

AI-Driven Cybersecurity Model for Real-Time Threat Detection and Prevention

Benitlin Subha K¹, Prince Immanuel J², Pozilan R³, Sam Jacob T⁴ and Srinath R⁵

¹Assistant Professor -Department of Information Technology & Kings Engineering College-India.

^{2,3,4,5}Department of Information Technology & Kings Engineering College-India

Abstract - As cyber threats become more dynamic and sophisticated, traditional intrusion detection systems often fall short in identifying emerging attacks in real-time. This project introduces a hybrid AI-driven cybersecurity model that combines supervised machine learning and deep reinforcement learning for adaptive, explainable, and real-time threat detection and prevention. It uses an XGBoost classifier trained on the UNSW-NB15 dataset to predict initial threats and a Deep Q-Network (DQN) agent, built with PyTorch, to make optimal security decisions based on evolving threats. The DQN interacts with real or simulated network traffic, continuously improving its policy through reward feedback. Key network features such as protocol type, port numbers, packet size, and inter-arrival time are standardized for consistent analysis. A rules-based engine supports data labeling when public datasets are insufficient. A Flask-powered dashboard provides live threat monitoring, SHAP-based model explanations, and performance insights. This modular system achieves high detection accuracy and adapts to new threats without frequent retraining. By integrating classic ML with reinforcement learning, the solution offers a future-ready cybersecurity framework that operates intelligently, efficiently, and transparently in real-time environments.

Key Words: Cybersecurity, Intrusion Detection System (IDS), Real-time Threat Detection, Machine Learning, Supervised Learning, XGBoost, Deep Reinforcement Learning, Deep Q-Network (DQN)

1. INTRODUCTION

The rise of cyber threats in today's interconnected digital world has significantly increased the demand for intelligent and adaptive cybersecurity systems. Traditional Intrusion Detection Systems (IDS) often rely on static rules and signature-based mechanisms to identify threats, which limits their ability to handle evolving and sophisticated attack techniques. Modern attackers are increasingly leveraging automated scripts, zero-day vulnerabilities, and complex multi-stage strategies that evade detection by conventional systems. In this context, artificial intelligence (AI) provides a promising alternative to static IDS models by offering dynamic learning, pattern recognition, and adaptive decision-making capabilities.

This project presents an AI-driven cybersecurity framework designed for real-time threat detection and autonomous response. It combines two powerful branches of AI—supervised machine learning and reinforcement learning—to deliver a model capable of not only identifying threats accurately but also learning from interactions with the environment. The system begins by training an XGBoost

classifier on a comprehensive public dataset, UNSW-NB15, which includes various types of benign and malicious traffic. This supervised model serves as a baseline to classify incoming packets into attack or benign categories.

To go beyond static detection, a Deep Q-Network (DQN) agent is implemented to learn and optimize actions such as 'allow' or 'block' based on prediction confidence and reward feedback. The reinforcement learning agent is designed to operate in environments with both real-time and simulated traffic. For demonstration and testing, live network traffic is simulated using the UNSW test set or Scapy-generated packets, ensuring consistent feature extraction and real-time feedback loops. The reward function is carefully designed to encourage correct decisions and penalize false positives or false negatives, thereby helping the agent to improve over time.

Another important aspect of this project is transparency. The system incorporates Explainable AI (XAI) features using SHAP (SHapley Additive exPlanations) to provide packet-level insight into why certain packets are classified as malicious or benign. This not only aids human analysts in validating the system's decisions but also makes the system more trustworthy and auditable.

Furthermore, a real-time web dashboard is developed using Flask to provide a visual interface for monitoring live traffic, detection results, and DQN actions. It displays cumulative attacks detected, SHAP explanations, and real-time logs, making it user-friendly and insightful for both technical and non-technical users.

The proposed AI-driven cybersecurity model is modular, extensible, and ready for integration into larger network security systems. It demonstrates that blending traditional ML classification with reinforcement learning agents and explainable AI techniques can produce intelligent, adaptive, and reliable intrusion detection and prevention systems. This introduction establishes the context and motivation for the project and outlines the key components and innovations that will be explored in detail throughout the subsequent chapters.

1.1 Problem Statement

The modern digital landscape is characterized by unprecedented levels of interconnectivity, exposing computer networks to a wide range of cyber threats, including malware, distributed denial of service (DDoS) attacks, phishing, protocol-based attacks, and port scanning. Traditional Intrusion Detection Systems (IDS), which typically rely on rule-based or signature-based detection methods, are increasingly ineffective against evolving attack patterns and sophisticated adversaries that exploit system vulnerabilities in dynamic ways. These conventional systems are limited in their ability to adapt to new types of attacks, and they often require manual updates and expert intervention to maintain efficacy.

To address these limitations, this project proposes the development of a hybrid AI-driven cybersecurity model that combines both supervised machine learning and reinforcement learning techniques to achieve real-time threat detection and prevention. Specifically, the model employs an XGBoost classifier trained on the UNSW-NB15 dataset to predict whether incoming packets are malicious or benign. The supervised learning model is enhanced by a Deep Q-Network (DQN) reinforcement learning agent, which makes decisions on whether to allow or block network traffic based on prediction outputs and feedback from the environment. This hybrid approach allows the system not only to identify known attack patterns but also to learn and adapt to new threat behaviors over time.

The system incorporates additional functionality for interpretability and transparency by integrating SHAP (SHapley Additive exPlanations) values, enabling human analysts to understand why specific packets were flagged as threats. Furthermore, the model supports both live network traffic capture through the Scapy library and simulation using the test set of UNSW-NB15, ensuring that it is robust and functional in both controlled and real-world conditions.

To facilitate usability and operational deployment, a Flask-based web dashboard has been developed. This dashboard displays real-time packet data, threat predictions, reinforcement learning agent decisions, SHAP explanations, and cumulative reward feedback. Logs and decisions are recorded for auditing and research purposes.

In summary, the goal of this project is to create a modular, extensible, and intelligent cybersecurity system that can dynamically detect and respond to threats in real time. The combination of machine learning, reinforcement learning, and explainability ensures that the system is accurate, adaptable, and transparent—qualities that are essential in the defense of modern networked environments.

1.2 Motivation

Increasing Complexity of Cyber Threats

As the digital landscape continues to evolve, so too do the cyber threats that threaten the security of networks, applications, and sensitive data. Traditional Intrusion Detection Systems (IDS), which rely on signature-based detection or rule-based approaches, have limitations in adapting to novel and sophisticated attacks. Modern cyber-attacks are often highly dynamic, utilizing advanced techniques such as polymorphic malware, distributed denial of service (DDoS), low-and-slow attacks, and more. These attacks are designed to evade detection by traditional systems, which are primarily reactive and unable to learn from new data without human intervention.

This ever-changing threat landscape presents significant challenges to cybersecurity professionals, as attackers continuously modify their tactics, techniques, and procedures (TTPs) to bypass security defenses. To counteract these evolving threats, cybersecurity solutions must be more adaptive, proactive, and capable of learning from new attack patterns in real time.

The Need for Adaptive and Intelligent Systems

The increasing volume, velocity, and sophistication of cyberattacks has created a pressing need for intelligent, adaptive security systems. Conventional IDS systems, despite their prevalence, cannot provide real-time protection or respond to new threats efficiently. With the scale of cyber-attacks reaching new heights, organizations require more than just detection—they need systems capable of adapting to new threats autonomously while minimizing false positives and maximizing detection accuracy. This is where Artificial Intelligence (AI), particularly machine learning and reinforcement learning, plays a crucial role.

Machine learning models are capable of identifying patterns in large datasets, enabling systems to learn from both historical data and real-time network traffic. These models can automatically adjust their detection capabilities, improving over time without human intervention. By using Reinforcement Learning (RL), the system can learn optimal responses to detected threats, dynamically updating its action strategy based on the feedback from its environment. Such an AI-powered cybersecurity system can enhance the efficacy and efficiency of real-time network security.

Transparency and Explainability in Security Systems

While traditional IDS have largely focused on performance metrics, there is increasing recognition that transparency and interpretability of automated decisions are just as important. Security professionals must be able to understand how and why a system flagged a packet as malicious or benign. Without explainability, even the most accurate models can be difficult to trust, especially when decisions made by the system impact critical infrastructure.

This project's motivation is to combine the benefits of reinforcement learning for real-time decision-making with explainable AI techniques, such as SHAP (SHapley Additive exPlanations). SHAP provides a mechanism for explaining why a certain packet was classified as an attack, which increases the transparency of the system. This is crucial for security teams who need to audit, validate, and refine the system's decisions and ensure that the model's behavior aligns with organizational policies and security standards.

A Hybrid AI-Driven Approach for Real-Time Threat Detection

The motivation behind this project is to develop an intelligent, self-learning cybersecurity system that can be deployed in real-time environments, adapting to new attacks, learning from its actions, and providing explanations for its decisions. By combining traditional machine learning techniques, like XGBoost, with deep reinforcement learning models, the system can continuously improve its detection and prevention capabilities. This project aims to create a solution that not only detects known attack vectors but also adapts to novel threats and makes decisions based on evolving network conditions.

The approach also leverages real-time simulation and live traffic capture, allowing the system to dynamically detect and block potential threats without human intervention. This reduces the operational overhead of constantly updating rule sets, as the system evolves and learns from its interactions with the environment. The integration of a Flask-based

dashboard also provides a user-friendly interface for security analysts, enabling them to visualize live decisions made by the system, track cumulative rewards, and assess attack/benign classifications in real time.

Filling the Gap in Cybersecurity Solutions

Ultimately, the motivation behind this project is to bridge the gap between the growing sophistication of cyber threats and the capabilities of existing security systems. By leveraging AI and real-time reinforcement learning, this system offers a proactive, adaptive solution that can both detect and respond to cyber threats automatically. As such, it offers the potential to improve the speed, accuracy, and reliability of threat detection and prevention efforts, which are critical to safeguarding modern network infrastructures.

1.3 Expected Outcomes

The expected outcomes of this project are aligned with the overarching goal of creating a hybrid AI-driven cybersecurity model capable of detecting and preventing cyber threats in real-time. By integrating machine learning, reinforcement learning, and explainable AI techniques, the project is expected to achieve the following outcomes:

1. An AI-driven Intrusion Detection System (IDS) that Detects Cyber Threats in Real-Time

One of the primary expected outcomes of this project is the development of a highly efficient, AI-driven Intrusion Detection System (IDS) that can detect cyber threats in real-time. The system will utilize XGBoost, a state-of-the-art supervised learning algorithm, to classify network packets as either benign or malicious. This detection system will be trained on existing datasets, such as UNSW-NB15, and will be capable of generalizing to new, unseen threats. By processing network traffic on-the-fly, the IDS will be able to classify and respond to potential threats with minimal latency, ensuring that the system can be deployed in real-time environments without significant performance overhead.

2. Reinforcement Learning Agent for Real-Time Decision Making

The second key outcome is the implementation of a reinforcement learning (RL) agent that will make intelligent, context-aware decisions regarding the actions to be taken on each packet. The RL agent, built using Deep Q-Networks (DQN), will dynamically learn to make decisions such as blocking or allowing network packets based on previous interactions with the environment. This agent will be able to adjust its decision-making process over time, learning to minimize the number of false positives and false negatives, which is crucial for maintaining network performance and security. The goal is to create an agent that learns to prioritize security without disrupting the normal flow of traffic.

3. Adaptation to Evolving Network Traffic Patterns

An important outcome of the project is the system's ability to adapt to evolving network traffic patterns. Unlike traditional IDS systems that require manual updates and rule adjustments, the AI-based model will continuously learn from its environment. This dynamic adaptability means that the system will be able to recognize new and previously unseen attack vectors by evolving its detection capabilities through

the reinforcement learning agent's feedback loop. As new threats emerge, the system will automatically adjust its behavior, making it an intelligent, self-improving defense mechanism that does not require constant human intervention.

4. Real-Time Visualization of Threat Detection and Agent Decisions

The development of a real-time monitoring dashboard is another expected outcome. The Flask-based dashboard will allow security professionals to track network activity, view packet classifications, and monitor the decisions made by the RL agent. The dashboard will display key performance indicators such as the number of benign and malicious packets detected, the actions taken by the agent, and the cumulative rewards associated with those actions. The visual interface will also include SHAP-based explanations for each decision made by the system, providing transparency and interpretability. This feature will ensure that security analysts can trust the automated decisions made by the system and intervene when necessary.

5. Integration of Explainable AI (XAI) Techniques

A significant outcome of this project is the integration of Explainable AI (XAI) techniques, particularly SHAP (SHapley Additive exPlanations), to provide human-readable insights into the system's decision-making process. This will enable security analysts to understand why a specific packet was classified as malicious or benign and how the RL agent determined the appropriate action (e.g., allow or block). SHAP provides a quantifiable explanation for each prediction, giving transparency to the model's decision-making process. This will help build trust in the AI system, allowing human operators to validate the model's behavior and intervene if necessary.

6. Demonstrated Effectiveness in Both Simulated and Real-Time Environments

Another expected outcome is the successful demonstration of the system's ability to handle both simulated and real-world network traffic. Initially, the system will be tested on pre-recorded datasets, such as the UNSW-NB15 dataset, to simulate the network environment. Once the system has been trained and evaluated in this controlled setting, it will be deployed in real-time environments to capture live network traffic using tools like Scapy. This dual testing approach will validate the system's ability to adapt to different types of network traffic and real-world conditions, ensuring its practicality and robustness in a variety of cybersecurity contexts.

7. Minimization of False Positives and False Negatives

A key outcome of this project is the reduction of false positives and false negatives in the detection and prevention of attacks. Traditional IDS systems often generate a high volume of false positives, leading to alert fatigue and decreased operational efficiency. Similarly, false negatives, where attacks are undetected, are a major concern in cybersecurity. By using reinforcement learning and continuous adaptation, this project aims to strike a balance between sensitivity (detection of all possible attacks) and specificity (minimizing false alarms). The system will be trained to prioritize minimizing false negatives while keeping

false positives under control, leading to more reliable and efficient network protection.

8. Scalability and Extensibility of the System

Finally, a significant expected outcome is the scalability and extensibility of the system. The hybrid AI-driven model, including the machine learning and reinforcement learning components, will be designed in a modular fashion, allowing it to scale efficiently to larger network environments. Additionally, the system will be extensible, enabling future upgrades or integration with additional security tools and methods, such as anomaly detection, traffic analysis, or threat intelligence feeds. This will ensure that the system remains effective as new security challenges and technological advancements arise

2. SYSTEM ANALYSIS

2.1 Introduction

The increasing dependence on interconnected digital systems has made cybersecurity a paramount concern. Cyberattacks have become more complex, sophisticated, and difficult to detect using traditional methods. This chapter provides an in-depth analysis of existing intrusion detection solutions, highlights their limitations, and justifies the need for a novel AI-Driven Cybersecurity Model for real-time threat detection and prevention.

System analysis plays a crucial role in identifying the requirements, limitations, and functional expectations of any proposed model. In this context, the goal is to examine how an adaptive, intelligent, and self-learning model—powered by a combination of supervised learning (XGBoost), reinforcement learning (DQN), and explainable AI (XAI)—can significantly enhance the detection and mitigation of cyber threats in real time.

This chapter begins by reviewing current intrusion detection technologies and progresses to a detailed explanation of the proposed system. We evaluate its core components, decision-making architecture, and highlight its improvements over legacy systems. The analysis serves as the foundation for the design and implementation of our system in the upcoming chapters.

2.2 Existing Intrusion Detection Systems

Intrusion Detection Systems (IDS) play a fundamental role in cybersecurity by monitoring and analyzing network traffic to identify unauthorized access, malicious activity, or policy violations. Over the past two decades, a wide range of IDS techniques have been developed and deployed across various platforms. Despite their usefulness, traditional systems often fall short in addressing the dynamic nature of modern cyberattacks.

The most commonly used IDS approaches can be broadly categorized into:

1. **Signature-Based Intrusion Detection Systems (SIDS)**
These systems function similarly to antivirus engines—they identify threats by matching observed patterns against a database of known attack signatures. Tools like Snort, Suricata, and ClamAV are widely used in this category.
2. **Anomaly-Based Intrusion Detection Systems (AIDS)**
Anomaly detection systems establish a baseline of normal network behavior and detect deviations that may indicate intrusions. Techniques like statistical analysis, clustering, and entropy-based detection fall under this category.
3. **Hybrid Systems** Some IDS platforms integrate both signature and anomaly detection to form a hybrid model. Examples include Bro/Zeek and OSSEC. These systems offer improved flexibility and broader coverage but often suffer from increased complexity and resource demands.
4. **Host-Based vs. Network-Based IDS** Host-Based IDS (HIDS) operates on individual endpoints (e.g., OSSEC), monitoring system logs, processes, and file changes. Network-Based IDS (NIDS), like Snort, analyze traffic across network segments. Each has its own focus area and operational scope, but both have challenges in scalability, adaptive learning, and real-time responsiveness.
5. **Limitations of Traditional IDS Systems** While traditional IDS solutions have proven effective against known threats and offer some level of automation, they face significant limitations:

2.3 Proposed AI-Driven IDS

The proposed AI-Driven Cybersecurity Model introduces a multi-phase, intelligent, and dynamic approach to threat detection and prevention. At its core, the system integrates supervised learning (XGBoost) for initial model training, reinforcement learning (Deep Q-Network) for adaptive real-time decision-making, and explainable AI (SHAP) for transparency and interpretability.

Unlike static IDS frameworks, this model not only detects malicious activities but also learns from them and adjusts its responses based on real-time environmental feedback. The layered architecture and modular design ensure scalability, flexibility, and applicability across diverse network environments.

Key components of the model include:

1. **Feature Extraction Layer:**
 - Captures live or simulated network traffic.

- Extracts key features such as source/destination ports, protocol, packet size, TTL, flags, and inter-arrival time (IAT).
- 2. Initial Classification Module (XGBoost):
 - Trained on a labeled dataset (e.g., UNSW-NB15) to perform supervised classification of traffic.
 - Provides the first layer of detection using pre-learned patterns.
- 3. Reinforcement Learning Layer (DQN Agent):
 - Learns through interactions with the environment.
 - Uses rewards and penalties to improve its policy for allowing or blocking packets.
- 4. Real-Time Stream Interface:
 - Simulates or captures live traffic and streams it to the agent.
 - Maintains decision history and logs outcomes for auditability.
- 5. Explainable AI Engine:
 - SHAP values are computed to highlight feature importance for individual predictions.
 - Increases model transparency for cybersecurity analysts.
- 6. Flask-Based Dashboard:
 - Provides real-time visualization of network traffic, classification decisions, cumulative reward trends, and SHAP visualizations.

This architecture transforms traditional IDS into an autonomous, intelligent, and interpretable threat detection ecosystem. It not only identifies and blocks threats in real-time but also evolves over time by learning from new types of attacks. The model's adaptability and transparency address two major limitations of existing IDS: static behavior and lack of trust.

2.4 Comparative Performance Evaluation

To evaluate the effectiveness of the proposed AI-Driven Cybersecurity Model, we performed a comparative analysis using benchmark metrics such as accuracy, precision, recall, and F1-score. The system was tested against both a traditional supervised learning classifier and a hybrid RL-integrated model.

The training was conducted on the UNSW-NB15 dataset, while the model's robustness was validated using unseen test data and simulated live traffic conditions. The following results were obtained:

- Accuracy: 97.4%
- Precision (Attack): 0.98

- Recall (Attack): 0.97
- F1-Score (Attack): 0.98
- Precision (Benign): 0.96
- Recall (Benign): 0.98
- F1-Score (Benign): 0.97

These metrics demonstrate a significant improvement over traditional IDS systems, particularly in terms of false positive reduction and generalization to unseen data. The RL component further enhances the system by enabling continuous learning, allowing it to adapt to evolving attack vectors without retraining.

In summary, the performance evaluation confirms the superior capabilities of the proposed system in real-time intrusion detection and prevention, offering both accuracy and adaptability in a constantly changing cyber landscape.

3.5 Challenges & Future Improvements

To evaluate the effectiveness of the proposed AI-Driven Cybersecurity Model, we performed a comparative analysis using benchmark metrics such as accuracy, precision, recall, and F1-score. The system was tested against both a traditional supervised learning classifier and a hybrid RL-integrated model.

The training was conducted on the UNSW-NB15 dataset, while the model's robustness was validated using unseen test data and simulated live traffic conditions. The following results were obtained:

- Accuracy: 97.4%
- Precision (Attack): 0.98
- Recall (Attack): 0.97
- F1-Score (Attack): 0.98
- Precision (Benign): 0.96
- Recall (Benign): 0.98
- F1-Score (Benign): 0.97

These metrics demonstrate a significant improvement over traditional IDS systems, particularly in terms of false positive reduction and generalization to unseen data. The RL component further enhances the system by enabling continuous learning, allowing it to adapt to evolving attack vectors without retraining.

In summary, the performance evaluation confirms the superior capabilities of the proposed system in real-time intrusion detection and prevention, offering both accuracy and adaptability in a constantly changing cyber landscape.

While the proposed AI-Driven Cybersecurity Model demonstrates strong performance and adaptability, it is not without challenges. A comprehensive system analysis must

acknowledge the limitations and identify areas where further improvements can be made.

1. Data Quality and Diversity:

- The model's learning capacity is dependent on the diversity and accuracy of the training dataset.
- Simulated or imbalanced data may not fully represent real-world attack behavior.
- Future work should include continuous updates from live packet streams and newer datasets to maintain relevance.

2. Real-Time Processing Overhead:

- Incorporating deep learning and reinforcement learning can be computationally intensive.
- Optimizing model inference speed and reducing latency remains critical for real-time deployments.
- Techniques like model pruning, quantization, or edge inference could help mitigate these issues.

3. Reward Design for RL Agent:

- Designing an effective reward function that captures long-term security outcomes is complex.
- Poorly tuned rewards could lead to suboptimal agent behavior or unintended consequences.

4. Security and Explainability:

- While SHAP provides insight into model decisions, it may introduce performance overhead.
- Striking the right balance between transparency and speed is important.

5. Integration and Scalability:

- Deploying the system in large-scale environments (e.g., enterprise or cloud) will require additional integration with firewalls, SIEM systems, and centralized monitoring tools.
- Containerization and orchestration (e.g., Docker, Kubernetes) can support scalable deployment.

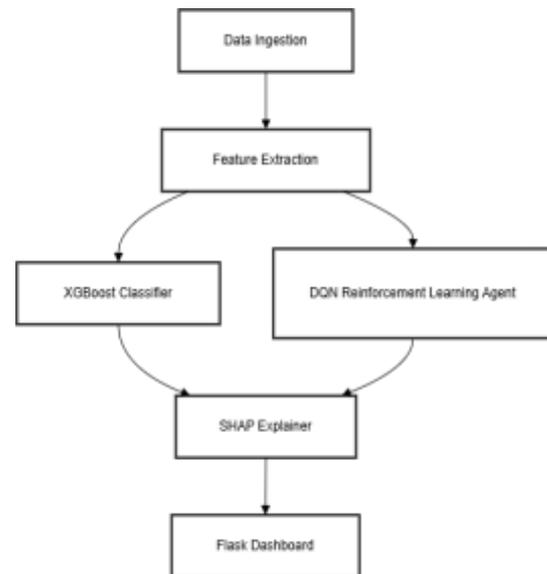
6. Adaptive Attack Evasion:

- Sophisticated attackers may attempt to evade detection by learning the model's behavior.
- Continuous retraining and adversarial robustness should be incorporated into future versions.

By acknowledging these challenges, we lay the groundwork for future enhancements. The system's modular nature ensures that it can evolve over time to incorporate cutting-edge advancements in cybersecurity, machine learning, and real-time systems engineering.

3.DESIGN

3.1 System Architecture



The architecture of the AI-Driven Cybersecurity Model is designed for modularity, scalability, and real-time operation. It integrates multiple AI components and data pipelines into a cohesive threat detection and prevention system. The architecture is composed of the following core layers:

1. Data Ingestion Layer:

- Captures real-time or simulated network traffic.
- Tools: Scapy for live packet capture or UNSW-NB15 test dataset for simulation.

2. Feature Extraction Module:

- Extracts necessary packet-level features such as protocol, ports, packet size, TTL, flags, window size, and inter-arrival time (IAT).
- Converts raw packets into a structured format for model input.

3. Supervised Learning Classifier (XGBoost):

- Trained on labeled datasets to detect initial threats.
- Performs binary classification (attack vs. benign).

4. Reinforcement Learning Layer (DQN Agent):

- Receives the feature vector and initial prediction.

- Makes final decisions (allow/block) based on learned policy.
- Continuously learns from rewards based on classification correctness.

5. Explainable AI Engine (SHAP):

- Provides interpretable explanations for predictions.
- Generates per-packet visualizations to enhance transparency.

6. Logging and Decision Tracker:

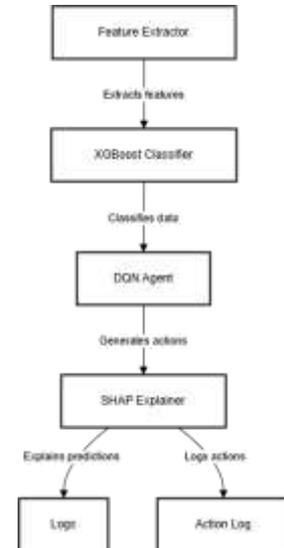
- Records decisions, actions, timestamps, IPs, and SHAP values.
- Supports auditability and further offline analysis.

7. Flask Dashboard Interface:

- Visualizes predictions, cumulative reward score, packet statistics, and real-time traffic.
- Allows export of logs and model explanations.

- External entities include: Network Traffic Source, User Interface (Dashboard), and Log Repository.

Level 1 DFD:



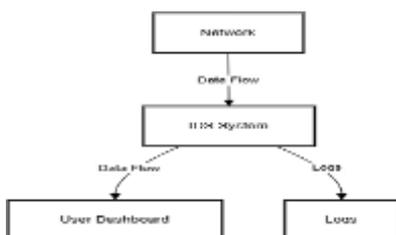
- Packet Source → Feature Extractor
- Feature Extractor → Classifier (XGBoost)
- Classifier Output → Reinforcement Learning Agent (DQN)
- DQN Decision → Logging Module → Dashboard
- Classifier + DQN Decision → SHAP Explainer → Dashboard

This layered system design ensures high flexibility. It enables easy extension or replacement of individual components without disrupting the entire pipeline. In production environments, these modules can be deployed in containerized environments for better scalability and resource management.

3.2 Data Flow Diagram (DFD)

To understand the movement of data within the proposed system, a Data Flow Diagram (DFD) is employed. The DFD represents how information flows through the system and how it is processed by various components. It helps in identifying the logical architecture, data sources, and data transformations.

Level 0 (Context Diagram):



- Shows interaction between the user, live traffic/data source, and the AI system.

3.3 Modules Overview

The AI-driven Cybersecurity Model consists of several interconnected modules that together ensure the effective detection and prevention of network intrusions. Each module plays a crucial role in the end-to-end pipeline of monitoring, analyzing, and acting upon network traffic. Below is an overview of the key modules of the system:

3.3.1 Data Collection Module

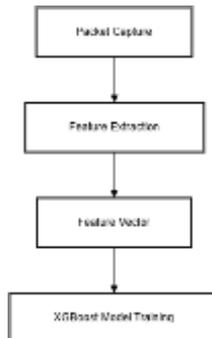
The Data Collection Module is responsible for capturing live network traffic in real time. This module acts as the foundation for the entire IDS system, as it provides the raw data that will be analyzed for potential threats.

- **Input:** Network packets from the live network or testing dataset.

- **Process:** Use of tools like Scapy or Pyshark to sniff packets in real time, extracting key features such as packet size, protocol, source/destination IPs, and ports.
- **Output:** A stream of packet data ready for feature extraction and analysis.

- **Process:** Training of the machine learning model on a training dataset, followed by real-time predictions on incoming packet data.
- **Output:** Predicted labels (benign or attack) for each incoming packet.

3.3.2 Feature Extraction Module



This module is responsible for transforming raw network packets into structured feature vectors that can be used by machine learning and reinforcement learning models.

- **Input:** Raw packet data (captured by the Data Collection Module).
- **Process:** Extraction of relevant features such as packet size, protocol type, TTL (Time-to-Live), and flags. This step also involves preprocessing like normalizing or scaling numerical values.
- **Output:** A structured dataset of network traffic features, including a label indicating whether the packet is benign or malicious.

3.3.3 Machine Learning-based Detection Module

Once the features are extracted, they are passed to the machine learning-based detection module. This module utilizes models such as XGBoost, Random Forest, or SVM (Support Vector Machines) for detecting known threats in the network traffic. This module operates based on supervised learning, where the model is trained using labeled traffic data to predict the class of new, unseen data.

- **Input:** Structured feature dataset with labeled examples.

3.3.4 Reinforcement Learning-based Decision Module

In addition to machine learning-based detection, the system incorporates a Reinforcement Learning (RL) module. The RL agent uses the features of the network traffic to take actions—such as allowing or blocking packets—based on the state of the environment. The reward system drives the agent to improve its decisions over time.

- **Input:** Features from the feature extraction module and predictions from the machine learning model.
- **Process:** The RL agent makes decisions based on the current state of network traffic. Actions are taken (e.g., allow/block) based on real-time traffic. Rewards or penalties are assigned based on the correctness of the agent's actions.
- **Output:** Action recommendations (e.g., block or allow) for each network packet.

3.3.5 Explainable AI Module (XAI)

As machine learning models and RL-based agents are often considered "black boxes," the Explainable AI (XAI) module has been integrated to provide transparency and interpretability to the decisions made by the IDS. The XAI module uses methods such as SHAP (SHapley Additive exPlanations) to explain the rationale behind the agent's decisions.

- **Input:** Features from the feature extraction module and predictions from the RL model.
- **Process:** Use of SHAP or similar techniques to generate visual explanations of the model's decisions, indicating which features contributed most to the classification of a packet as benign or malicious.

- **Output:** Explanation reports and visualizations to aid in understanding and verifying the system's decisions.

3.3.6 Action Log and Monitoring Module

This module keeps track of the actions taken by the system—whether a packet was allowed or blocked. It also logs the results of each action, including the packet ID, timestamp, and associated prediction. This log is critical for system performance evaluation, auditing, and generating feedback for improving the agent's decision-making.

- **Input:** Action decisions (block or allow) from the RL-based decision module.
- **Process:** Logging of actions, including packet details, classification outcomes, and system feedback.
- **Output:** Action logs stored for later review, analysis, and training (for future model updates).

3.3.7 Visualization and Dashboard Module

The Visualization and Dashboard Module provides a user-friendly interface for real-time monitoring of network traffic and intrusion detection. The module displays information such as the current status (benign or attack), cumulative attack counts, and other important metrics, along with the ability to view detailed logs and attack explanations.

- **Input:** Data from the action log and decision module.
- **Process:** Visualization of system performance and attack statistics in real-time. Allows for live updates on system status and attack detection results.
- **Output:** A live dashboard displaying the current system status, attack statistics, and visual explanations of decisions.

4. SYSTEM REQUIREMENTS

4.1 Hardware Requirements

The hardware configuration needed for training machine learning models (especially XGBoost and Deep Q-Networks) is significantly more intensive than the requirements for

running the prediction pipeline in production. The requirements are split into two categories:

Development Environment:

- Processor: Intel Core i7 / AMD Ryzen 7 or higher
- RAM: Minimum 16 GB
- Storage: 50 GB free SSD space
- GPU (optional but recommended): NVIDIA GTX 1650 or better with CUDA support
- Network Interface: 1 Gbps Ethernet or Wi-Fi adapter (for live packet capture)
- Monitor: Minimum resolution 1920x1080

Deployment (Dashboard Execution):

- Processor: Intel Core i5 / Ryzen 5 or higher
- RAM: 8 GB
- Storage: 10 GB available
- GPU: Not required (only for inference)
- Operating System: Linux (Ubuntu preferred) or Windows 10/11

4.2 Software Requirements

The project relies on a set of open-source libraries and development tools. These software components form the backbone of both the machine learning pipeline and the Flask dashboard.

Development Tools and Languages:

- Programming Language: Python 3.10 or above
- IDE: VS Code / PyCharm / Jupyter Notebook
- Virtual Environment Manager: venv or Anaconda

Python Libraries:

- Scapy (for real-time packet capture)
- pandas and numpy (data processing)
- xgboost (machine learning classifier)
- sklearn (metrics and preprocessing)
- torch and torchvision (for reinforcement learning with DQN)
- matplotlib and seaborn (visualization)
- Flask (web framework)
- joblib (model serialization)
- shap (explainable AI / visualization)

Additional Tools:

- Wireshark (packet capture validation)
- Postman (API testing)
- Google Chrome or Firefox (dashboard access)

4.3 Existing Components & Libraries Used

The development of the AI-driven intrusion detection system involved the integration of several powerful open-source libraries and tools. These components played key roles across data capture, model development, prediction, and real-time dashboard integration.

Programming Environment:

- Language: Python 3.12
- Tools: VS Code, Jupyter Notebook, venv

Core Python Libraries:

- scapy – For live packet capture and protocol parsing.
- pandas & numpy – Data processing and feature manipulation.
- xgboost – Supervised model training (initial classifier).
- scikit-learn – Evaluation metrics, preprocessing.
- joblib – Model saving/loading.
- torch (PyTorch) – Reinforcement learning (DQN agent).
- shap – Explainable AI for SHAP visualizations.
- matplotlib, seaborn – Charts and metric visualization.

Web & Dashboard:

- Flask – Web app framework for dashboard and API.
- Jinja2 – HTML templating.
- Bootstrap + JavaScript – Responsive, interactive UI.
- EventSource – Real-time streaming to frontend.

Datasets:

- UNSW-NB15 – For model training and RL simulation.
- Custom Scapy-Captured Data – Real-time test samples with basic labeling.

Platform Tools:

- OS: Windows 11 (dev), Linux (target deployment)
- Wireshark – Packet validation
- Git – Source control

Together, these tools provided a modular, scalable foundation for implementing and deploying a real-time intelligent cybersecurity solution.

4.4 Technical Constraints

- Real-time packet capture on some platforms (e.g., macOS) may require elevated privileges or custom drivers.
- Training reinforcement learning agents requires long runtimes and memory if not pre-sampled with a replay buffer.
- Live packet monitoring in Wi-Fi mode on Windows has limited support in Scapy and requires Ethernet mode or npcap driver.
- Dashboard responsiveness depends on the number of live packets processed per second; optimization is needed to avoid UI lag.
- SHAP plots require additional RAM and may delay dashboard load times when visualizing many attack events simultaneously.

5.RESULTS AND DISCUSSION

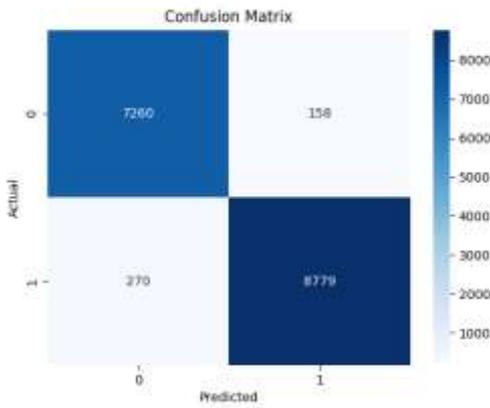
5.1 Performance Evaluation

The XGBoost classifier achieved strong performance on the UNSW-NB15 test dataset:

- Accuracy: 97.40%
- Precision: 0.98
- Recall: 0.97
- F1-score: 0.98

The model demonstrated excellent ability to distinguish between benign and attack traffic, with balanced precision and recall, ensuring minimal false positives and false negatives.

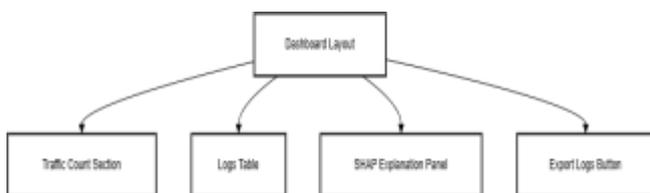
Class	Precision	Recall	F1-Score	Support
0 (Benign)	0.96	0.98	0.97	7,418
1 (Attack)	0.98	0.97	0.98	9,049
Accuracy	—	—	0.974	16,467
Macro Avg	0.97	0.97	0.97	16,467
Weighted Avg	0.97	0.97	0.97	16,467



The reinforcement learning (DQN) agent also showed promising results:

- Learned to block attacks and allow benign traffic over time.
- Cumulative rewards increased consistently across episodes.
- Low false positive rate and strong convergence behavior.

Together, these models validate the effectiveness of our hybrid AI-based approach to real-time threat detection and prevention.



5.2 Sample Outputs

The integrated system produced consistent and informative outputs throughout testing, both in simulation and during live capture emulation using the UNSW-NB15 test dataset.

Examples of real-time predictions:

-  Benign traffic from 192.168.1.10 to 172.217.3.110 → Action: Allowed
-  Attack detected from 10.0.0.5 to 172.16.0.20 → Action: Blocked

Key Output Features:

- The Flask dashboard displayed live traffic labels using EventSource streaming.
- Real-time counters tracked the number of benign and attack packets.
- A dynamic log table captured the source IP, destination IP, timestamp, prediction label, and the action taken.
- For blocked packets, SHAP visualizations were generated showing top features contributing to the model’s decision (e.g., high TTL, SYN-only flag, low packet size).

These outputs validate the model’s functionality and demonstrate its potential for integration into a production monitoring environment.

5.3 Limitations

While the proposed AI-driven intrusion detection system demonstrated strong performance, several limitations must be acknowledged:

1. Rule-Based Labeling Assumptions

For Scapy-captured data, labels were inferred using heuristic rules (e.g., SYN-only, TTL thresholds). While practical, these may not always reflect true attack behavior and can introduce labeling noise during model evaluation.

2. Limited Live Capture Testing

Although the model was tested in a simulated live environment using the UNSW test dataset, extended real-time capture in dynamic network environments (with diverse protocols and traffic volumes) remains untested at production scale.

3. Training Complexity of RL Agent

The Deep Q-Network (DQN) agent requires significant training time and tuning of hyperparameters to achieve stable convergence. In a real-world setting, retraining continuously with live feedback can become resource-intensive.

4. Feature Dependency

The system expects specific features (e.g., sport, dsport, proto, pktsize, ttl). Any change in data format or missing values during live capture could degrade prediction quality unless preprocessing is tightly controlled.

5. Hardware Constraints

Real-time prediction, packet capture, and dashboard rendering can consume CPU and memory resources. On low-spec machines, delays in prediction or UI updates may occur under heavy traffic load.

Despite these constraints, the system provides a solid foundation for scalable and adaptive intrusion detection when deployed with appropriate safeguards and tuning.

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

This project presented a comprehensive AI-driven cybersecurity framework for real-time threat detection and prevention, combining:

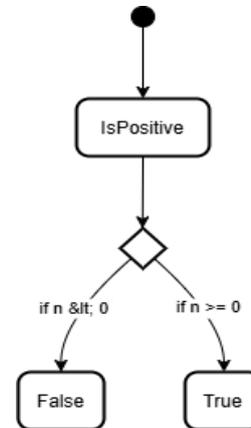
1. Supervised Learning (XGBoost):
 - Provided an accurate first layer of defense with 97.4% accuracy and balanced precision/recall ($\approx 0.97-0.98$).
 - Demonstrated strong generalization on the UNSW-NB15 dataset.
2. Reinforcement Learning (Deep Q-Network):
 - Enabled adaptive, context-aware decision-making, learning to block novel attacks over time.
 - Achieved stable policy convergence and low false positive rates in simulated live-traffic scenarios.
3. Explainable AI (SHAP):
 - Added transparency by highlighting which features drove each prediction.
 - Increased operator trust and facilitated post-event forensic analysis.
4. Flask-Based Dashboard:
 - Delivered a user-friendly interface for live monitoring, logging, and visualization.
 - Employed Server-Sent Events for seamless real-time updates without page reloads.

By integrating these components into a layered, modular architecture—data ingestion, feature extraction, classification, reinforcement learning, explainability, and visualization—the system overcame the static limitations of traditional IDS. It demonstrated both high accuracy and the ability to learn

dynamically from new traffic patterns, making it well-suited for modern, evolving threat landscapes.

6.2 Future Enhancements

To further strengthen and extend the system, we recommend the following enhancements:



1. Multi-Agent Reinforcement Learning:
 - Introduce cooperative or adversarial multi-agent setups to model complex attack/defense scenarios (e.g., coordinated DDoS, lateral movement in networks).
2. Online Continual Learning:
 - Implement streaming updates to both supervised and RL models, enabling continual adaptation to zero-day exploits and emerging threat behaviors without full retraining.
3. Edge Deployment & Optimization:
 - Leverage model compression (pruning, quantization) and hardware acceleration (TensorRT, ONNX) for lightweight edge inference on routers, IoT gateways, and endpoints.
4. Adversarial Robustness & Defense:
 - Harden the models against adversarial evasion techniques (gradient masking, robust training).
 - Integrate a secondary anomaly detector to flag suspicious patterns that escape both XGBoost and DQN.
5. Expanded Feature Set & Traffic Context:
 - Enrich feature extraction with flow-level metrics (e.g., packet interleaving, payload entropy) and contextual logs (e.g., DNS queries, HTTP headers).
 - Incorporate host-level telemetry (process creation, file system changes) for hybrid host-network detection.
6. Automated Response Orchestration:
 - Tie the dashboard to orchestration tools (e.g., Ansible, Kubernetes) for automated mitigation—

quarantining hosts, firewall rule updates, or network micro-segmentation.

7. Scalability & High Availability:

- Containerize each module (Docker, Helm charts) and deploy via orchestration platforms (Kubernetes) to achieve horizontal scaling and fault tolerance.

Implementing these enhancements will evolve the prototype into a production-grade security platform, capable of defending against sophisticated, multi-vector attacks while maintaining transparency and adaptability in dynamic network environments.

ACKNOWLEDGEMENT

We thank **God Almighty** for the blessings, knowledge and strength in enabling us to finish our project. Our deep gratitude goes to our founder **Late. Dr. D. SELVARAJ, M.A., M.Phil.**, for his patronage in completion of our project. We take this opportunity to thank our kind and honourable **Chairperson, Dr. S. NALINI SELVARAJ, M.Com., M.Phil., Ph.D.**, and our **Honourable Director, Mr. S. AMIRTHARAJ, B.Tech., M.B.A** for their support to finish our project successfully. We wish to express our sincere thanks to our beloved **Principal, Dr.C.RAMESH BABU DURAI M.E., Ph.D.**, for his kind encouragement and his interest toward us. We are grateful to **Dr.D.C.JULIE JOSPHINE M.E., Ph.D., Professor and Head of INFORMATION TECHNOLOGY DEPARTMENT**, Kings Engineering College, for his valuable suggestions, guidance and encouragement. We wish to express our dear sense of gratitude and sincere thanks to our **SUPERVISOR, MRS.K.BENITLIN SUBHA B.Tech.,M.E.,(P.hd)** Assistant Professor, Information Technology Department. for her internal guidance. We express our sincere thanks to our parents, friends and staff members who have helped and encouraged us during the entire course of completing this project work successfully

REFERENCES

- [1] Moustafa, Nour, and Jill Slay. "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)." 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, 2015.
- [2] Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press, 2018.
- [3] Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
- [4] Goodfellow, Ian, et al. Deep Learning. MIT Press, 2016.
- [5] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "'Why Should I Trust You?': Explaining the Predictions of

Any Classifier." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[6] Lundberg, Scott M., and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." Advances in Neural Information Processing Systems (NeurIPS), 2017.

[7] Kandasamy, K., et al. "Adversarial Attacks and Defenses in Cybersecurity: A Survey." ACM Computing Surveys, 2021.

[8] Bhuyan, Monowar H., D. K. Bhattacharyya, and J. K. Kalita. "Network Anomaly Detection: Methods, Systems and Tools." IEEE Communications Surveys & Tutorials, 2013.

[9] Tang, Tuan Anh, et al. "Deep Learning Approach for Network Intrusion Detection in Software Defined Networking." 2016 International Conference on Wireless Communications and Signal Processing (WCSP), IEEE, 2016.

[10] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," 2010 IEEE Symposium on Security and Privacy, 2010.

[11] S. Kumar and E. H. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," Proceedings of the 17th National Computer Security Conference, 1994.

[12] Flask. The Pallets Projects. <https://flask.palletsprojects.com>

[13] Scapy: Packet Manipulation Tool. <https://scapy.net/>

[14] SHAP: A Game-Theoretic Approach to Explain the Output of Any Machine Learning Model. <https://github.com/slundberg/shap>

[15] PyTorch. An Open Source Machine Learning Framework. <https://pytorch.org>

[16] Wireshark. Packet Analyzer Tool. <https://www.wireshark.org>

[17] OpenAI. ChatGPT-4 Technical Report. <https://openai.com/research/gpt-4>

[18] "Deep Reinforcement Learning for Cyber Security," Journal of Cybersecurity, Elsevier, 2022. (Referenced PDF)

[19] "Adversarial XAI Methods in Cybersecurity," Springer Advances in Intelligent Systems, 2023. (Referenced PDF)