AI-Driven Network Intrusion Detection Using Weka

Prof. Minal Rahul Chaudhari
Assistant Professor, M.E. Terna Engineering College

***Abstract*— With the rise in sophisticated cyber-attacks, traditional security measures are no longer sufficient. This study employs Artificial Intelligence (AI) to detect anomalies in network traffic. Using the Waikato Environment for Knowledge Analysis (Weka) tool, we compare the performance of three machine learning classification algorithms—J48 (Decision Tree), Random Forest, and Naïve Bayes—to detect intrusions. Our methodology focuses on data preprocessing, feature selection, and classification modeling using a preprocessed KDD Cup dataset to detect malicious traffic, with results indicating that the Random Forest algorithm provides the highest accuracy.**

Index Terms- Cyber Security, Machine Learning, Artificial Intelligence, Classification Model.

I. INTRODUCTION

Cybersecurity threats are evolving rapidly, necessitating the development of automated detection systems. AI, specifically machine learning (ML), enables security tools to learn from historical data and detect new attacks in real-time. Weka is a widely used open-source suite of machine learning algorithms developed in Java, making it ideal for experimental studies in intrusion detection.

II. METHODOLOGY

The methodology consists of five key steps within the Weka Explorer environment, as demonstrated in:

A. Data Collection:

Using the KDD Cup 99 dataset, which contains labeled network traffic data (normal or attack types). There are 41 parameters used in the dataset that are further divided into 4 parts as follows:

1. Basic Features of Individual Connections (Parameters 1–9)

These parameters are directly extracted or derived from the headers of IP packets and TCP/UDP segments. They track foundational session characteristics:

duration: Length of the connection in seconds.

protocol_type: Type of protocol utilized (e.g., tcp, udp, icmp).

service: The destination network service (e.g., http, ftp, smtp, telnet).

flag: Error or normal status of the connection (e.g., SF for normal, S0 or REJ for errors).

src_bytes: Number of data bytes transmitted from source to destination.

dst_bytes: Number of data bytes transmitted from destination to source.

land: Binary indicator (1 if source and destination IP/port are identical, 0 otherwise).

Wrong fragment: Number of bad or corrupted packet fragments.

urgent: Number of urgent packets detected.

2. Content Features within Connections (Parameters 10–22)

hot: Number of "hot" indicator actions (e.g., entering a system directory or running programs).

num_failed_logins: Count of unsuccessful login attempts.

logged_in: Binary state (1 if successfully logged in, 0 otherwise).

num_compromised: Count of compromised conditions encountered.

root_shell: Binary indicator (1 if a root shell or administrative terminal was gained).

su_attempted: Binary indicator (1 if a su root command was attempted).

num_root: Number of operations performed as a root user.

num_file_creations: Number of system files created during the connection.

num_shells: Count of concurrent command shells opened.

num_access_files: Count of operations on access control files.

num_outbound_cmds: Number of outbound commands in an FTP session.

is_hot_login: Binary indicator if the login belongs to a custom hot list.

is_guest_login: Binary indicator (1 if the user logged in using a "guest" account).

3. Time-Based Traffic Features (Parameters 23–31)

Count: Total number of connections pointing to the same destination host within the past 2 seconds.

srv_count: Total number of connections utilizing the same service type within the past 2 seconds.
serror_rate: Percentage of connections that resulted in "SYN" errors among count.
srv_serror_rate: Percentage of connections that resulted in "SYN" errors among srv_count.
error_rate: Percentage of connections that resulted in "REJ" (reject) errors among count.
srv_rerror_rate: Percentage of connections that resulted in "REJ" errors among srv_count.
same_srv_rate: Percentage of connections going to the same service among count.
diff_srv_rate: Percentage of connections going to different services among count.
srv_diff_host_rate: Percentage of connections pointing to different destination hosts among srv_count.
4. Host-Based Traffic Features (Parameters 32–41)
dst_host_count: Number of connections pointing to the same destination host within the 100-connection history window.
dst_host_srv_count: Number of connections pointing to the same service on the same destination host within the window.
dst_host_same_srv_rate: Percentage of identical service connections targeting the host.
dst_host_diff_srv_rate: Percentage of varying service connections targeting the host.
dst_host_same_src_port_rate: Percentage of connections originating from the exact same source port targeting the host.
dst_host_srv_diff_host_rate: Percentage of connections pointing to different destination hosts for the same service.
dst_host_serror_rate: Percentage of connections that triggered SYN errors to the destination host.
dst_host_srv_serror_rate: Percentage of connections that triggered SYN errors to that host's service.
dst_host_rerror_rate: Percentage of connections that triggered REJ errors to the destination host.
dst_host_srv_rerror_rate: Percentage of connections that triggered REJ errors to that host's service

B. Data Preprocessing:
ARFF Conversion: Converting CSV files to Attribute-Relation File Format (.arff), the native format for Weka.
Missing Values: Utilizing Weka's "ReplaceMissingValues" filter to clean data.
Discretization: Converting numeric attributes into nominal ones if necessary for specific classifiers (e.g., Naïve Bayes).

C. Feature Selection:
Reducing dimensionality by selecting relevant features using Ranker and ClassifierSubsetEval in the Select Attributes tab.
Classification Model Training (Classify Tab):
J48: A C4.5 Decision Tree classifier.
Random Forest: An ensemble method that trains multiple decision trees.
Naïve Bayes: A probabilistic classifier based on Bayes' theorem.
Evaluation: Using 10-fold Cross-Validation to ensure the model generalizes well and avoids over-fitting.

III. MATH

To validate a cybersecurity model academically, you must explain the exact mathematical logic behind the performance metrics. Weka calculates these values using a Confusion Matrix, which segments network traffic into four distinct counts: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

A. Classification Accuracy
Accuracy measures the total percentage of network connections that the model classified correctly, whether they were normal or malicious.
text{Accuracy}=frac{TP+TN}{TP+TN+FP+FN}
Cybersecurity Context: While a high accuracy (e.g., 99% looks impressive, it can be mathematically misleading if a dataset is imbalanced. For instance, if 99% of your network traffic is normal and only 1% consists of U2R exploits, a naive model that classifies *everything* as normal will achieve 99% accuracy while failing to detect a single attack.

B. Precision (Positive Predictive Value)
Precision determines the exact ratio of true attacks out of all traffic flagged as anomalous by the system.
text{Precision}=frac{TP}{TP+FP}
Cybersecurity Context: A model with low precision creates a high number of False Positives (FP). In a real-world Security Operations Centre (SOC), low precision translates to thousands of false alarms every day. This creates alert fatigue, causing security analysts to ignore notifications and potentially miss actual breaches.

C. Recall / True Positive Rate (TPR) / Sensitivity
Recall measures the model's capacity to find and capture every single malicious attack present within the network traffic.

text{Recall}=frac{TP}{TP+FN}
Cybersecurity Context: This is the most vital metric for high-risk networks. A low recall score indicates that the system has a high False Negative (FN) rate, allowing hazardous attacks (such as ransomware or data exfiltration) to bypass your perimeter undetected.

D. False Positive Rate (FPR) / Fall-Out
FPR calculates the percentage of completely safe, normal network sessions that were incorrectly blocked or flagged as threats.
text{FPR}=frac{FP}{FP+TN}
Cybersecurity Context: A high FPR directly disrupts business operations. If your machine learning model misclassifies safe database queries or valid employee login attempts as attacks, it will automatically block legitimate traffic, causing self-inflicted downtime.

IV. RESULTS AND ANALYSIS

The models were evaluated based on Accuracy, Precision, Recall, and F-Measure.
J48 Decision Tree: Provided high interpretability and strong performance in classifying normal traffic vs. attacks.
Random Forest: Achieved the highest accuracy (>95%) and lowest False Positive Rate in identifying subtle attacks like DoS, as reported in.
Naïve Bayes: Executed very quickly but showed lower accuracy compared to tree-based models due to the independence assumption of features.

Table 1: Comparative Analysis Results

Classifier	Accuracy	True Positive Rate	False Positive Rate
Random Forest	High	High	Very Low
J48	Medium-High	High	Low
Naïve Bayes	Medium	Medium-High	Medium

V. CONCLUSION

The study demonstrates that AI tools, such as Weka, are effective for cybersecurity analysis without requiring extensive programming. Random Forest was found to be the most reliable for detecting network intrusions. Future work will involve applying these models to live network traffic.

VI. FUTURE WORK

Future work will focus on transitioning from static dataset analysis to live, headless packet classification for immediate, actionable security alerts.

REFERENCES

[1]. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set for network intrusion detection," in Proceedings of the 2009 IEEE International Conference on Computational Intelligence for Cyber Security and Defence (IWDW), 2009, pp. 53–58.
[2]. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining: Practical Machine Learning Tools and Techniques, 4th ed. Morgan Kaufmann, 2016.
[3]. J. Quinlan, C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann Publishers, 1993.
[4]. L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[5]. A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," Cybersecurity, vol. 2, no. 1, pp. 1–22, 2019.
[6]. University of Waikato, "Weka 3: Data Mining Software in Java," [Online]. Available: https://waikato.ac.nz. [Accessed: May 2026].