IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

AI Integrated Hybrid Intrusion Detection System for Enhanced Security and Threat Detection

Aditya Saxena
Department of Computer Science,
MIT ADT University, Pune

Pratik Sankpal
Department of Computer Science,
MIT ADT University, Pune

Sai Smita Sahoo
Department of Computer Science,
MIT ADT University, Pune

Kavin Pednekar
Department of Computer Science,
MIT ADT University, Pune

Prof. Shweta Yadav
Department of Computer Science,
MIT ADT University, Pune

***Abstract*— One major challenge in the field of security and data mining techniques is intrusion detection. As a result, scientists have studied the search for the fastest and most accurate way to detect intrusions in great detail. In essence, the job of intrusion detection systems is to identify any misuse, illegal activity, or damage done to a system by either internal or external users. Artificial intelligence and machine learning (AI & ML) techniques, each with unique benefits and features, have been applied recently to the design of intrusion detection systems. As a result, the main goal of this article is to increase the intrusion detection process' accuracy through the application of machine learning. In actuality, machine learning makes it simple to recognize trends and patterns, which can then be used to network environments to identify intrusions. It can be quite helpful. To enhance decision-making and intrusion detection, we employ Deep Packet Inspection (DPI) and Anomaly Detection. Please make any necessary changes. Improve the Isolation forest & KNN algorithm's efficiency and the solution's overall performance.**

***Keywords*—Intrusion Detection System, Artificial Intelligence, Machine Learning**

I. INTRODUCTION

The proliferation of malicious files poses a significant threat to digital ecosystems. Traditional signature-based approaches for malware detection have become less effective due to the rapid evolution of malware families and the adoption of evasion tactics. To address this challenge, intelligent malware detection techniques, such as machine learning (ML) and deep learning (DL) models, have emerged as promising solutions. These techniques offer superior performance in analyzing malware instances and detecting new threats with high accuracy. This paper explores the significance of enhancing malware detection accuracy and discusses how ML and DL models contribute to this goal. .Malware detection and classification are critical tasks in cybersecurity. Machine learning (ML) models have emerged as a powerful tool for automating these tasks. However, the effectiveness of ML models depends heavily on the quality of the features extracted from the malware samples.

Feature engineering is the process of transforming raw data into features that are relevant and informative for the ML model. In the context of malware detection, feature engineering involves extracting features that capture the malicious behavior and characteristics of malware samples.

Feature Extraction Methods

Numerous feature extraction methods have been proposed for malware analysis. Some of the most commonly used methods include:

1. Static Analysis

DNS Requests: Analyzing the DNS requests made by the malware sample can reveal its communication patterns and potential targets.

API Calls: Examining the API calls made by the malware sample can provide insights into its functionality and potential malicious actions.

Registry Keys: Modifying registry keys is a common technique used by malware to persist and execute malicious operations. Analyzing the registry keys modified by the malware sample can reveal its persistence mechanisms.

2. Dynamic Analysis

Behavioral Details: Monitoring the behavior of the malware sample in a controlled environment can provide detailed information about its actions, such as file modifications, process creation, and network connections.

Requested Permissions: Analyzing the permissions requested by the malware sample can indicate its potential capabilities and malicious intent.

System Calls: Examining the system calls made by the malware sample can reveal its low-level interactions with the operating system and potential vulnerabilities exploited.

3. Code Structure

Analyzing the code structure of the malware sample can provide insights into its complexity, modularity, and potential obfuscation techniques.

4. Dynamic Behavioral Analysis

Executing the malware sample in a sandbox environment and monitoring its behavior in real-time can provide detailed information about its malicious activities, such as data exfiltration, ransomware attack and botnet communication.

II. CONCEPTS AND METHODS

1. Machine learning (ML) models

Machine learning (ML) models are powerful tools that can be used to solve a wide variety of problems, including classification tasks. Classification tasks involve predicting the class label of a given data point, based on a set of input features. Two commonly used ML models for classification tasks are Isolation forest and k-nearest neighbors (KNN).

- Isolation Forest
  An unsupervised machine learning technique for anomaly identification is the Isolation Forest algorithm. Data points in a dataset are isolated, and anomalies are identified as those that are simpler to isolate. Here is a thorough synopsis:
  The Isolation Principle: Since anomalies are rare and distinct from typical occurrences, they are simpler to identify. The approach splits the dataset iteratively to produce a tree structure called an isolation tree.
  Dividing: A feature and a Isolation split value within the feature's range are chosen to randomly partition the data.
  Length of Path: The path length of a data point is the number of splits needed to isolate it. Since anomalies are isolated rapidly, their path lengths are typically shorter.

- K-Nearest Neighbors
  K-nearest neighbors (KNN) is a simple and intuitive ML model that classifies data points based on the class labels of their nearest neighbors. To make a prediction, KNN first finds the k most similar data points to the new data point, based on a distance metric such as Euclidean distance. The new data point is then assigned the class label that is most common among its k nearest neighbors.

  KNN is a relatively simple and easy-to-understand ML model. It is also computationally efficient, making it suitable for large datasets. However, KNN maybe sensitive to the choice of distance metric and the value of k.

2. Network Traffic Data

Network traffic data refers to the information generated by devices communicating over a network. It includes details such as:

- Source and destination IP addresses:

In network traffic data, the source IP address is the IP address of the device that has sent the packet, while the destination IP address is the IP address of the device that is supposed to receive the packet. IP addresses are 32-bit numbers that are used to identify devices on a network. They are typically written in dotted-decimal notation.

- Port numbers:

Port numbers are 16-bit numbers that are used to identify specific applications or services on a device. When a device sends a packet, it includes the port number of the application or service that is sending the packet. The receiving device uses the port number to determine which application or service should receive the packet.

- Packet size:

The packet size is the length of the packet in bytes. The packet size includes the header information, which contains information about the source and destination IP addresses, the port numbers, and the protocol type. The packet size also includes the payload, which is the data that is being sent.

- Timestamps:

Timestamps are used to record the time that a packet was sent or received. Timestamps are typically stored in the header information of the packet. They can be used to track the progress of a packet as it travels through a network.

- Protocol type (e.g., TCP, UDP):

The protocol type is a field in the header information of a packet that indicates the type of protocol that is being used to send the packet. Common protocol types include TCP, UDP, and ICMP.

3. Feature Extraction

Feature extraction involves identifying relevant attributes from network traffic data that can be used for intrusion detection. Common features include:

- Packet size distribution:

Packet size distribution refers to the frequency of occurrence of packets of different sizes within a given network traffic dataset. It provides insights into the nature of the traffic, such as the presence of large file transfers, streaming media, or control messages. By analyzing the packet size distribution, security analysts can identify potential anomalies or suspicious patterns that may indicate malicious activity or network congestion.

- Packet arrival rate:

Packet arrival rate measures the number of packets received per unit of time. It provides information about the volume and intensity of network traffic. High packet arrival rates can indicate network congestion, denial-of-service attacks, or excessive resource consumption. Conversely, low packet arrival rates may suggest network inactivity or connectivity issues.

- Inter-arrival time:

Inter-arrival time refers to the time interval between the arrivals of consecutive packets. It complements packet arrival

rate by providing insights into the temporal characteristics of network traffic. Consistent inter-arrival times may indicate periodic or predictable traffic patterns, while highly variable inter-arrival times may suggest bursty or erratic traffic.

- Protocol distribution:

Protocol distribution identifies the types of network protocols used in the traffic. Common protocols include TCP, UDP, HTTP, and DNS. Analyzing the protocol distribution helps in understanding the applications and services running on the network. It can also assist in detecting unauthorized or malicious protocols that may indicate security breaches or targeted attacks.

- Port usage patterns:

Packet size distribution, packet arrival rate, inter-arrival time, protocol distribution, and port usage patterns are essential features for extracting meaningful information from network traffic data. These features provide insights into the nature, intensity, and characteristics of network traffic, enabling security analysts to identify anomalies, detect threats, and optimize network performance. By leveraging these features, network traffic analysis becomes a powerful tool for ensuring network security, improving performance, and maintaining the integrity of critical infrastructure.

4. Feature Selection

Feature selection aims to choose the most informative and discriminative features for intrusion detection. Techniques include:

Filter methods:

Filter methods evaluate features independently of any machine learning model. They use statistical measures, such as information gain, chi-square, or correlation, to rank features based on their relevance to the target variable. Features with higher scores are considered more important and are selected for inclusion in the model.

Wrapper methods:

Wrapper methods use a machine learning model to iteratively select features. They start with an initial set of features and evaluate the performance of the model on different subsets of features. The subset that results in the best performance is selected as the optimal feature set.

Embedded methods:

Embedded methods incorporate feature selection into the model training process. They use regularization techniques, such as L1 or L2 regularization, to penalize the coefficients of less important features, effectively shrinking them to zero. Features with non-zero coefficients are considered selected.

5. Data Preprocessing

Network traffic data is a valuable source of information for network administrators and security analysts. However, before this data can be analyzed, it often requires preprocessing to remove noise and outliers, normalize features, handle missing values, and convert categorical features to numerical.

- Remove noise and outliers:

Noise is typically caused by Isolation fluctuations in the data, while outliers are extreme values that do not represent the typical behavior of the network. Both noise and outliers can make it difficult to identify patterns and trends in the data. There are a number of techniques that can be used to remove noise and outliers from network traffic data. One common technique is to use a moving average filter. A moving average filter calculates the average value of a specified number of data points and then replaces each data point with the average value. This helps to smooth out the data and remove noise. Another technique that can be used to remove noise and outliers is to use a median filter. A median filter calculates the median value of a specified number of data points and then replaces each data point with the median value. This helps to remove extreme values from the data.

- Normalize features:

Normalization is a process of scaling the features in a dataset so that they all have the same range of values. This makes it easier to compare the features and to identify patterns and trends in the data. There are a number of different normalization techniques that can be used. One common technique is to use min-max normalization. Min-max normalization scales the features so that they all have a range of values from 0 to 1. Another common technique is to use z-score normalization. Z-score normalization scales the features so that they all have a mean of 0 and a standard deviation of 1.

- Handle missing values:

Missing values can occur in network traffic data for a variety of reasons. For example, a network device may fail to report data for a period of time, or a data collection tool may not be able to capture all of the data. Missing values can make it difficult to analyze the data and can lead to biased results. There are a number of different techniques that can be used to handle missing values. One common technique is to simply ignore the missing values. However, this can lead to biased results if the missing values are not randomly distributed. Another technique that can be used to handle missing values is to impute the missing values. Imputation is the process of estimating the missing values based on the other data in the dataset. There are a number of different imputation methods that can be used, such as mean imputation, median imputation, and k-nearest neighbors imputation.

- Convert categorical features to numerical:

Categorical features are features that can take on a limited number of discrete values. For example, a categorical feature might represent the type of network traffic (e.g., HTTP, FTP, DNS). Categorical features can be difficult to analyze because they cannot be directly compared to numerical features. There are a number of different techniques that can be used to convert categorical features to numerical features. One common technique is to use one-hot encoding. One-hot encoding creates a new binary feature for each category in the categorical feature. For example, a categorical feature with three categories would be converted into three binary features. Another technique that can be used to convert categorical features to numerical features is to use label encoding. Label encoding assigns a unique integer value to each category in the categorical feature.

6. Experimental Design

An experimental design defines the methodology for evaluating intrusion detection models. It includes:

- Data collection strategy
- Feature extraction and selection method

- Model training and evaluation techniques
- Validation strategies

7. Data Collection

Data collection involves gathering network traffic data from various sources, such as:

- Network capture tools:

Network capture tools are software programs that can be used to capture and analyze network traffic. These tools can be used to collect data from a variety of sources, including wired and wireless networks. Network capture tools can be used to identify security threats, troubleshoot network problems, and perform a variety of other tasks. Some of the most popular network capture tools include Wireshark, tcp dump, Snort, Bro.

- Network intrusion detection systems (NIDS):

Network intrusion detection systems (NIDS) are security devices that can be used to detect and prevent intrusions. NIDSs monitor network traffic for suspicious activity and can generate alerts when they detect a potential threat. NIDSs can be used to collect data on a variety of security threats, including Denial of service attacks, Port scans, Malware infections, Phishing attacks.

- Publicly available datasets:

There are a number of publicly available datasets that can be used for network security research. These datasets include data from a variety of sources, including real-world network traffic and simulated attacks. Some of the most popular publicly available datasets include DARPA, KDD Cup, ISCX.

8. Ethical Consideration

Ethical considerations arise when collecting and using network traffic data for intrusion detection. These include:

- Privacy concerns:

Network traffic data contains sensitive information, such as personal browsing habits, financial transactions, and medical records. Collecting and using this data without proper safeguards can violate individuals' privacy rights.

- Data anonymization:

To mitigate privacy concerns, data anonymization techniques can be employed. These techniques remove or alter personally identifiable information from data. However, anonymization can be challenging, as it requires a balance between protecting privacy and preserving data utility for intrusion detection.

- Informed consent:

In some cases, it may be necessary to collect network traffic data from individuals directly. In these situations, obtaining informed consent is crucial.

9. Model Evaluation Metrics

Model evaluation metrics assess the performance of intrusion detection models. Common metrics include:

- Accuracy:

Proportion of correctly or accurately classified instances.

- Precision:

Proportion of true positives to predicted positives.

- Recall:

Proportion of true positives to actual positives.

10. Validation Strategies

Validation strategies ensure that model evaluation results are reliable and generalize well to unseen data. Techniques include:

- Cross-validation:

Dividing the data into multiple subsets for training and testing.

- Holdout validation:

Splitting the data into a training set and a separate test set.

11. Experimental Setup

The experimental setup defines the specific parameters and conditions under which the intrusion detection models are trained and evaluated. This includes:

- Model architecture:

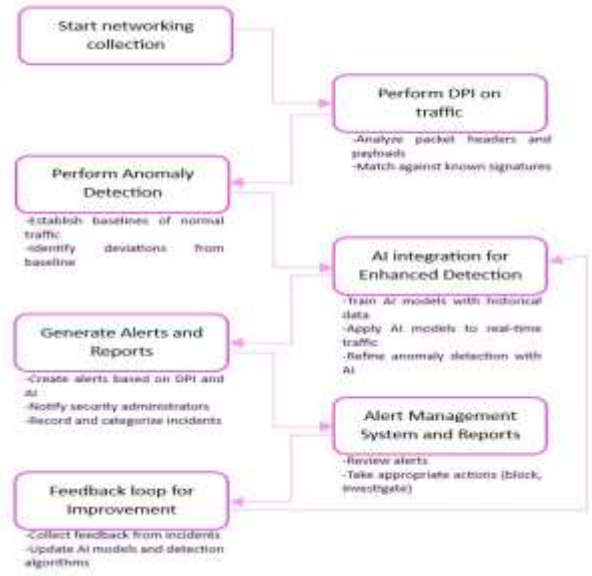
The model architecture defines the structure and connectivity of the intrusion detection model. It determines the type of features used, the number of layers and nodes, and the connections between them.

- Training algorithm:

The training algorithm specifies how the model learns from the training data. It updates the model's parameters to minimize a loss function, which measures the accuracy of the model's predictions.

- Hyperparameter settings:

Hyperparameters are parameters that control the training process but are not learned from the data.



III. METHODOLOGY

To create an efficient intrusion detection system (IDS), this study uses a hybrid methodology that combines machine learning-based anomaly detection with rule-based detection mechanisms. The method entails gathering, preprocessing, feature extraction, training a model, and deploying an AI-enhanced system that can recognize known and unknown

threats in network traffic. Deep packet inspection, anomaly detection models, and structured network logs are all used in the system's design to improve detection accuracy and lower false positives.

Suricata, an open-source intrusion detection engine with deep packet inspection (DPI) and high-performance traffic analysis capabilities, is deployed to start the data collection process. Suricata has been set up to keep an eye on network interfaces and produce comprehensive JSON-based logs that include a variety of information, including protocol details, packet headers, flow records, and alerts brought on by recognized attack signatures. The main source of information used to train and assess the intrusion detection models is these logs.

After being gathered, the raw log data is preprocessed by the ELK stack (Elasticsearch, Logstash, Kibana). The data is parsed, normalized, and transformed into an analysis-ready structured format using Logstash. This entails standardizing field values, eliminating ambiguous or missing information, and cleaning the dataset by eliminating entries that are incomplete or irrelevant. For precise model training and inference, the preprocessing step guarantees data quality and consistency.

The cleaned dataset is subjected to feature engineering in order to extract valuable attributes that can help differentiate between malicious and benign network behavior. To capture the behavioral patterns of network flows, features like average packet size, connection frequency, distribution of protocol usage, alert signature counts, and inter-arrival times are calculated. Principal Component Analysis (PCA) and other dimensionality reduction techniques are used when the feature space becomes high-dimensional in order to remove redundancy and preserve the most informative features.

The study investigates the application of both supervised and unsupervised machine learning methods for the detection stage. Supervised models like Random Forest or Support Vector Machines (SVM) are trained to categorize traffic into benign and malicious groups when labeled datasets are available. Simultaneously, unsupervised anomaly detection models such as Isolation Forest and One-Class SVM are utilized to identify anomalous patterns in situations where labeled data is limited or zero-day attacks are probable. Metrics like accuracy, precision, recall, and F1-score are used to assess the models' performance after they have been trained on reputable public datasets like CICIDS2017 and UNSW-NB15.

Lastly, a real-time detection framework incorporates the trained model. Elasticsearch indexes processed logs, where the model analyzes incoming network events and highlights irregularities. Kibana dashboards are used to visualize the results, allowing security analysts to keep an eye on alerts, look into irregularities, and act quickly. To guarantee the system's robustness and practical applicability in real-world scenarios, it is validated in a controlled testbed environment through simulated attacks. Sensitive network data is handled responsibly thanks to rigorous adherence to ethical principles like data anonymization and privacy.

IV. LITERATURE REVIEW

Traditional network security measures like firewalls and simple Intrusion Detection Systems (IDS) are failing to keep up with the growing complexity and frequency of cyber threats. Tools like Snort and Suricata have long employed signature-based detection methods, like Deep Packet Inspection (DPI), to find known attack patterns in network traffic. DPI uses a pre-established signature database to analyse packet headers and payloads in order to identify threats. DPI-based systems work well for identifying known threats, but they are not very good at spotting unknown or zero-day attacks, and they frequently need to be updated frequently to stay current. Higher false negative rates and an inability to react quickly to emerging threat vectors are the outcomes of this.

Researchers have resorted to anomaly detection driven by machine learning (ML) in order to overcome these constraints. ML provides the capability of learning and recognizing patterns of typical behaviour and identifying deviations as possible dangers. K-Nearest Neighbours (KNN) and Isolation Forest (IF) are two of the machine learning algorithms that have demonstrated encouraging outcomes. Isolation Forest is an unsupervised algorithm that is effective for large-scale outlier detection because it isolates anomalies by recursively partitioning data. KNN, on the other hand, is a supervised learning technique that effectively separates malicious from benign behaviour in network traffic by classifying new instances based on their proximity to known labelled examples.

Hybrid IDS models, which combine the advantages of both strategies, have become a reliable option. To achieve a more complete security posture, these models combine ML-based anomaly detection for unknown threats with DPI for accurate detection of known threats. Studies like those conducted by Moustafa and Slay (2015) have shown that hybrid IDS models that use datasets like UNSW-NB15 or CICIDS2017 perform better in terms of accuracy and false positive rate than single-method systems. A DPI layer for packet inspection, a preprocessing layer for feature extraction, and a machine learning layer for behaviour classification are commonly found in the architecture of a hybrid intrusion detection system. Then, to lower false positives and increase reliability, the alerts produced by DPI and ML are correlated.

Additionally, in order to provide real-time log analysis, visualization, and alert management, hybrid IDS solutions are increasingly integrating tools like Wazuh and the ELK Stack (Elasticsearch, Logstash, Kibana). These systems' modular design makes it possible to deploy them effectively with containerization platforms like Docker, guaranteeing scalability and maintainability in business settings.

Hybrid IDS implementations have drawbacks despite their benefits, including a high computational overhead, a reliance on high-quality datasets, and the requirement for ongoing learning to keep up with emerging threats. Advanced feature selection, federated learning, and risk-based models are some of the strategies being investigated by researchers to make these systems lighter and more effective.

To sum up, the research strongly favours the creation of hybrid intrusion detection systems that make use of machine learning techniques like KNN and Isolation Forest in addition to Deep Packet Inspection. By providing defence against both established and new threats, these systems offer a flexible, scalable, and more precise response to today's cybersecurity issues. In order to guarantee strong and intelligent network defence mechanisms, the capabilities of hybrid IDS models are being improved by the ongoing research in this field.

V. CONCLUSION AND FUTURE WORK

In this study, we proposed several techniques to enhance IDS detection performance. The problem with IDS is the volume of features and data, which leads to irrelevant and outlier characteristics in the dataset. In order to solve that problem, we present a hybrid machine learning mechanism that combines the feature selection process, which stands for supervised learning, with the data reduction process, which is an unsupervised learning technique. To limit the number of

features in the ensemble-based feature selection process, we calculate the median non-zero data from feature importance ranking. Furthermore, in order to configure the boundary of the outlier data, we propose a standard normal distribution/Gaussian distribution algorithm on the LOF score for distinguishing the normal from outlier data.

Evaluations have also been carried out this study to determine how successful the recommended strategy is. The experimental results indicate that reducing the number of features and selecting only the most important ones can improve the system's performance. Furthermore, when key features are separated, the wrapper-based feature selection strategy yields better results. The cut-off value is also set using the Gaussian distribution, and local outlier data can be found using the Local Outlier Factor (LOF) method.

More work on the sensitivity, specificity, and false alarm rate is required to enhance performance. The amount of biased, unbalanced, and outlier data can be increased further to enhance IDS performance.

In the future, imbalanced data in multiple classes can be addressed, the cluster size can be optimized, and techniques to set the cut-off value for identifying the outlier data should be improved. This parameter has an impact on the system's overall performance, as the experiment illustrates.

Last but not least, further research and innovation initiatives will be given top priority in order to investigate cutting-edge technologies and research directions in network security, machine learning, and encryption. This will promote ongoing development and expansion of malware detection capabilities within the network.

REFERENCES

[1] *Wei Zhao, Zhitong Zhao," Providing a hybrid approach to increase the accuracy of intrusion detection systems in computer networks",2024*

[2] *Raisa Abedin Disha, Sajjad Waheed, "Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique",2022*

[3] *Hu, Fan Bai, Xuemiao Yang, Yafe, "IDSDL: a sensitive intrusion detection system based on deep learning Yanjun " 2021*

[4] *Akbar Megantara, Tohari Ahmad "A hybrid machine learning method for increasing the performance of network intrusion detection systems ",2021*

[5] *Sugandh Seth, Gurvinder Singh, Kuljit," A novel time efficient learning-based approach for smart intrusion detection system" ,2021*

[6] *Ansam Khraisat, Iqbal Gondal, Peter Va, "Survey of intrusion detection systems: techniques, datasets and challenges", 2019*

[7] Ilhan Firat Kilincer, Fatih Ertam, Abdulkadir Sengur "Machine Learning methods for cyber security intrusion Detection: Datasets and Comparative study",2021

[8] Mohammed Sayeeduddin Habeeb, T. Ranga Babu, "Network Intrusion Detection system: A survey on artificial intelligence-based techniques", 2022

[9] Patrick Vanin, Thomas Newe, Lubna Luxmi Dhirani, Eoin O'Connell, Donna O'Shea, Brian Lee, Muzaffar Rao "A study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning",2022

[10] Michael MARKEVYCH, Maurice DAWSON, "A review of enhancing Intrusion Detection Systems for cyber security using Artificial Intelligence (AI)",2023

[11] Sowmya T., Mary Anita E.A.," A Comprehensive review of AI based Intrusion Detection system",2023