statistical methods or machine learning models to learn patterns of legitimate network activity. When unusual behavior is detected, it is flagged as a potential threat. Although this approach significantly improves the ability to detect unknown attacks, it often comes at the cost of higher false

positive rates. Legitimate but uncommon activities may be incorrectly classified as malicious, which can lead to unnecessary alerts and increased workload for system administrators.

To overcome the drawbacks of individual approaches, hybrid systems have been introduced. These systems combine rule-based and anomaly-based techniques to leverage the strengths of both. In a hybrid IDS, rule-based methods are used for quick detection of known threats, while anomaly detection models identify unknown or suspicious activities. This combination helps in improving overall detection accuracy while reducing false positives. As a result, hybrid systems are increasingly being considered a practical solution for modern network security challenges.[7],[8]

2.2 Comparative Analysis

A wide range of machine learning models has been explored for intrusion detection, each offering unique advantages and facing certain limitations. Among them, Support Vector Machines (SVM) are known for their strong classification capabilities and ability to handle high-dimensional data. They often achieve high accuracy, especially in well-structured datasets. However, SVM models can become computationally expensive and slow when applied to large-scale network traffic, making them less suitable for real-time applications without optimization.[3],[4]

- K-Nearest Neighbors (KNN) is another commonly used algorithm due to its simplicity and ease of implementation. It classifies data points based on their proximity to neighboring instances, making it intuitive and effective in certain scenarios. Despite its simplicity, KNN suffers from high computational costs, particularly when dealing with large datasets, as it requires distance calculations for every new data point.
- Isolation Forest is specifically designed for anomaly detection and has gained popularity due to its efficiency and scalability. It works by isolating anomalies rather than profiling normal data, which makes it faster compared to many traditional methods. However, one of its limitations is interpretability, as it can be difficult to explain why a particular instance is classified as anomalous.
- Autoencoders, a type of neural network, have shown very high effectiveness in detecting unknown attacks. They work by learning compressed representations of normal data and identifying anomalies based on reconstruction errors. While they offer strong performance, they require proper training and tuning, as well as sufficient computational resources, which may limit their usability in lightweight systems.
- Logistic Regression, on the other hand, is a simple yet efficient model that performs well in binary classification tasks. It is fast, easy to implement, and requires fewer computational resources. However, its ability to capture complex patterns is limited, which can affect its performance in highly dynamic and non-linear network environments.

Overall, the literature suggests that no single model is sufficient to address all challenges in intrusion detection. This has led to increasing interest in hybrid and optimized approaches that combine multiple techniques to achieve better performance, efficiency, and adaptability in real-world scenarios.

Model	Accuracy Level	Key Advantage	Limitation
Support Vector Machine (SVM)	High	Strong classification in high-dimensional data	Computationally expensive for large datasets
K-Nearest Neighbors (KNN)	Medium	Simple and easy to implement	High computation cost during prediction
Isolation Forest	High	Efficient for anomaly detection	Less interpretable results
Autoencoder (Neural Network)	Very High	Effective in detecting unknown attacks	Requires extensive training and tuning
Logistic Regression	Medium	Fast and computationally efficient	Limited ability to model complex relationships

III. SYSTEM DESIGN

3.1 Packet Capture

The first stage of the proposed system involves capturing live network traffic from the host machine. This is achieved using Python-based libraries such as Scapy and PyShark, which provide powerful packet sniffing and inspection capabilities. Scapy allows low-level packet manipulation and real-time capture, while PyShark leverages Wireshark's decoding engine for detailed protocol analysis. The system continuously monitors incoming and outgoing packets across the network interface and filters relevant traffic for further processing. This real-time capture mechanism ensures that no critical data packets are missed, forming the foundation for accurate intrusion detection.

3.2 Protocol Analysis

After capturing the packets, the system performs protocol-level classification and analysis. Each packet is examined to determine its associated protocol, such as ICMP, HTTP, or DNS. This classification is essential because different attack types are associated with specific protocols. For instance, ICMP packets may indicate flooding attacks, while HTTP packets may contain malicious payloads. By separating packets based on protocol type, the system enables targeted analysis, improving detection accuracy and

efficiency. Key attributes such as source IP, destination IP, port numbers, and payload data are extracted during this stage.

3.3 Detection Engine

The detection engine is the core component responsible for identifying malicious activities. It primarily uses a rule-based approach, where predefined thresholds and conditions are applied to detect suspicious patterns. For example, an unusually high number of ICMP packets within a short time interval may indicate a flood attack. Similarly, abnormal HTTP requests or excessive DNS queries can signal potential threats. These rules are designed based on known attack patterns and network behavior. The detection engine continuously evaluates incoming traffic against these conditions and flags any deviations from normal behavior.

3.4 Alert System

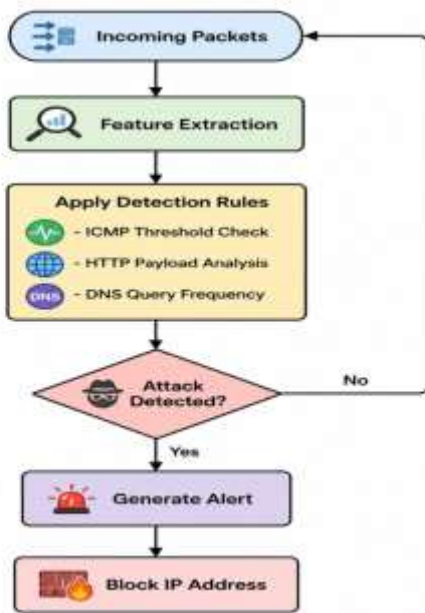
Once a potential threat is detected, the system generates an alert to notify the user or administrator. Each alert contains detailed information, including the timestamp, source IP address, protocol type, and nature of the detected attack. These alerts are stored in a structured database, allowing for easy retrieval and analysis. The alert system plays a crucial role in maintaining a log of network activity and provides valuable insights for further investigation and response.

3.5 Prevention Module

In addition to detection, the system incorporates a prevention mechanism to actively block malicious entities. When a suspicious IP address is identified, it is automatically added to firewall rules, preventing further communication with the system. This real-time blocking capability enhances network security by minimizing the impact of ongoing attacks. The prevention module ensures that identified threats are not only detected but also mitigated promptly.

3.6 Dashboard

To provide a user-friendly interface, the system includes a web-based dashboard developed using Flask. The dashboard displays real-time alerts, graphical representations of network activity, and system performance metrics. Users can monitor detected threats, view logs, and analyze trends through interactive visualizations. This interface simplifies system management and enables quick decision-making, making the overall solution accessible even to users with limited technical expertise.



IV. METHODOLOGY

This section explains how the proposed Intrusion Detection and Prevention System (IDPS) was built, from collecting network data to deploying detection and prevention mechanisms. The overall design follows a practical, step-by-step approach so that the system remains lightweight while still delivering reliable results in real-world conditions.

4.1 Data Collection

The foundation of the system lies in capturing live network traffic. Instead of relying on static datasets, real-time packets are collected directly from the network to better reflect actual operating conditions. Packet capturing is implemented using the Python library **Scapy**, which allows low-level access to network packets and supports flexible filtering.

To ensure efficiency, only relevant traffic is captured by applying filters based on protocol type and packet characteristics. This reduces unnecessary data processing and helps maintain system performance. The collected packets are then parsed to extract useful information such as source and destination IP addresses, protocol details, and packet structure. This raw data serves as the input for further analysis and model training.

4.2 Feature Engineering

Once packets are collected, the next step is transforming raw data into meaningful features that can be used by detection models. Careful feature selection plays a critical role in improving detection accuracy while keeping computational overhead low. In this system, a set of lightweight yet informative features is extracted, including:

- **Packet Length:** Helps identify unusual packet sizes that may indicate malicious payloads.
- **Protocol Type:** Distinguishes between TCP, UDP, ICMP, and other protocols to detect abnormal usage patterns.

- **Time Interval:** Measures the time gap between consecutive packets, useful for identifying flooding or burst attacks.

- **Request Frequency:** Tracks how frequently requests are sent from a particular source, helping detect suspicious behavior such as brute-force attempts.

These features are normalized and structured into a dataset that can be processed efficiently by machine learning models. The focus remains on selecting features that balance detection capability with system performance.

4.3 Machine Learning Models

To enhance detection capabilities, multiple machine learning models are integrated into the system. Each model contributes differently, allowing the system to handle a variety of attack scenarios.[4]

- **Multilayer Perceptron (MLP):** A neural network model that uses multiple hidden layers and activation functions such as ReLU and Sigmoid to capture complex patterns in network traffic. It is particularly useful for identifying non-linear relationships in data.

- **Autoencoder:** This model follows an encoder-decoder structure, where normal network behavior is learned during training. Anomalies are detected when the reconstruction error, typically measured using Mean Squared Error (MSE), exceeds a predefined threshold.

- **Isolation Forest:** Designed specifically for anomaly detection, this model isolates unusual data points using randomly generated trees. It is efficient and well-suited for handling large datasets with minimal computational cost.

- **Support Vector Machine (SVM):** SVM works by finding an optimal hyperplane that separates normal and malicious traffic. It is effective in high-dimensional spaces but may require optimization for large-scale deployment.

- **Logistic Regression:** A simple yet efficient model used for binary classification. It applies a sigmoid function to predict the probability of a packet being malicious or benign.

- **K-Nearest Neighbor (KNN):** This model classifies data based on proximity to neighboring points. While easy to implement, it can be computationally expensive during real-time prediction.

4.4 System Implementation and Workflow

The system is developed in a modular manner to ensure clarity and scalability. The workflow begins with packet capturing using Scapy, followed by filtering to retain only relevant traffic. Protocol analysis is then performed to extract key details such as IP addresses and packet types.

After feature extraction, a **rule-based detection layer** is applied as the first level of defense. This layer uses

predefined thresholds (e.g., excessive request frequency or abnormal packet size) to quickly identify known attack patterns. The advantage of this step is its speed and low computational cost.

Next, the processed data is passed through the machine learning models for deeper analysis. These models help identify complex and previously unseen attack patterns that cannot be detected using simple rules alone. By combining rule-based and learning-based approaches, the system achieves a balance between speed and intelligence.

Detected alerts are stored in an **SQLite database**, which acts as a lightweight logging system. This allows easy retrieval and analysis of past events. For user interaction, a **Flask-based dashboard** is implemented to display real-time monitoring results, including detected threats and system status.

Finally, a **prevention module** is integrated to actively respond to detected threats. Malicious IP addresses are automatically blocked, ensuring that the system not only detects attacks but also mitigates them in real time.

V. Mathematical Foundations

The proposed system relies on a few key mathematical measures to evaluate performance and support model behavior. The Mean Square Error (MSE) is used to quantify the difference between actual and predicted values, making it especially useful in models like autoencoders where reconstruction quality matters.[5]

$(x + a)^n = \sum_{k=0}^n \binom{n}{k} x^k a^{n-k}$ The F1 Score provides a balanced evaluation by combining precision and recall into a single metric, which is important when dealing with imbalanced datasets.

Finally, the logistic function is applied in classification tasks to convert outputs into probability values between 0 and 1.

$$(1 + x)^n = 1 + \frac{nx}{1!} + \frac{n(n-1)x^2}{2!} + \dots$$

VI. Results and Analysis

The performance of the proposed Intrusion Detection and Prevention System (IDPS) was evaluated through a series of practical tests designed to simulate real-world network attacks. The system follows a structured workflow, beginning with packet capture and ending with prevention actions, as illustrated in the detection flow diagram. Each stage of the system was analyzed to understand its effectiveness, responsiveness, and overall reliability.[2],[4]

6.1 Detection Capability

The detection engine performed efficiently in identifying suspicious network activities. By applying rule-based conditions such as ICMP packet thresholds, HTTP payload inspection, and DNS query monitoring, the system was able to detect different types of attacks in real time.

For instance, during the ICMP flood test, a large number of ping requests were sent within a short period. The system

quickly recognized this unusual spike in traffic and flagged it as a potential attack. Similarly, abnormal HTTP requests containing irregular patterns were detected successfully. DNS-based anomalies, such as repeated query bursts, were also identified when they exceeded normal limits.

These results indicate that the system is capable of recognizing common attack patterns effectively. Since the detection is based on predefined rules, it responds quickly without requiring complex computations.

6.2 Alert Generation and Logging

Once suspicious activity was identified, the system generated alerts almost instantly. Each alert included useful details such as the time of detection, source IP address, type of protocol involved, and nature of the attack. These alerts were stored in a database, making it easy to review past incidents.

The alert system proved to be reliable and consistent throughout the testing phase. It ensured that no major threat went unnoticed. Additionally, maintaining a log of detected events can help administrators analyze trends and understand how frequently certain types of attacks occur.

6.3 Prevention Effectiveness

One of the key strengths of this system is its ability not only to detect attacks but also to respond to them. After identifying a malicious source, the system automatically blocked the corresponding IP address using firewall rules.

This immediate action helped in stopping further malicious traffic from reaching the system. During testing, once an IP was blocked, subsequent packets from the same source were successfully prevented. This reduced the overall load on the system and minimized potential damage.

The combination of detection and prevention makes the system more practical and useful compared to detection-only solutions.

6.4 System Performance

The system was designed to be lightweight, and the results support this objective. Even when handling continuous network traffic, CPU usage remained stable and did not show any significant spikes. This indicates that the system can operate efficiently in real-time environments without consuming excessive resources.

Memory usage was also kept under control since the system processes packets dynamically instead of storing large volumes of data. This makes it suitable for smaller setups such as student projects, labs, or local networks where hardware resources may be limited.

6.5 False Positives

While the system performed well overall, a small number of false positives were observed. These occurred when legitimate network activity temporarily resembled suspicious behavior. For example, a sudden increase in DNS queries from a normal application could sometimes be flagged as an anomaly.

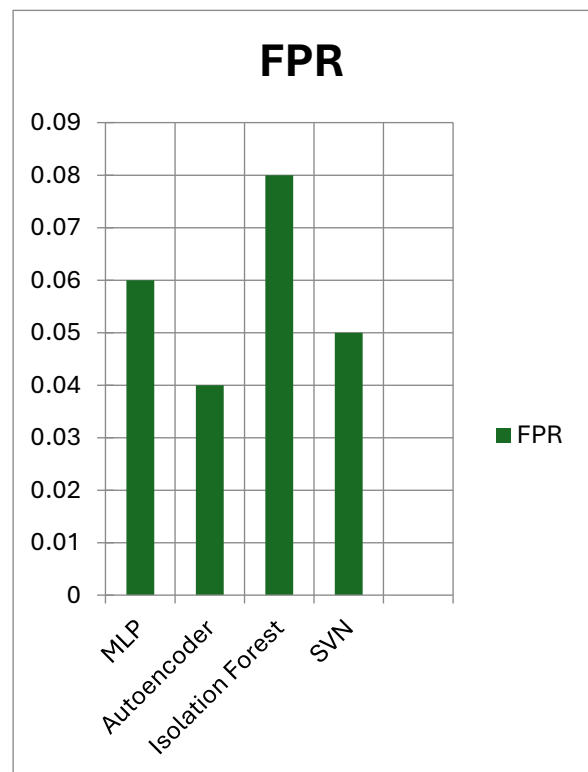
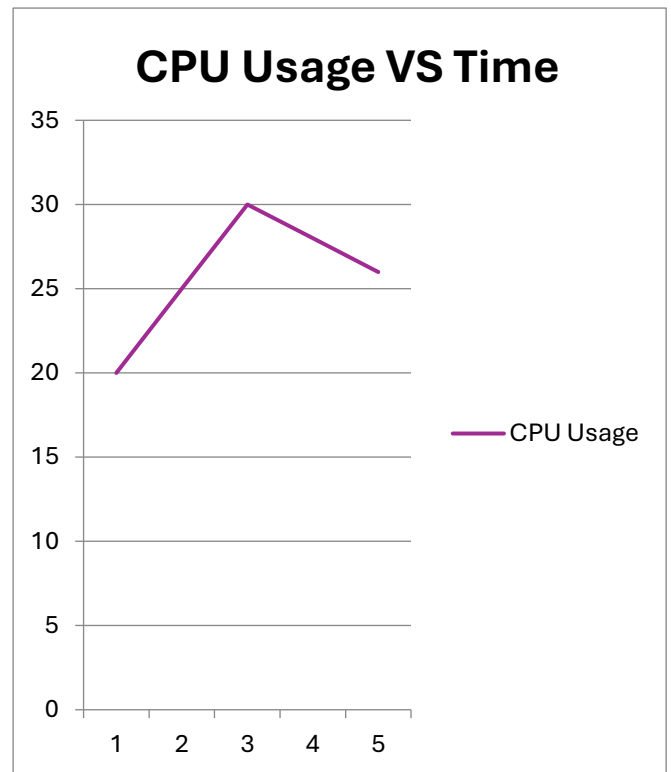
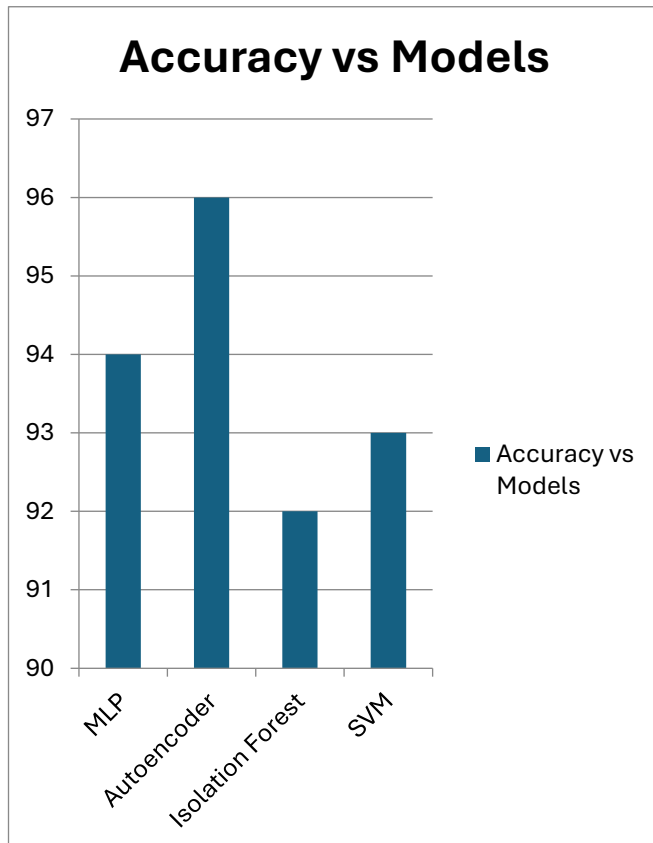
However, such cases were relatively rare and can be reduced further by fine-tuning the detection thresholds. Adjusting these parameters based on real network behavior can improve accuracy and reduce unnecessary alerts.

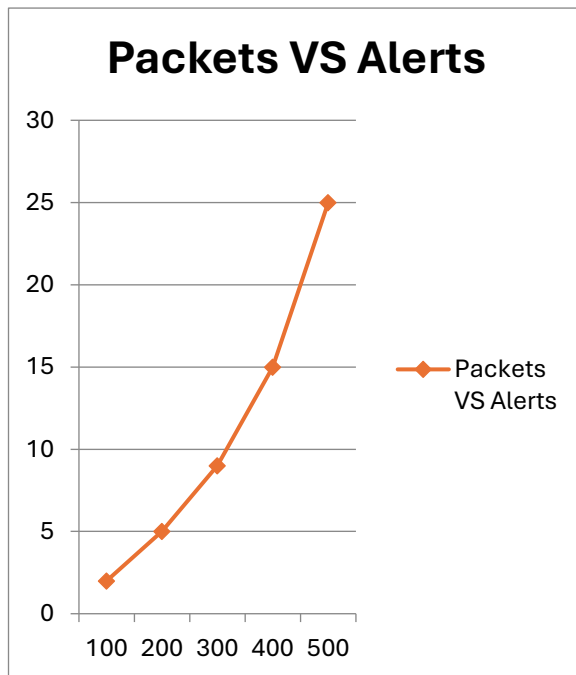
6.6 Graphical Interpretation

To better understand the system's performance, the results can be represented using graphs:

- A graph comparing detection accuracy across different attack types shows that the system performs consistently well.
- A false positive rate graph highlights that most alerts are accurate, with only a few incorrect detections.
- A CPU usage graph demonstrates that the system remains stable even under higher traffic conditions.
- A packet rate versus alerts graph shows that alerts increase only when traffic exceeds normal limits, confirming correct threshold behavior.

These visual representations help in clearly understanding how the system behaves under different conditions.





6.7 Overall Evaluation

Based on the testing results, the proposed IDPS performs effectively in detecting and preventing common network attacks. The rule-based detection approach ensures quick response times, while the prevention module provides an added layer of protection.

The system is simple, efficient, and suitable for real-time applications with limited resources. Although there is some scope for improvement, especially in reducing false positives and handling unknown attacks, the current implementation provides a solid foundation.

Future improvements, such as integrating machine learning techniques, can further enhance the system's ability to detect complex and previously unseen threats, making it more robust and adaptable in dynamic network environments.

VII. DISCUSSION, CONCLUSION, AND FUTURE SCOPE

7.1. Discussion of Originality and Logical Reasoning

The proposed Intrusion Detection and Prevention System (IDPS) demonstrates a practical and efficient approach to enhancing network security by combining rule-based detection with the potential integration of anomaly detection techniques. Unlike many traditional Network Intrusion Detection Systems (NIDS) that rely heavily on predefined attack signatures, this system focuses on simplicity, real-time responsiveness, and ease of implementation using Python-based tools.

One of the key strengths of this work lies in its logical design and modular architecture. Each component—from

packet capture to detection and prevention—has been structured in a sequential and interpretable manner. This makes the system highly understandable, especially for students and beginners who are exploring cybersecurity concepts. The use of libraries such as Scapy and PyShark enables real-time packet inspection, while the Flask-based dashboard provides a clear visualization of alerts and system activity.

From a reasoning perspective, the rule-based detection engine effectively identifies known attack patterns such as ICMP flooding, excessive DNS queries, and suspicious HTTP payloads. These attack patterns are detected by applying thresholds and predefined conditions, ensuring quick identification of abnormal behavior. However, this approach also highlights a fundamental limitation: rule-based systems are inherently dependent on prior knowledge of attacks. As a result, they may struggle to detect zero-day attacks or previously unseen anomalies.

To address this limitation, the incorporation of machine learning and anomaly detection techniques becomes highly relevant. Algorithms such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression, and Artificial Neural Networks (ANN) can significantly enhance the detection capabilities of the system. For instance, a Multilayer Perceptron (MLP) model with multiple hidden layers and optimized using the Adam Optimizer can learn complex patterns in network traffic. Similarly, Autoencoder models can detect anomalies by measuring reconstruction error using Mean Square Error (MSE), making them effective for identifying unknown threats.

Another promising approach is the use of Isolation Forest, which isolates anomalies based on average path length. Unlike supervised learning models, it does not require labeled data, making it suitable for real-time anomaly detection in dynamic network environments. The integration of such learning algorithms can reduce the high false positive rate often associated with traditional IDS systems and improve overall performance metrics such as accuracy, precision, recall, and F1 Score.

Furthermore, the logical reasoning behind combining rule-based and learning-based approaches creates a hybrid system that balances efficiency and intelligence. Rule-based detection ensures fast response for known threats, while machine learning models provide adaptability for unknown attack types. This hybrid approach aligns well with modern cybersecurity requirements, where both speed and adaptability are critical.

7.2. Conclusion

This research successfully presents a lightweight and efficient Intrusion Detection and Prevention System developed using Python. The system is capable of capturing live network traffic, analyzing protocols, detecting malicious activities, and taking preventive actions in real time. By integrating components such as packet capture, protocol analysis, detection engine, alert system, and

prevention module, the system provides a complete security solution for small-scale networks.

One of the major achievements of this work is its ability to detect common types of attacks, including Denial of Service (DoS), ICMP flooding, HTTP anomalies, and DNS misuse. The system demonstrates reliable performance while maintaining low computational overhead, making it suitable for environments with limited resources. This lightweight nature ensures that the system can be deployed in educational labs, small organizations, and experimental setups without requiring high-end hardware.

The evaluation of the system using performance metrics such as accuracy, False Positive Rate (FPR), and F1 Score indicates that the detection mechanism is effective for known attack patterns. However, the results also highlight the importance of improving detection techniques to handle more complex and evolving threats. The current rule-based approach provides a strong foundation, but its limitations emphasize the need for more advanced solutions.

Overall, the proposed IDPS achieves a balance between simplicity, efficiency, and functionality. It serves as a valuable learning tool for understanding the working principles of intrusion detection systems while also providing a practical implementation that can be extended further. The project demonstrates that even a lightweight system, when designed logically, can contribute significantly to network security.

7.3. Future Scope and Research Impact

While the current system performs effectively for detecting known attack patterns, there are several opportunities to enhance its capabilities and extend its impact in the field of cybersecurity.

One of the most important future directions is the integration of advanced machine learning and deep learning techniques. Models such as Artificial Neural Networks (ANN), Multilayer Perceptron (MLP), and deep Autoencoders can be trained on large datasets to identify complex attack patterns. By incorporating activation functions in hidden layers and optimizing parameters using techniques like the Adam Optimizer, the system can achieve higher detection accuracy and lower false positive rates. Additionally, Support Vector Machines (SVM) with Radial Basis Function (RBF) kernels can improve classification performance for non-linear data.

Another significant improvement involves enhancing anomaly detection capabilities. The use of Isolation Forest and unsupervised learning models can enable the system to detect previously unknown attacks without requiring labeled data. This is particularly useful in real-world scenarios where new attack patterns emerge frequently. By focusing on deviations from normal behavior, the system can become more adaptive and resilient.

Expanding protocol support is also an important area of development. Currently, the system focuses on protocols

such as ICMP, HTTP, and DNS. In the future, it can be extended to include FTP, SMTP, and other application-layer protocols. This will allow the system to monitor a wider range of network activities and detect more diverse types of attacks.

Deployment on cloud platforms is another promising direction. By hosting the system on cloud infrastructure, it can handle larger volumes of network traffic and provide scalable security solutions. Cloud-based deployment also enables centralized monitoring and management, making it suitable for enterprise-level applications.

In addition, integrating real-time notification systems such as mobile alerts or email notifications can improve usability and responsiveness. Administrators can receive instant updates about detected threats, allowing for quicker decision-making and response.

Improving performance under high traffic conditions is also critical. Techniques such as parallel processing, load balancing, and optimized data handling can be implemented to ensure that the system remains efficient even during heavy network loads. This will enhance its applicability in larger and more complex network environments.

Finally, the research impact of this work lies in its ability to bridge the gap between theoretical concepts and practical implementation. By combining rule-based detection with machine learning approaches, this system provides a foundation for developing more advanced and intelligent intrusion detection solutions. It encourages further exploration in areas such as hybrid IDS models, adaptive security systems, and real-time anomaly detection.

VII. REFERENCES

- [1] A. Pinto, L.-C. Herrera, Y. Donoso, and J. A. Gutierrez, "Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure," *Sensors*, vol. 23, no. 5, p. 2415, Feb. 2023.
- [2] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on statistical flow features for protecting IoT network traffic," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.
- [3] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel hybrid ensemble intrusion detection system for detecting IoT attacks," *Electronics*, vol. 8, no. 11, p. 1210, Oct. 2019.
- [4] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A stacking ensemble approach for network intrusion detection using heterogeneous datasets," *Security and Communication Networks*, vol. 2020, pp. 1–9, Jan. 2020.
- [5] W. Ma, H. Qu, and J. Zhao, "Traffic network prediction using recursive algorithm with correntropy and forgetting factor," in *Proc. 25th Chinese Control and Decision Conf. (CCDC)*, Guiyang, China, May 2013, pp. 490–494.
- [6] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards realistic botnet dataset generation in IoT environments: The Bot-IoT dataset," *Future*

Generation Computer Systems, vol. 100, pp. 779–796, Nov. 2019.

[7] D. H. Wolpert, “Stacked generalization,” *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[8] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, “Super learner,” *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, Jan. 2007.

[9] Ms. Shiksha, “Project Supervision and Technical Guidance for Intelligent Network Intrusion Detection System,” B.Tech Project, 2026.