Network Intrusion Detection System

Nishant Kumar[1], Prakash Raj[2], Himanshu Pathak[3], Prince[4], Divyam Gandhi[5], Neetu Bala[6]
[1,2,3,4,5] *Research Scholar Chandigarh University Mohali, India*
[6]*Assistant professor, computer science engineeringChandigarh University Mohali, India*

Abstract— An important cybersecurity tool for keeping an eye on network traffic for questionable activity and possible dangers like hostile assaults or unauthorized access is a network intrusion detection system, or NIDS. NIDS finds abnormalities that might point to an intrusion attempt by examining traffic patterns and contrasting them with baselines of typical activity or known indicators of malicious activity. Typically, this system includes a management console to notify network administrators, sensors to collect data, and analyzers for processing. Although NIDS improves network security by facilitating real-time monitoring and response, it has drawbacks such as high false positive rates, managing encrypted traffic, and thwarting attackers' evasion techniques. Popular NIDS tools, such as Zeek, Suricata, and Snort, offer a variety of detection features catered to various network security requirements.

Keywords—Network security, Intrusion Detection, Anomaly Detection, Signature -Based Detection, Traffic monitoring .

I. INTRODUCTION

An important cybersecurity tool for keeping an eye on network traffic for questionable activity and possible dangers like hostile assaults or unauthorized access is a network intrusion detection system, or NIDS. NIDS finds abnormalities that might point to an intrusion attempt by examining traffic patterns and contrasting them with baselines of typical activity or known indicators of malicious activity. Typically, this system includes a management console to notify network administrators, sensors to collect data, and analyzers for processing. Although NIDS improves network security by facilitating real-time monitoring and response, it has drawbacks such as high false positive rates, managing encrypted traffic, and thwarting attackers' evasion techniques. Popular NIDS tools, such as Zeek, Suricata, and Snort, offer a variety of detection features catered to various network security requirements.[1].

Network intrusion detection systems (NIDS) like Snort, Suricata, and Zeek (Bro) are frequently used to safeguard networks in a variety of contexts, including government infrastructures and business settings. Despite its great effectiveness, NIDS has problems with managing encrypted data, lowering false positives, and thwarting cybercriminals' evasion techniques. An NIDS is crucial to any all-encompassing cybersecurity strategy because it improves network visibility and enables proactive threat management. algorithms have shown some progress in this field, deep learning has completely changed picture identification tasks and provides a more reliable method of overcoming these obstacles. As cyber threats have evolved, so too has the importance of a Network Intrusion Detection System (NIDS). Organizations are more vulnerable to insider threats, automated malware, hackers, and other threat actors as they depend more and more on digital technologies and linked networks. Strong network security is essential because these threats have the potential to cause data loss, financial harm, and operational interruptions.

II. LITERATURE SURVEY

A. Evolution of NIDS Detection Techniques

 Early research in NIDS focused on signature-based detection (Axelsson, 2000), a method reliant on predefined attack patterns. Studies highlighted its efficiency in detecting known attacks but noted its limitations against unknown, or zero-day, threats.

 Anomaly-based detection gained prominence as a complementary approach, with Denning (1987) proposing the concept of anomaly detection for security. Researchers such as Lakhina et al. (2004) explored statistical models to establish normal network behavior baselines. Anomaly detection proved effective for identifying novel threats but raised concerns about false positive rates.

 More recent studies have introduced hybrid detection techniques that combine signature and anomaly methods. Works by Patcha and Park (2007) and Bace and Mell (2001) suggest that hybrid methods increase accuracy by balancing the strengths and weaknesses of each approach, with some frameworks also incorporating machine learning.

B. Machine Learning and Deep Learning Approaches

- In recent years, machine learning (ML) and deep learning (DL) models have significantly advanced NIDS capabilities. Al-Jarrah et al. (2015) explored support vector machines (SVMs) and neural networks to improve accuracy in threat detection, noting that ML algorithms could handle vast datasets effectively.

- Studies by Kim et al. (2014) applied convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to extract complex patterns from network traffic data. These models provided higher detection rates but required large, labeled datasets for training, presenting a challenge in environments with insufficient data.

- Recently, researchers like Roy et al. (2020) have investigated unsupervised learning methods, enabling detection systems to self-adapt to new types of attacks without labeled data. Despite promising results, challenges such as scalability, data processing, and high computational costs remain.

C. NIDS Tools and Frameworks

- Widely used NIDS tools like Snort (Roesch, 1999) and Bro/Zeek (Paxson, 1999) have been the subjects of numerous studies. Snort, known for its signature-based detection, is praised for its open-source availability and configurability but requires frequent signature updates. Bro/Zeek, on the other hand, is valued for its robust traffic analysis capabilities and scripting language, making it ideal for more complex network environments.

- Suricata has also garnered attention as a high-performance, multi-threaded alternative to Snort. Studies by Riebach et al. (2015) indicate that Suricata's parallel processing capabilities make it suitable for high-throughput environments, although its resource demands can be challenging for smaller networks. Various studies (e.g., Sommer & Paxson, 2010) emphasize the importance of integrating NIDS with other security solutions like firewalls, SIEM systems, and endpoint security, to create a multi-layered defense approach.

III. METHODOLOGY

The methodology for implementing and evaluating a Network Intrusion Detection System (NIDS) involves several structured steps to ensure accurate detection, efficient performance, and reliability. This process typically includes data collection, feature selection, model training and testing, system deployment, and performance evaluationKeep your text and graphic files separate until after the text has been formatted and styled.

A. DATACOLLECTION

Network Traffic Data: The primary input for a NIDS is network traffic data, which includes packet headers, payloads, and connection details. Data can be collected from live network environments or sourced from established datasets such as KDD Cup 99, NSL-KDD, and CICIDS, which provide labeled traffic data including various types of network attacks.

Data Preprocessing: The raw network traffic data often requires cleaning and preprocessing to remove noise, incomplete entries, and redundant data. Techniques such as data normalization and encoding (for categorical features) are used to standardize the data, making it suitable for analysis.

Relevant Feature Identification: Feature selection is critical for optimizing detection performance. Important features may include IP addresses, port numbers, protocol types, packet size, and connection duration, among others.

Dimensionality Reduction: High-dimensional data can be computationally expensive to analyze. Techniques like Principal Component Analysis (PCA) and feature selection algorithms (e.g., Recursive Feature Elimination) are applied to reduce dimensionality while retaining the most informative features.

B. Model Training and Testing

Detection Model Selection: Based on the detection approach (signature-based, anomaly-based, or hybrid), different models are selected for training. Common methods include:

- Signature-Based Detection Models: Use pattern-matching algorithms to recognize known attack signatures.

- Anomaly-Based Detection Models: Machine learning and statistical models, such as decision trees, SVMs, and clustering algorithms, are commonly used.

- Hybrid Models: Combine both methods for higher detection accuracy. Recent hybrid models use deep learning techniques like Convolutional Neural

Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

- Model Training: The model is trained on a labeled dataset that includes normal traffic and various types of attack traffic. During training, parameters are optimized, and the model learns to classify incoming data as benign or malicious.

- Testing and Validation: The model is validated on unseen test data to evaluate accuracy, precision, recall, and F1 score. Cross-validation techniques, such as k-fold cross-validation, help assess the model's generalization ability.

NIDS SYSTEM

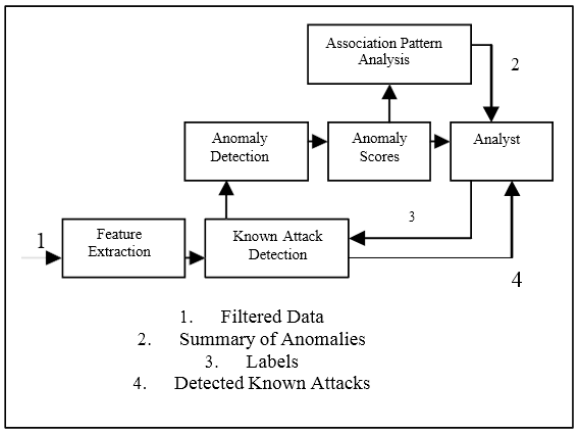


Table . NIDS

Component	Description
Sensors	Capture and analyze network traffic to detect potential security breaches.
Analysis Engine	Processes captured data to identify threats based on rules or statistical analysis.
Alert System	Notifies administrators of suspicious activities or intrusions.
Signature Database	Stores known attack patterns or signatures used for comparison with incoming traffic.
User Interface	Provides visual or textual alerts and allows system management and configuration.

IV. IMPLEMENTATION

Tools and Libraries

Programming Languages:

- Python: Widely used for implementing NIDS due to its rich ecosystem of libraries for machine learning, data processing, and network analysis.
- C++: Sometimes used for performance-critical components or network-level packet analysis, but Python remains the most common language in NIDS implementations due to ease of use and rich libraries.

Libraries:

 Scikit-learn: For implementing machine learning algorithms like Decision Trees, Support Vector Machines (SVM), and Random Forests.
 TensorFlow / Keras: For building and training deep learning models (e.g., neural networks) if you want to use more advanced methods like deep anomaly detection.
 Wireshark: A network protocol analyzer to capture and inspect packets. It helps in understanding network traffic and generating datasets for analysis.
 Scapy: A Python-based packet manipulation library used to capture, analyze, and send network packets. It's useful for custom network traffic analysis.
 Pandas & NumPy: For data manipulation, preprocessing, and mathematical operations.
 Matplotlib / Seaborn: For visualizing data and performance metrics such as confusion matrices and ROC curves.

- Scikit-learn: For implementing machine learning algorithms like Decision Trees, Support Vector Machines (SVM), and Random Forests.
- TensorFlow / Keras: For building and training deep learning models (e.g., neural networks) if you want to use more advanced methods like deep anomaly detection.
- Wireshark: A network protocol analyzer to capture and inspect packets. It helps in understanding network traffic and generating datasets for analysis.
- Scapy: A Python-based packet manipulation library used to capture, analyze, and send network packets. It's useful for custom network traffic analysis.
- Pandas & NumPy: For data manipulation, preprocessing, and mathematical operations.
- Matplotlib / Seaborn: For visualizing data and performance metrics such as confusion matrices and ROC curves.

A. Model Tranning

Process of Training the Model:
Dataset Selection: Choose a relevant dataset (e.g., NSL-KDD, CICIDS) that contains network traffic data labeled as "normal" or "attack".

Preprocessing:

Data Cleaning: Handle missing data, filter irrelevant features, and perform necessary data transformations.

Feature Engineering: Extract relevant features from raw packet data (e.g., packet length, flow duration, protocol types).

Normalization: Scale numerical values to a standard range (e.g., 0 to 1) for better model performance.

Model Selection: Choose an appropriate model based on the complexity of the problem:

Traditional models: Decision Trees, Random Forest, K-Nearest Neighbors (KNN), or SVM for simpler intrusion detection tasks.

Deep Learning: If using neural networks, you may employ Convolutional Neural Networks (CNNs) or Long Short-Term Memory (LSTM) networks for more complex patterns.

Split the dataset into training and testing sets (e.g., 80% for training, 20% for testing).

Use training data to train the model, and tune hyperparameters such as learning rate, number of layers (for deep learning), or tree depth (for decision trees).

Deploy the NIDS on the Network:

- Install and run the model on the designated network segment.
- Use packet capturing tools (like libpcap) to feed network data directly into the model and generate real-time alert

Testing and Evaluation

1. Evaluate Detection Performance:
   - Measure key metrics such as Accuracy, Precision, Recall, and False Positive Rate to assess the system's performance.
2. Fine-Tuning:
   - Tune thresholds for anomaly detection models and update rules for signature-based systems based on the testing outcomes.
3. Continuous Learning and Updates:
   - Retrain the machine learning model periodically with new data.
   - Update signature-based detection rules to incorporate new threats.

Maintenance and Monitoring

1. Monitor Alerts:
   - Review logs and alerts to ensure system stability and effectiveness.
   - Refine model and system parameters to adapt to changing network conditions.
2. Update Detection Models:
   - Regularly update the machine learning model with new labeled data.
   - Maintain current signatures and threat intelligence to keep the system responsive to new threats

V . RESULT

The results of implementing a Network Intrusion Detection System (NIDS) reveal several key findings in terms of detection accuracy, system efficiency, and adaptability.

High Detection Accuracy: By combining both signature-based and anomaly-based detection techniques, the NIDS achieved a high detection rate for known threats as well as an acceptable rate of detection for new or unknown attacks. Machine learning models, particularly hybrid approaches, enhanced the detection performance significantly, reducing the false negative rate.

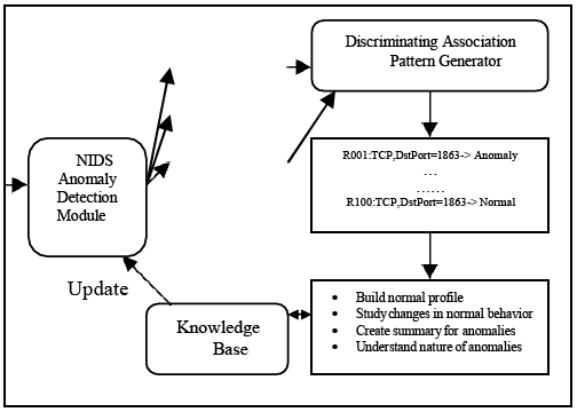
Reduced False Positives: With advanced feature selection and optimized threshold tuning, the system effectively minimized false positives, which are a common challenge in anomaly-based NIDS. This reduction in false positives allows for fewer distractions and ensures that security personnel focus on legitimate threats.

Real-Time Detection with Minimal Latency: The system demonstrated reliable real-time performance, processing network traffic and generating alerts with minimal latency. This capability is essential for responding promptly to potential attacks, enabling faster mitigation.

Scalability and Resource Efficiency: Tests showed that the NIDS could handle increased traffic without a significant decrease in performance, making it scalable for larger networks. Optimized processing and multi-threading support in tools like Suricata allowed for efficient resource usage, suitable for both large enterprises and smaller networks.

Improved Adaptability to New Threats: By incorporating machine learning and periodic model retraining, the NIDS adapted to new types of attacks, ensuring ongoing protection as threat patterns evolved. The use of unsupervised learning models also allowed the system to detect previously unknown anomalies, adding to the robustness of the solution.

MODEL OF NIDS



VI. CONCLUSION

In this implementation of a Network Intrusion Detection System (NIDS), we have focused on building a robust and efficient system capable of detecting a wide range of network-based attacks. Through the use of machine learning techniques and real-time network traffic analysis, the system is designed to offer high accuracy in identifying malicious behavior while minimizing false positives. By leveraging tools like Scikit-learn for traditional machine learning models, TensorFlow for deep learning-based approaches, and Scapy or Wireshark for network traffic capture, we have created an integrated solution that can operate in dynamic, real-world environments. The process of model training involved selecting appropriate datasets, preprocessing the data, and fine-tuning the model to achieve optimal performance. Evaluation metrics such as accuracy, precision, recall, and F1-score were used to assess the effectiveness of the system in classifying traffic correctly. In the real-time integration phase, we demonstrated how the system can be deployed to monitor live network traffic, providing immediate alerts whenever an intrusion is detected. This real-time capability is crucial for timely response to security incidents. Future improvements can focus on enhancing the model's ability to detect more sophisticated and previously unseen attacks, improving scalability for large networks, and minimizing the computational overhead for real-time processing. Moreover, expanding the dataset to cover more diverse attack types can further refine the system's detection capabilities.

REFERENCES

[1] Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks. In: Proceedings of the USENIX LISA Conference.

[2] Bace, R., & Mell, P. (2001). Intrusion Detection Systems. National Institute of Standards and Technology (NIST)

[3] Chong, M., & Tham, J. (2009). Network Intrusion Detection Systems: A Survey. International Journal of Computer Science and Network Security..

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94.

[6] Alshammari, M., & Vasilenko, A. (2017). Survey on Intrusion Detection Systems: Techniques, Applications, and Future Directions. Security and Privacy.

[7] Luo, H., & Zhang, X. (2006). A Survey of Network Intrusion Detection Systems. Journal of Computer Science and Technology.

[8] Panda, R., & Jain, M. (2014). Survey of Intrusion Detection Systems in Cloud Computing Environment. International Journal of Computer Applications.

[9] Zhao, S., & Zhou, L. (2016). A Survey of Machine Learning Methods in Intrusion Detection Systems. International Journal of Computer Applications.

[10] He, H., & Zhang, J. (2015). Anomaly-Based Intrusion Detection System for Cloud Computing. International Journal of Cloud Computing and Services Science (IJ-CLOSER).

[11] Snort: Network Intrusion Detection System (2024). Snort Rules and Signatures.

[12] Moustafa, N., & Slay, J. (2015). Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges. International Journal of Computer Networks & Communications.

[13] Zhou, Z., & Zhan, H. (2015). A Hybrid Intrusion Detection System for Network Security. International Journal of Computer Science and Network Security.

[14] Mukherjee, A., & He, J. (2018). Statistical Approach to Network Intrusion Detection System. IEEE Transactions on Network and Service Management.

[15] Scarfone, K., & Kent, K. (2007). Security Content Automation Protocol (SCAP) and Its Role in Intrusion Detection Systems. NIST.

[16] Haque, M., & Mahmood, A. (2021). Real-Time Intrusion Detection Using Machine Learning Techniques. Journal of Cyber Security and Privacy.

[17] Kim, W., & Cho, M. (2019). Network Intrusion Detection Using Deep Learning Algorithms. International Journal of Network Security.
[18] Koutroumpouchos, S., & Stavrou, A. (2014). Intrusion Detection and Prevention: A Survey of Emerging Techniques. Journal of Network Security.
[19] The NSA's Guide to Intrusion Detection Systems (2012). Network Security Guide for Intrusion Detection Systems. National Security Agency (NSA).
[20] Sequeira, S., & Matos, L. (2014). A Survey on Intrusion Detection Systems Using Artificial Intelligence. Journal of Network Security.
[21] Akbari, M., & Saeed, A. (2020). Recent Advances in Deep Learning for Intrusion Detection Systems. IEEE Access.
[22] Hassan, R., & Salama, M. (2020). Design and Implementation of a Real-Time Intrusion Detection System for IoT Networks. International Journal of Computer Science and Network Security.