1.INTRODUCTION

In the rapidly transforming domain of cybersecurity, where attack techniques evolve in sophistication and frequency, the importance of deploying intelligent and resilient network security mechanisms has become paramount. Conventional Intrusion Detection Systems (IDS), which largely rely on rule-based or signature-based detection methods, often face limitations when confronted with zero-day attacks, encrypted traffic, and highly dynamic network behaviors.

These traditional approaches struggle to generalize beyond known attack patterns, resulting in delayed detection and increased false alarm rates. To address these shortcomings, the AI-Based Network Intrusion Detection System using Machine Learning and Deep Learning emerges as a forward-looking solution that leverages the adaptive and predictive capabilities of Artificial Intelligence.

By integrating Machine Learning algorithms capable of identifying hidden patterns in large-scale network traffic and Deep Learning models adept at capturing complex, non-linear relationships among features, the proposed system significantly enhances the accuracy, scalability, and robustness of intrusion detection. The project strategically utilizes two benchmark datasets-NSL-KDD and UNSW-NB15-each contributing unique characteristics to the learning process. NSL-KDD provides a refined and balanced dataset that addresses redundancy issues present in earlier datasets, enabling effective evaluation of traditional and modern ML techniques, while UNSW-NB15 introduces contemporary attack scenarios and realistic network traffic behaviors that reflect current cyber threat landscapes.

By training and validating the IDS across these diverse datasets, the system gains the ability to learn from historical attack signatures while simultaneously adapting to novel and emerging intrusion patterns. This dual-dataset approach not only improves generalization across different network environments but also strengthens the system’s resilience against evolving cyber threats. Ultimately, the project aims to deliver a future-ready, intelligent intrusion detection framework that can proactively safeguard network infrastructures,

minimize security breaches, and support real-time decision-making in modern cybersecurity operations.

A key strength of this project lies in its use of two comprehensive and widely recognized datasets—**NSL-KDD** and **UNSW-NB15**—which together provide a holistic representation of network traffic and attack scenarios. The NSL-KDD dataset serves as a foundational benchmark that addresses redundancy and imbalance issues found in earlier datasets, allowing for more reliable training and evaluation of intrusion detection models. It facilitates the classification of various attack categories and supports comparative analysis of different ML and DL algorithms. In contrast, the UNSW-NB15 dataset introduces modern and realistic network traffic patterns, including contemporary attack types that reflect real-world cyber threats faced by current networks.

By training the IDS on both datasets, the system benefits from exposure to both traditional and modern intrusion techniques, thereby enhancing its generalization capability and reducing overfitting to a single traffic profile. This dual-dataset strategy ensures that the proposed IDS does not merely rely on historical signatures but evolves into an adaptive defense mechanism capable of identifying previously unseen attacks. Ultimately, the project aims to develop a future-ready, intelligent intrusion detection framework that strengthens network resilience, supports proactive threat mitigation, and significantly improves the reliability and effectiveness of cybersecurity defenses in an ever-evolving threat landscape.

2. ENVIRONMENT SETUP & PREREQUISITES

To ensure successful deployment, the host machine must meet the following software requirements:

2.1.CORE SOFTWARE

1. **Python Interpreter (v3.8 or higher)**
    - **Source:** Official Python Website
    - **Critical Configuration:** During installation, ensure the checkbox **"Add Python to PATH"** is selected. This allows Python commands to be executed from the terminal.
2. **Integrated Development Environment (IDE)**
    - **Recommended:** Visual Studio Code (VS Code)

Alternative: PyCharm or Jupyter Notebook

2.2.SYSTEM TOOLS

- **Command Line Interface:** Windows Command Prompt (cmd), PowerShell, or macOS/Linux Terminal.

3.INSTALLATION AND CONFIGURATION

3.1 Workspace Initialization

1. Navigate to your Desktop or preferred directory.
2. Create a new directory named **AI_NIDS_Project**.
3. Launch **VS Code** and select **File > Open Folder**, choosing the directory created in the previous step.

3.2 Dependency Management

The project relies on specific Python libraries for data processing, machine learning, and visualization.

1. Open the integrated terminal in VS Code (Ctrl + `).
2. Execute the following command to install all dependencies:
   pip install pandas numpy scikit-learn streamlit seaborn matplotlib

Library Overview:

- pandas & numpy: Data manipulation and numerical operations.
- scikit-learn: Implementation of the Random Forest algorithm.
- streamlit: Framework for the web-based dashboard UI.
- seaborn & matplotlib: Data visualization and plotting.

4.SOURCE CODE

```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from groq import Groq
import os

# --- PAGE SETUP ---
st.set_page_config(page_title="AI-NIDS Student Project", layout="wide")

st.title("AI-Based Network Intrusion Detection System")
st.markdown("""
**Student Project**: This system uses **Random Forest** to detect Network attacks and **Groq AI** to explain the packets.
""")

# --- CONFIGURATION ---
DATA_FILE = "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv"

# --- SIDEBAR: SETTINGS ---
st.sidebar.header("1. Settings")
groq_api_key = st.sidebar.text_input("Groq API Key (starts with gsk_)", type="password")
st.sidebar.caption("[Get a free key here](https://console.groq.com/keys)")

st.sidebar.header("2. Model Training")

@st.cache_data
def load_data(filepath):
    try:
```

```
        df = pd.read_csv(filepath, nrows=15000)
        df.columns = df.columns.str.strip()
        df.replace([np.inf, -np.inf], np.nan, inplace=True)
        df.dropna(inplace=True)
        return df
    except FileNotFoundError:
        return None

def train_model(df):
    features = ['Flow Duration', 'Total Fwd Packets', 'Total Backward Packets',
                'Total Length of Fwd Packets', 'Fwd Packet Length Max',
                'Flow IAT Mean', 'Flow IAT Std', 'Flow Packets/s']
    target = 'Label'

    missing_cols = [c for c in features if c not in df.columns]
    if missing_cols:
        st.error(f"Missing columns in CSV: {missing_cols}")
        return None, 0, [], None, None

    X = df[features]
    y = df[target]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    clf = RandomForestClassifier(n_estimators=10, max_depth=10, random_state=42)
    clf.fit(X_train, y_train)

    score = accuracy_score(y_test, clf.predict(X_test))
    return clf, score, features, X_test, y_test

# --- APP LOGIC ---
df = load_data(DATA_FILE)
```

```
if df is None:
    st.error(f"Error: File '{DATA_FILE}' not found. Please upload it to the Files tab.")
    st.stop()

st.sidebar.success(f"Dataset Loaded: {len(df)} rows")

if st.sidebar.button("Train Model Now"):
    with st.spinner("Training model..."):
        clf, accuracy, feature_names, X_test, y_test = train_model(df)
        if clf:
            st.session_state['model'] = clf
            st.session_state['features'] = feature_names
            st.session_state['X_test'] = X_test
            st.session_state['y_test'] = y_test
            st.sidebar.success(f"Training Complete! Accuracy: {accuracy:.2%}")

st.header("3. Threat Analysis Dashboard")

if 'model' in st.session_state:
    col1, col2 = st.columns(2)

    with col1:
        st.subheader("Simulation")
        st.info("Pick a random packet from the test data to simulate live traffic.")

        if st.button(" Capture Random Packet"):
            random_idx = np.random.randint(0, len(st.session_state['X_test']))
            packet_data = st.session_state['X_test'].iloc[random_idx]
            actual_label = st.session_state['y_test'].iloc[random_idx]

            st.session_state['current_packet'] = packet_data
```

```
        st.session_state['actual_label'] = actual_label

if 'current_packet' in st.session_state:
    packet = st.session_state['current_packet']

    with col1:
        st.write("**Packet Header Info:**")
        st.dataframe(packet, use_container_width=True)

    with col2:
        st.subheader("AI Detection Result")
        prediction = st.session_state['model'].predict([packet])[0]

        if prediction == "BENIGN":
            st.success(f" STATUS: **SAFE (BENIGN)**")
        else:
            st.error(f" 🚨 STATUS: **ATTACK DETECTED ({prediction})**")

        st.caption(f"Ground Truth Label: {st.session_state['actual_label']}")

        st.markdown("---")
        st.subheader(" Ask AI Analyst (Groq)")

        if st.button("Generate Explanation"):
            if not groq_api_key:
                st.warning(" Please enter your Groq API Key in the sidebar first.")
            else:
                try:
                    client = Groq(api_key=groq_api_key)

                    prompt = f"""
                    You are a cybersecurity analyst.
```

```
                A network packet was detected as: {prediction}.

                Packet Technical Details:
                {packet.to_string()}

                Please explain:
                1. Why these specific values (like Flow Duration or Packet Length) might indicate {prediction}.
                2. If it is BENIGN, explain why it looks normal.
                3. Keep the answer short and simple for a student.
                """

                with st.spinner("Groq is analyzing the packet..."):
                    completion = client.chat.completions.create(
                        model="llama-3.3-70b-versatile",  # <--- UPDATED MODEL NAME
                        messages=[
                            {"role": "user", "content": prompt}
                        ],
                        temperature=0.6,
                    )
                    st.info(completion.choices[0].message.content)

            except Exception as e:
                st.error(f"API Error: {e}")
else:
    st.info(" Waiting for model training. Click **'Train Model Now'** in the sidebar.")
```

5.DATA MANAGEMENT STRATEGY

A well-defined data management strategy is a critical component of an AI-based intrusion detection network system, as the performance and reliability of Machine Learning and Deep Learning models are highly dependent on the quality, consistency, and handling of data. The strategy begins with data collection, where network traffic is gathered from reliable and diverse sources such as benchmark datasets (NSL-KDD and UNSW-NB15) as well as real-time network logs, packet captures, and flow records. This ensures that the system is exposed to a wide range of normal and malicious behaviors. Once collected, data preprocessing plays a vital role in improving data quality by removing redundant records, handling missing or noisy values, normalizing numerical features, and encoding categorical attributes into machine-readable formats. Effective preprocessing reduces bias, minimizes overfitting, and enhances the learning capability of AI models.

Following preprocessing, data labeling and categorization are carefully managed to ensure accurate classification of traffic into normal and attack classes, which is essential for supervised learning approaches. The strategy also includes data partitioning, where datasets are systematically divided into training, validation, and testing sets to enable unbiased performance evaluation and model tuning. To address the dynamic nature of cyber threats, the system incorporates data versioning and continuous updating, allowing new traffic patterns and emerging attack signatures to be periodically integrated into the dataset. This supports model retraining and adaptation over time, ensuring long-term effectiveness. Additionally, secure data storage and access control mechanisms are implemented to protect sensitive network data from unauthorized access or tampering, maintaining data integrity and confidentiality. By combining structured data handling, continuous updates, and strong security practices, the data management strategy ensures that the AI-based intrusion detection system remains accurate, scalable, and resilient in the face of evolving network environments and cyber threats.

5.1 Simulation Mode (Default)

The application includes a built-in data simulator using numpy. This allows the system to function immediately for demonstration purposes without external dependencies. No action is required to enable this mode.

5.2 Production Mode (Real-World Data)

For advanced demonstrations using the **CIC-IDS2017** benchmark dataset:

1. **Source:** Download the CSV files from the UNB CIC Dataset Repository.
2. **Integration:** Move the downloaded .csv file into the AI_NIDS_Project root directory.

Code Adaptation: In nids_main.py, locate the load_data() function and replace the simulation logic with:

```
# Example for real data loading
df = pd.read_csv('Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv')
```

6.EXECUTION & OPERATION

To launch the NIDS Dashboard, perform the following steps:

1. Ensure your terminal path is set to the project directory: AI_NIDS_Project.
2. Execute the startup command:
   streamlit run nids_main.py
3. **Access:** The system will initialize a local web server and automatically open the default web browser to http://localhost:8501.

Operational Workflow:

1. **Dashboard Overview:** Review the project description.
2. **Model Training:** Click the **"Train Model Now"** button in the sidebar to initialize the Random Forest Classifier.
3. **Live Simulation:** Use the "Live Traffic Simulator" section to input packet parameters and test the model's detection capabilities.

7.REFERENCES & TECHNICAL RESOURCES

- **Language Documentation:** Python.org Docs
- **Editor:** Visual Studio Code
- **Dataset:** CIC-IDS2017 - Canadian Institute for Cybersecurity
- **UI Framework:** Streamlit Documentation
- **ML Algorithm:** Scikit-Learn Random Forest

8.TROUBLESHOOTING

- **'streamlit' is not recognized:** This usually means Python was not added to your PATH. Reinstall Python and check the "Add to PATH" box, or try running python -m streamlit run nids_main.py.
- **ModuleNotFoundError:** Ensure you ran the pip install command in Section 3.2.
- **Browser doesn't open:** Manually open your browser and type http://localhost:8501.

9.CONCLUSION

This implementation guide provides the necessary steps to build a functional AI-driven security tool. By following these instructions, you will have a working prototype capable of training a model and detecting simulated network anomalies, suitable for academic demonstration and further research. In conclusion, the AI-based intrusion detection network system represents a significant advancement in the field of cybersecurity by addressing the inherent limitations of traditional intrusion detection approaches. Through the integration of Machine Learning and Deep Learning techniques, the system demonstrates an enhanced ability to analyze large-scale network traffic, learn complex behavioral patterns, and accurately distinguish between normal and malicious activities. The use of diverse and well-established datasets such as NSL-KDD and UNSW-NB15 enables the system to gain both historical insight and contemporary threat awareness, resulting in improved detection accuracy, reduced false alarms, and better adaptability to evolving cyber-attack strategies. By moving beyond static, rule-based detection methods, the proposed AI-driven IDS offers a dynamic and intelligent security solution capable of identifying known as well as previously unseen intrusions. Ultimately, this approach contributes to the development of a more resilient, scalable, and future-ready network security framework, reinforcing the role of artificial intelligence as a critical component in safeguarding modern digital infrastructures against increasingly sophisticated cyber threats.