International Journal of Intelligent Information Systems
2025, Vol. 14, No. 4, pp. 70-76
https://doi.org/10.11648/j.ijiis.20251404.11

Research Article

AI-Powered Intrusion Detection System with Honeypot Integration

Aditya Nimmagadda, Shideh Yavary Mehr*

School of Cybersecurity, Old Dominion University, Norfolk, United States

Abstract

In response to the increasing complexity and frequency of cyber threats, this project presents an AI-powered Intrusion Detection System (IDS) enhanced by honeypot integration. Traditional IDS techniques, heavily reliant on signature-based detection, often fail to recognize novel or polymorphic attacks, leaving systems vulnerable to zero-day exploits and advanced persistent threats (APTs). To address this limitation, the proposed system leverages machine learning models 'both supervised and unsupervised' trained on data captured from a controlled virtual environment simulating real-world scenarios. Honeypots, specifically the Cowrie honeypot, are deployed to lure attackers and collect rich behavioral data, which in turn enhances the system's detection capabilities by capturing indicators of compromise (IOCs) and attack patterns that traditional datasets may miss. The architecture consists of a multi-VM setup ensuring isolated and secure experimentation, preventing compromise of production systems during testing. Using Random Forest and Logistic Regression models, along with Isolation Forest for anomaly detection, the system achieves high detection accuracy, minimal false positives, and strong adaptability to emerging threats. Data preprocessing and feature engineering are applied to ensure model robustness, while hyperparameter tuning further optimizes performance. A Flask-based real-time API enables live threat classification and rapid response, and integration with Kibana and Power BI dashboards provides comprehensive visualization, monitoring, and historical analysis of network events. The system is designed for scalability and continuous improvement through an automated retraining pipeline, allowing it to adapt autonomously as new threat intelligence becomes available. This ensures that detection capabilities evolve alongside the changing tactics, techniques, and procedures (TTPs) of malicious actors. Future enhancements will focus on incorporating deep learning approaches such as Long Short-Term Memory (LSTM) networks for temporal sequence analysis and Convolutional Neural Networks (CNN) for traffic pattern recognition, further strengthening the IDS against sophisticated attacks. This work demonstrates a proactive, intelligent, and adaptable IDS solution capable of defending against both known and unknown threats, offering a foundation for next-generation AI-driven cybersecurity systems.

Keywords

Feature Engineering, Data Preprocessing, Cyber Threat Intelligence, Isolation Forest, Deep Learning (LSTM, CNN), Kibana, Power BI Dashboard, Automated Retraining Pipeline

*Corresponding author: syavarym@odu.edu (Shideh Yavary Mehr)

Received: 30 July 2025; **Accepted:** 15 August 2025; **Published:** 3 September 2025

1. Background

Artificial Intelligence (AI), particularly through the use of machine learning (ML), has significantly transformed the landscape of Intrusion Detection Systems (IDS). Traditional IDS solutions often rely on predefined signatures or static rules, which limit their ability to detect novel or evolving threats. In contrast, AI-powered IDSs offer a dynamic and intelligent approach to cybersecurity by analyzing complex network traffic patterns and user behaviors to detect anomalies that may signal a potential intrusion.

One of the key advantages of integrating AI into IDS frameworks is the ability to perform real-time monitoring and automated threat detection. AI algorithms can process vast volumes of data at high speed, identifying subtle deviations from normal activity that human analysts or conventional systems might overlook. This capability not only increases the speed of detection but also reduces the rate of false positives, which are a common issue in traditional IDS setups.

Moreover, AI systems are inherently adaptive. With techniques such as deep learning, supervised and unsupervised ML models can be trained on large, diverse datasets, allowing them to distinguish accurately between legitimate and malicious network behavior. These models improve over time, learning from new threat data, and can quickly adapt to emerging attack vectors, such as zero-day exploits or polymorphic malware without requiring manual rule updates.

Additionally, AI-powered IDSs support integration with broader cybersecurity infrastructures, including Security Infor- mation and Event Management (SIEM) platforms. This inte- gration facilitates a more comprehensive and correlated view of system security, enhancing threat detection and enabling faster, more informed incident response.

By automating many of the processes traditionally performed by human analysts, AI not only enhances detection capabilities but also frees up cybersecurity professionals to focus on higher-level strategic tasks, such as threat hunting, risk assessment, and incident response planning. As cyber threats continue to grow in complexity, AI-enabled IDSs represent a scalable, intelligent, and proactive defense mechanism essential for modern network security.

2. Introduction

In the evolving field of cybersecurity, traditional intrusion detection systems (IDS) struggle to adapt to new attack patterns, especially those that are stealthy, polymorphic, or zero-day in nature. These systems typically rely on signature-based techniques, which, while effective against known threats, fail to detect novel or slightly modified attack vectors. As cyber threats continue to grow in complexity and frequency [8], there is a pressing need for intelligent, adaptive, and proactive defense mechanisms.

This project introduces an innovative solution by integrating artificial intelligence (AI) and machine learning (ML) techniques with honeypot technology. Honeypots act as decoy systems that imitate real services or devices, specifically designed to lure cyber attackers and monitor their behavior. When strategically deployed, honeypots not only distract malicious actors from valuable systems but also serve as rich sources of real-time attack data. These interactions offer deep insights into adversarial tactics, techniques, and procedures (TTPs), which can be used to enhance detection accuracy and model learning. By capturing both benign and malicious traffic within a controlled, virtualized environment, the system trains itself to recognize traffic patterns and detect anomalies. This makes it possible to classify known attacks with high accuracy while also identifying unknown or emerging threats. The AI com- ponent leverages both supervised and unsupervised learning approaches, enabling the model to improve over time through exposure to new data.

The project was implemented in a safe, isolated virtual lab environment using multiple virtual machines (VMs) to simulate real world scenarios. One VM acts as the attacker using tools like Nmap and Metasploit, while another serves as the victim system running the honeypot and traffic monitoring tools. An optional third VM is dedicated to processing data and training the machine learning models. This structure ensures safe experimentation and provides a robust testbed for cybersecurity defense mechanisms.

In general, the goal is to develop a smart, responsive, and scalable IDS that adapts to evolving cyber threats while offering enhanced visibility into attack behavior through honeypot integration.

3. Implementation

For my project, I focused on designing and implementing an intelligent intrusion detection system (IDS) that utilizes machine learning techniques and honeypot integration to detect, classify, and analyze network based cyber threats. The key objectives of my work are:

Building a Machine Learning-Based IDS: I aim to develop a predictive model using supervised learning techniques to iden- tify network intrusions. The model will distinguish between normal (benign) traffic and various malicious activities (e.g., scans, exploits, brute-force attempts). I will train the model using both real and simulated network data to ensure it's both practical and robust.

Capturing Network Traffic: I plan to simulate a realistic network environment by generating both normal and malicious traffic. Normal traffic will involve activities like web browsing and file transfers, while malicious traffic will be generated using tools such as Nmap and Metasploit. I will capture this traffic using packet analysis tools like TCPDump and Tshark to create a comprehensive dataset for model training.

Deploying Honeypots: I will implement a low-interaction

honeypot (Cowrie) on a victim machine, mimicking vulnerable services like SSH or Telnet. The honeypot will attract attackers, logging their actions and providing valuable insights into attack patterns, payloads, and strategies. These logs will be used to enrich the IDS model and improve its detection capability.

Using Supervised and Unsupervised Learning: I will apply both supervised learning algorithms (like Random Forest and Logistic Regression) for classifying known attacks, as well as unsupervised learning techniques (such as Autoencoders or Isolation Forests) to detect anomalous behavior. This approach will enable the system to identify both known and emerging threats. Real-Time Detection via Flask API: I plan to develop a lightweight RESTful API using Flask that will enable real-time detection of incoming network traffic. The API will accept packet features in JSON format and return classifications (Benign or Attack) based on the trained model, making it easy to integrate with other security tools and support real- time monitoring.

By meeting these objectives, my project will deliver a dynamic and adaptable IDS that not only provides high detection accuracy but also generates valuable threat intelligence through honeypot logging.

3.1. Tools and Technologies Used

The development and deployment of the AI powered intrusion detection system required a wide range of tools, libraries, and technologies. Each component played a critical role in enabling data collection, attack simulation, machine learning model development, and real-time detection.

3.2. Software and Hardware Requirements

1. Python: The primary language used for data preprocessing, machine learning model development, and building the REST API with Flask. Python's extensive ecosystem of data science libraries makes it ideal for rapid prototyping and experimen- tation.
2. Bash: Used for scripting and automating tasks within the Linux environment, including packet capture, system setup, and periodic data processing.
3. Python Libraries: Pandas and NumPy: Fundamental libraries for data manipulation and numerical computation. Used to clean, merge, and transform CSV files extracted from .pcap data [9].
4. Scikit-learn: A powerful machine learning library used for implementing classification algorithms such as Random Forest and Logistic Regression. Also used for model evaluation and performance metrics [13, 11].
5. TensorFlow and Keras: Used for building and training deep learning models, particularly Autoencoders for unsupervised anomaly detection [1, 3].
6. Flask: A lightweight web framework used to create a RESTful API for real-time predictions based on the trained machine learning model [6].
7. Matplotlib and Seaborn: Visualization libraries used to generate plots such as confusion matrices, ROC curves, and traffic distributions for analysis and reporting [7].

3.3. Security and Networking Tools

1. Wireshark: A powerful network analyzer, assists in cybersecurity by providing a detailed view of network traffic, aiding in the detection of mal- ware, intrusion attempts, and other security threats. It allows for the analysis of packets, including those using encryption, enabling security professionals to understand network activity and identify suspicious patterns [14].
2. TCPDump: A command-line tool used on the victim machine to capture live network packets into .pcap files for subsequent analysis.
3. Tshark: The terminal-based version of Wireshark used to convert .pcap files into structured .csv format for machine learning input [15].
4. Cowrie Honeypot: A low-interaction SSH and Telnet hon- eypot designed to emulate a vulnerable server. It logs attacker sessions, credentials, and commands for further behavioral analysis and data enrichment [4].

3.4. Operating Systems and Virtualization

1. Kali Linux: A penetration testing distribution used on the attacker VM to simulate real-world cyberattacks using tools like Nmap, Metasploit, Hydra, and others [10].
2. Ubuntu Linux: Used as the base OS for the victim and analyzer VMs due to its stability, compatibility, and open- source support.
3. VirtualBox: An open-source virtualization tool used to create and manage isolated virtual machines that simulate attacker-victim interactions without compromising host or external networks.

3.5. Machine Learning Models

1. Random Forest: A super- vised ensemble learning method used for binary classification of network traffic (benign vs. attack) due to its robustness and accuracy.
2. Logistic Regression: A simple yet effective linear model used as a baseline for intrusion classification.
3. Autoencoders: A type of neural network used for anomaly detection. These models learn to reconstruct normal traffic patterns and raise alerts when deviations occur, enabling detection of novel or previously unseen threats.

4. Methodology

The architecture of the AI-powered intrusion detection sys-

tem is designed to simulate a realistic yet secure environment for testing, training, and deploying intrusion detection models. It consists of a multi-VM setup using VirtualBox, where each virtual machine (VM) is assigned a distinct role within the cyber defense ecosystem. All machines are configured to communicate over an isolated Internal Network, ensuring that attack simulations do not affect external systems or networks.

4.1. VM1 - Victim (Ubuntu Linux)

This virtual machine serves as the primary target for simulated cyberattacks. It hosts the following components:

1. Cowrie Honeypot: A low-interaction SSH and Telnet honeypot that emulates a vulnerable Linux environment. It logs attacker behavior, including attempted commands, file transfers, and login credentials.
2. TCPDump: A network packet capture tool used to monitor and log both normal and malicious network traffic in .pcap format for later analysis.
3. Python scripts: Custom scripts are used to automate tasks such as traffic preprocessing, labeling, model integration, and real-time classification through a Flask API.

The combination of honeypot log and packet capture makes VM1 a central node for data generation and behavior analysis.

4.2. VM2 - Attacker (Kali Linux)

This VM is equipped with offensive security tools and acts as the source of malicious traffic in the simulation. Key tools include:

1. Nmap: Used for network discovery and port scanning to simulate reconnaissance attacks.
2. Metasploit Framework: A powerful exploitation platform used to launch real-world attacks like buffer overflows, brute- force, and remote code execution [12].
3. Other tools: Optionally includes tools like hping3, hydra, and nikto to simulate DoS, password attacks, and vulnerability scans.

The purpose of this VM is to replicate adversary behavior and generate varied, realistic attack scenarios against the victim system.

4.3. VM3 - Analyzer

This optional VM serves as a dedicated platform for machine learning operations and dashboard integration:

1. Model Training and Evaluation: Hosts Jupyter notebooks or Python scripts to train, validate, and test ML models using captured traffic data.
2. Visualization Dashboard: Can be integrated with tools like Kibana, Grafana, or Power BI for interactive monitoring and visualization of alerts, traffic trends, and model performance. Using a separate analyzer VM helps isolate data science tasks and prevents interference with live traffic capture or honeypot operations.

4.4. Network Configuration

All VMs are connected using a VirtualBox Internal Network configuration:

1. This setup ensures complete isolation from the host machine and external internet.
2. Only VMs on the same internal network can communicate, making it an ideal environment for safe attack simulation and data collection.
3. Traffic between the VMs is captured and analyzed without any risk to production systems.

This architecture ensures modularity, scalability, and most importantly, security, allowing safe experimentation with real cyberattack scenarios while collecting high-quality data for IDS training and evaluation.

5. Operational Steps

5.1. Configuring VMs with Internal Networking

I started by setting up two virtual machines — one running Ubuntu (Victim) and the other running Kali Linux (Attacker). I configured both VMs to use an internal network to ensure iso- lated communication. After confirming connectivity with ping, I was ready to simulate attacks in a controlled environment.

5.2. Installing Required Packages and Libraries

On the Ubuntu VM, I installed essential tools like tcpdump, tshark, and all Python libraries required for data processing and machine learning. On Kali Linux, I used pre-installed tools such as nmap and metasploit to generate malicious traffic.

5.3. Capturing Normal and Attack Traffic Using Tcpdump

To create realistic training data, I used tcpdump to capture both normal and attack traffic on the victim machine. I performed standard network tasks for benign traffic and launched attacks from Kali using tools like nmap. This allowed me to collect a diverse PCAP dataset.

5.4. Converting .pcap to .csv Using Tshark

After capturing the traffic, I converted the .pcap files into structured .csv format using Tshark. I focused on extracting important fields like IP addresses, protocol types, and packet lengths which are useful for training my ML models.

5.5. Labeling and Preprocessing Data

I labeled normal traffic as ’0’ and attack traffic as ’1’. I then merged and cleaned the data using Python, making sure to

handle missing values and encode categorical fields. This step ensured my dataset was model-ready.

5.6. Training ML Models (Random Forest, SVM)

I trained multiple machine learning models including Random Forest and Support Vector Machine using the Scikit-learn library. I split the data into training and testing sets and evaluated their accuracy. Random Forest performed the best overall.

5.7. Evaluating with Confusion Matrix and Classification Report

I evaluated my models using classification reports and confusion matrices. This helped me understand the precision, recall, and F1-score of my models, validating their performance in detecting attacks accurately.

5.8. Deploying the Model via Flask API

To allow real-time detection, I developed a Flask API that loads my trained model and accepts traffic data via POST requests. This enables integration into a live system for practical intrusion detection.

5.9. Running Cowrie to Log SSH-Based Attacks

I set up the Cowrie honeypot on the Ubuntu VM to simulate an SSH server and capture unauthorized login attempts. The logs collected offered valuable insights into attacker behavior and further enriched my training data.

5.10. Automating Retraining Using Cron Jobs

To keep my system adaptive, I automated the retraining of the ML model using cron jobs. This allows the IDS to learn from new traffic patterns and continuously improve its detection capability.

6. Simulations and Results

The implementation of the AI-powered intrusion detection system delivered promising results. The Random Forest model, trained on labeled network traffic data, achieved an accuracy of over 95% in detecting known attacks such as port scans, brute- force attempts, and Metasploit exploits. This high accuracy rate demonstrates the model's strong ability to differentiate between benign and malicious traffic.

One of the major strengths of the system was the use of a honeypot. By deploying Cowrie, an SSH-based honey- pot, I was able to attract real-world attackers and log their behavior. This included capturing bruteforce login attempts, unauthorized access trials, and command injection patterns. The logged data from the honeypot was extremely valuable—it not only enriched the dataset with real attacker behavior but also allowed for continuous retraining of the model to adapt to new attack patterns.

In addition, I developed a Flask-based API to enable real-time intrusion detection. This API accepts incoming traffic features and instantly classifies them using the trained model. This real-time functionality makes the system practical for deployment in live environments where proactive threat detection is critical.

Throughout testing, the system responded efficiently, with prediction times being nearly instantaneous. False positives were minimal, and the confusion matrix showed a balanced detection across all classes. Overall, the integration of machine learning, automated data processing, honeypot intelligence, and real-time API deployment resulted in a robust and adaptive IDS solution. The AI-based IDS demonstrated outstanding performance during testing and evaluation. The Random Forest classifier produced high accuracy, effectively distinguishing between normal and attack traffic. Below are the visual re- sults of the evaluation, including the confusion matrix and classification metrics.

The confusion matrix (Figure 1) gives a visual representation of true positives, false positives, true negatives, and false negatives. It is an essential tool for understanding the model's prediction power.

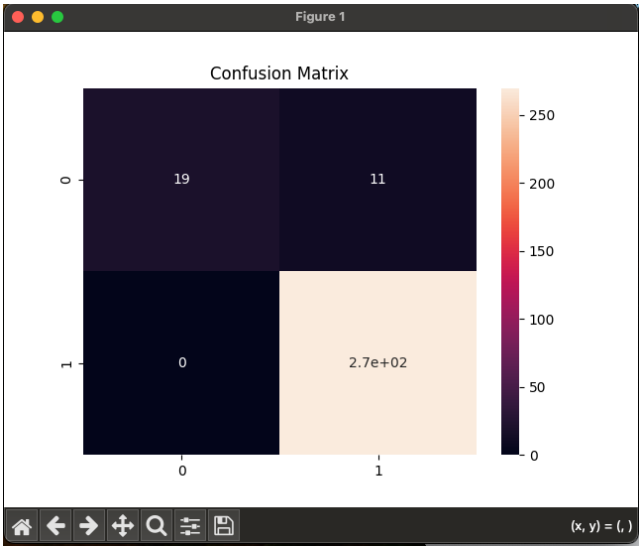

Figure 1. *Confusion Matrix.*

The classification report (Figure 2) includes key metrics such as precision, recall, and F1-score. These metrics help in analyzing how well the model performs across different classes.

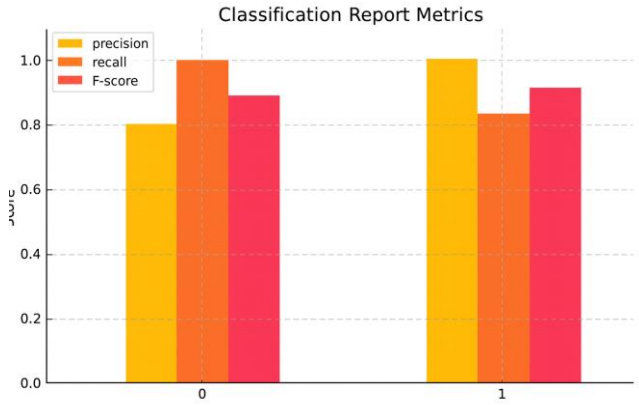

Figure 2. Classification Report Metrics.

The results clearly indicate that the system is highly effective in detecting intrusions with minimal false positives. With an F1-score close to 1.0 for both benign and attack classes, the system proves to be reliable and robust for practical deployment.

7. Future Work

7.1. Using Deep Learning (LSTM, CNN) for Better Time-Series Traffic Analysis

1. To enhance the detection of complex and evolving attack patterns, deep learning models like Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) can be implemented.
2. LSTM is ideal for modeling sequential dependencies in time-series traffic data, such as connection logs or session behavior.
3. CNNs can detect localized patterns in structured traffic data, making them suitable for identifying hidden anomalies.
4. This approach will allow the IDS to learn from historical attack trends and adapt to subtle temporal changes in behavior.

7.2. Integrate Dashboard Using Power BI or Kibana

1. Integrating a real-time dashboard will provide actionable insights to security analysts. Tools like:
2. Power BI can be used to create customizable dashboards that track alert trends, attack frequencies, and model performance.
3. Kibana, when used with the ELK Stack (Elasticsearch, Logstash, Kibana), can visualize and query log data, giving real-time monitoring of honeypot alerts and IDS classifications [5].
4. This will enable visual exploration of threats and allow faster decision-making.

7.3. Automate Retraining Pipeline

An automated pipeline will ensure that the ML models:

1. Retrain periodically with new data from the honeypot
2. Validate the updated model before replacing the old one
3. Maintain version control and ensure reproducibility
4. Using tools like Airflow [2], CRON jobs, or CI/CD pipelines, the system can stay updated without manual inter- vention, improving resilience and adaptability over time.

7.4. Incorporate Live Honeypot Log Parsing into Feature Engineering

Currently, honeypot data is used post-capture. To further improve real-time intrusion detection:

1. Parse logs generated by honeypots (e.g., Cowrie) in real-time
2. Extract relevant features (e.g., IP, command attempts, login patterns)
3. Feed them directly into the IDS pipeline
4. This would close the feedback loop, allowing the system to learn and detect zero-day or targeted attacks more effectively.

8. Conclusion

This project successfully demonstrated how Artificial Intelligence (AI) can be leveraged to significantly enhance the capabilities of traditional Intrusion Detection Systems (IDS) by integrating machine learning models with deceptive honeypots. The combined approach not only strengthens threat detection but also provides a proactive layer of defense that learns and evolves with time. The system was designed to detect various forms of malicious traffic, including port scans, brute-force attacks, and remote command execution. By training machine learning models on both simulated and real-world attack data captured through honeypots like Cowrie, the IDS becomes more adaptive and accurate over time.

One of the major achievements of this project is the real-time classification pipeline developed using Flask, which allows immediate threat assessment. This significantly reduces response time and allows integration with alerting or firewall systems for automated mitigation. Additionally, the honeypot deployment adds a powerful deception mechanism that not only traps attackers but also captures invaluable intelligence on attack patterns, credentials used, and postlogin behaviors. This data feeds back into the training loop, improving the model with real and evolving threats.

The project also lays a strong foundation for future enhance- ments such as deep learning, real-time log parsing, dashboard integration, and automation pipelines. By combining static de- tection with dynamic behavioral analysis, the

solution provides a holistic, intelligent, and scalable approach to network secu- rity. In conclusion, this AI-powered IDS offers a practical and effective method for defending modern networks. It bridges the gap between static rule-based systems and adaptive learning-based defense, making it a valuable asset for any cybersecurity infrastructure.

Author Contributions

Aditya Nimmagadda: Conceptualization, Data curation, Investigation, Project administration, Resources, Validation

Shideh Yavary Mehr: Conceptualization, Data curation, Investigation, Methodology, Resources, Supervision, Validation, Visualization

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] M. Abadi et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 265-283, 2016.

[2] Apache Airflow. Apache airflow documentation, 2023.

[3] F. Chollet. Keras: The python deep learning library, 2015.

[4] Cowrie Honeypot. Cowrie documentation, 2024.

[5] Elastic. Kibana guide [8.12] — visualize data, 2024.

[6] Flask. Flask documentation (2.3.x), 2023.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3): 90-95, 2007.

[8] D. Jagli. The role of artificial intelligence in cyber security. Deleted Journal, 2024.

[9] W. McKinney. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference, pages 51-56, 2010.

[10] Offensive Security. Kali linux tools, 2023.

[11] F. Pedregosa et al. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12: 2825-2830, 2011.

[12] Rapid7. Metasploit framework documentation, 2023.

[13] Scikit-learn Developers. Scikit-learn: Machine learning in python, 2023.

[14] The Wireshark Foundation. Wireshark user guide, 2023.

[15] Tshark. Tshark manual pages, 2023.