REAL-TIME NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING

Submitted By

Hafiz Farhad (22K4787)

Usman Ghani (22K4756)

Muhammad Ali (22K4691)

Department of Cyber Security
National University of Computer & Emerging Sciences

1. Motivation

The motivation for this project stems from the increasing sophistication of cyber threats and attacks targeting network infrastructure across organizations of all sizes. Traditional signature-based intrusion detection systems often fail to detect new, previously unseen attacks (zero-day exploits). By leveraging machine learning techniques, we aim to develop a real-time network intrusion detection system capable of identifying anomalous network traffic patterns that may indicate potential security breaches, without relying solely on predefined signatures. This project represents our efforts to contribute to the field of network security by implementing an intelligent, adaptive system that can help organizations better protect their critical assets and information.

2. Overview

2.1 Significance of the Project

Network security remains one of the most critical challenges in the cybersecurity landscape. As threat actors continuously develop new attack vectors and techniques, traditional security measures frequently prove inadequate. Our NIDS project is significant for several reasons:

1. **Real-time Detection**: Unlike traditional log analysis, our system monitors network traffic in real-time, enabling immediate detection and response to potential threats.

2. **Machine Learning Approach**: By employing neural networks, our system can detect anomalous patterns that signature-based systems might miss, especially zero-day attacks.

3. **Adaptability**: The system can be retrained with new data, allowing it to adapt to evolving threat landscapes.

4. **Practical Implementation**: Our solution works with standard network interfaces and requires minimal setup, making it accessible for organizations with limited security resources.

The academic value of this project lies in the practical application of machine learning to cybersecurity, demonstrating how neural networks can be effectively deployed in operational security settings.

2.2 Description of the Project

Our Network Intrusion Detection System (NIDS) is a Python-based application that captures and analyzes network packets in real-time to identify potential security threats. The system employs a neural network model trained on network traffic data to classify packets as either normal or anomalous.

The project involves:

1. **Packet Capture**: Using the Scapy library to capture network packets from specified interfaces.

2. **Feature Extraction**: Extracting relevant features from each packet, including protocol type, service, flags, and various traffic metrics.

3. **Machine Learning Classification**: Using a PyTorch neural network model to evaluate packet features and identify anomalies.

4. **Alert System**: Generating detailed alerts with specifics about detected threats, including source, destination, protocol information, and attack type classification.

5. **Logging System**: Maintaining comprehensive logs for later forensic analysis and system performance evaluation.

The scope of our work focuses on detecting common network attacks such as port scans, DoS attempts, and anomalous connection patterns. The system is designed to be deployed on network edge devices or security monitoring stations that have visibility into network traffic.

2.3 Background of the Project

Network Intrusion Detection Systems have evolved significantly since their inception in the 1980s. Early systems like Haystack and IDES relied primarily on statistical anomaly detection, while later systems incorporated signature-based detection methods. Recent advancements in machine learning have opened new possibilities for more effective intrusion detection.

The implementation uses modern Python libraries including:

- Scapy for packet manipulation
- PyTorch for the neural network implementation
- Pandas and NumPy for data processing
- Joblib for model persistence

2.4 Project Category

This project falls under the **Product-based** category, as we are developing a functional software tool that can be deployed in real-world environments. While incorporating research elements, particularly in the machine learning approach to traffic classification, the primary goal is to create a practical, deployable network security solution.

3. Features / Scope / Modules

Our Network Intrusion Detection System includes the following key features:

1. **Real-time Traffic Monitoring and Analysis**
    - Continuous monitoring of network interfaces
    - Packet capture and inspection without affecting network performance
    - Buffered processing to handle high traffic volumes efficiently

2. **Machine Learning-based Anomaly Detection**
    - Neural network model trained on network traffic data
    - Feature extraction optimized for network security analysis
    - Configurable anomaly threshold for fine-tuning detection sensitivity

3. **Specific Attack Pattern Recognition**
    - Detection of port scanning activities
    - Identification of stealth scanning techniques (NULL, FIN, XMAS)
    - Recognition of potential DoS/DDoS attack patterns
    - ICMP-based attack detection

4. **Comprehensive Logging and Alerting**
    - Detailed logs of detected anomalies
    - Classification of attack types with severity ratings
    - Timestamped entries for forensic analysis
    - Console and file-based alert mechanism

5. **Flexible Deployment Options**
    - Support for multiple network interfaces
    - Configurable capture duration
    - Trusted host exclusion capability
    - Minimal dependencies for easier deployment

6. **Dynamic Connection Tracking**
    - Monitoring connection patterns over time
    - Statistical analysis of connection frequencies
    - Automatic cleanup of stale connection data
    - Enhanced context for anomaly detection

7. **Performance Monitoring**
    - Real-time statistics on packets processed
    - Attack detection rates and percentages
    - System resource utilization tracking
    - Periodic status updates

8. **User-friendly Interface**
    - Interactive setup process
    - Clear presentation of available network interfaces
    - Readable alert formatting with color coding
    - Configurable system parameters

4. Project Planning

The project was executed according to the following schedule, with responsibilities divided between team members:

Responsibility Breakdown:

- **Usman Ghani**: Focused on model architecture, implementation, packet capture mechanism, and connection tracking logic.
- **Hafiz Farhad**: Led data collection, feature extraction development, alert system design, and testing methodology.

- **Muhammad Ali**: Requirements analysis, integration testing, documentation, and final system optimization.

5. Project Feasibility

Technical Feasibility

Our NIDS project is technically feasible for the following reasons:

- **Established Technologies**: The project utilizes well-established libraries (Scapy, PyTorch, Pandas) with proven track records in network analysis and machine learning.
- **Hardware Compatibility**: The system runs on standard computing hardware without specialized components.
- **Scalability**: The buffered processing approach allows the system to handle varying traffic loads.
- **Technical Risks**:
    - False positives/negatives in anomaly detection (mitigated through threshold tuning)
    - Performance bottlenecks in high-volume networks (addressed through efficient buffering)
    - Dependency on specific Python libraries (managed through explicit version control)

Economic Feasibility

The project demonstrates strong economic feasibility:

- **Development Costs**: Minimal, as all libraries and tools used are open source.
- **Operational Costs**: Low computing resource requirements mean the system can run on existing infrastructure.

Schedule Feasibility

The project timeline was realistic and achievable:

- **Time Constraints**: The 12-week development cycle provided adequate time for all phases.
- **Milestone Planning**: Clear milestones allowed for progress tracking and adjustment.
- **Resource Allocation**: The division of responsibilities ensured efficient use of team resources.
- **Buffer Time**: We incorporated buffer periods for unexpected challenges, particularly during the integration phase.

6. Hardware and Software Requirements

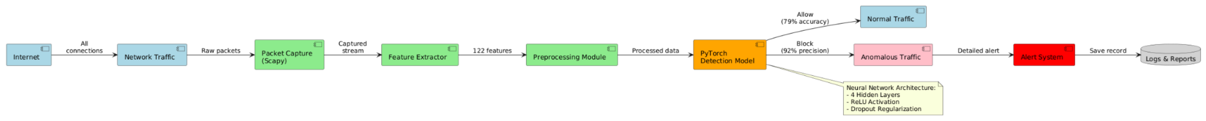
Hardware Requirements

- **Computing Platform**:
  - Processor: Dual-core CPU (2.0 GHz or higher)
  - Memory: 4GB RAM minimum, 8GB recommended
  - Storage: 1GB free disk space for the application and logs
  - Network Interface Card: Compatible with promiscuous mode operation
- **Network Environment**:
  - Access to network traffic (via mirrored port, network tap, or in-line deployment)
  - Sufficient bandwidth to handle traffic monitoring without degradation

Software Requirements

- **Operating System**:
  - Linux (Ubuntu 20.04 or newer recommended)
  - Windows 10/11 with appropriate packet capture drivers
  - macOS 11.0 or newer
- **Python Environment**:
  - Python 3.8 or newer
  - Package dependencies:
    - scapy==2.5.0
    - pandas==2.0.0
    - numpy==1.24.0
    - scikit-learn==1.2.2
    - torch==2.0.1
    - joblib==1.2.0
    - netifaces==0.11.0
- **Additional Software**:
  - Wireshark (optional, for verification and analysis)
  - Administrative/root access (required for packet capture)

7. Diagrammatic Representation of the Overall System



The diagram above illustrates the overall architecture of our Network Intrusion Detection System. The system consists of the following key components:

1. **Packet Capture Module**: Interfaces with network hardware to collect raw packet data
2. **Feature Extraction Engine**: Processes packets to derive relevant features
3. **Machine Learning Classifier**: Neural network that evaluates packets for anomalies
4. **Alert & Logging System**: Manages detection results and generates appropriate notifications
5. **User Interface**: Provides configuration options and displays system status

The data flow follows a pipeline architecture, with packets moving through each processing stage sequentially, allowing for efficient processing and minimal latency.

8. References

[1] **D. E. Denning,** "An Intrusion-Detection Model," IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222-232, 1987.

[2] **M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani,** *"A detailed analysis of the KDD CUP 99 data set,"* **Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA),** Ottawa, ON, Canada, 2009, pp. 1–6.

[3] **A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala,** *"PyTorch: An Imperative Style, High-Performance Deep Learning Library,"* **Advances in Neural Information Processing Systems 32 (NeurIPS),** pp. 8024–8035, 2019.

[4] **P. Biondi,** *"**Scapy**: A Powerful Interactive Packet Manipulation Tool,"* **Available at:** https://scapy.net