CSIT998 Professional Capstone Project

Zero-Day Network Attack Prevention Using AI

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Supervisor: Dr. Nan Li

Group: 2313

Team Members:

Abundio Edgar Cahuana Quispe	(Student ID: 7378671)
Nabila Ahmad	(Student ID: 7717593)
Neha K C	(Student ID: 7570879)
Md Aminul Islam	(Student ID: 7695044)
Tashrif Islam	(Student ID: 7736150)
Ariel Abarabar	(Student ID: 6936520)

May 17, 2024

Abstract

AI-Based Zero-Day Attack Prevention Systems (ZDAPS)

The rise of zero-day attacks poses a significant threat to modern cybersecurity, exploiting undiscovered vulnerabilities in software and hardware systems. Traditional security measures, relying heavily on signature-based detection, struggle to keep pace with these emerging threats. AI-based Zero-Day Attack Prevention Systems (ZDAPS) offer a robust solution by leveraging advanced machine learning algorithms for real-time detection, behavioural analysis, and adaptive learning. This project aims to develop a comprehensive AI framework that proactively identifies and mitigates zero-day network attacks.
Our approach combines supervised and unsupervised learning methods to enhance the detection capabilities of Intrusion Detection Systems (IDS). Key algorithms include Long Short-Term Memory (LSTM) networks, Random Forest, Logistic Regression, and semi-supervised models such as one-class support vector machines. These models are trained and validated using diverse datasets that encompass standard network activities and known cyberattack patterns. Through rigorous training, preprocessing techniques, and continuous feedback from cybersecurity experts, our models achieve high accuracy in distinguishing between normal and malicious network traffic.

The integration of AI in cybersecurity frameworks significantly reduces false positives and adapts to evolving threats by incorporating real-time threat intelligence feeds. This dynamic adaptability ensures that ZDAPS remains effective against the latest attack vectors, providing a proactive defence mechanism. Our system not only detects potential zero-day intrusions but also offers swift alerts to cybersecurity personnel, enabling immediate countermeasures.

Future research will focus on enhancing feature selection methods, exploring additional neural network variants, and expanding the knowledge base with emerging threat patterns. This continuous improvement cycle is crucial for maintaining the effectiveness of AI-driven cybersecurity solutions. As the landscape of cyber threats evolves, the predictive and detection capabilities of AI become indispensable, forming a critical component of contemporary cybersecurity strategies.

Table of Contents

1 Introduction

In today's digital age, zero-day attacks are among the most formidable challenges to cybersecurity, exploiting vulnerabilities that are unknown or unpatched. These attacks highlight the significant limitations of traditional cybersecurity responses, which typically involve significant delays in detection, response, and mitigation. In response, this project focuses on leveraging Artificial Intelligence (AI) to develop a more resilient and adaptive defence mechanism against such threats. Unlike conventional systems, AI-driven systems utilize a combination of anomaly detection and deep learning models to develop an understanding of normal versus anomalous network behaviour without prior exposure to specific attack signatures.

These systems integrate sophisticated algorithms capable of processing and analysing large volumes of data at high speeds, enabling them to detect subtle anomalies that could indicate a potential zero-day attack. This proactive detection is bolstered by machine learning models such as Neural Network, Random Forest and LSTM networks, which not only predict potential breaches based on historical data but also learn from ongoing activities, thus continually enhancing their predictive accuracy. The integration of continuous, real-time monitoring and the ability to adapt to new data inputs significantly reduces the window of vulnerability that zero-day attacks exploit [3].

Furthermore, these AI systems are designed to seamlessly integrate with existing security frameworks, enhancing them with capabilities to perform advanced threat analysis and response. This integration facilitates a multi-layered defence strategy, combining the rapid response times of AI with the depth of traditional security methods to form a comprehensive, adaptive security posture. By doing so, AI not only functions as a detection tool but also as a strategic asset that enhances overall cybersecurity resilience [10] [18] [26].

As cyber threats continue to evolve in complexity and scale, the importance of developing advanced defence mechanisms cannot be overstated. Traditional security measures, such as signature-based Intrusion Prevention Systems (IPS), are inherently reactive and often fall short in the face of novel, sophisticated attacks. These systems rely on pre-existing knowledge of attack patterns, leaving networks vulnerable to new and unknown threats that have not yet been catalogued. AI-driven security systems, on the other hand, offer a paradigm shift by providing a proactive approach to cybersecurity. By leveraging machine learning and deep learning algorithms, these systems can detect anomalous behaviour without prior knowledge of specific attack signatures. This ability to identify potential threats based on deviations from normal network behaviour is crucial for defending against zero-day attacks, which exploit unknown vulnerabilities[13] [14] [20].

The implementation of AI in cybersecurity also addresses the need for real-time threat detection and response. Traditional systems often suffer from delays in recognizing and responding to attacks, allowing adversaries to inflict significant damage before countermeasures can be deployed. AI systems, with their capacity for rapid data processing and analysis, can detect and respond to threats almost instantaneously. This immediate response capability is vital for minimizing the impact of zero-day attacks and protecting critical network infrastructure[21] [22] [23].

In addition to enhancing detection and response times, AI systems can continually learn and adapt to new threats. Machine learning models, such as Neural Networks, Random Forests, and LSTM networks, can be trained on historical data to recognize patterns indicative of malicious activity. These models can then be updated with new data, enabling them to adapt to evolving attack techniques and maintain their effectiveness over time[25].

Another significant advantage of AI-driven security systems is their ability to integrate with existing cybersecurity frameworks. This integration ensures that organizations can leverage their current security investments while augmenting them with advanced AI capabilities. By combining the strengths of traditional security methods with the agility and precision of AI, organizations can develop a comprehensive defence strategy that is both robust and adaptive[26].

Moreover, the deployment of AI in cybersecurity must be approached with a focus on minimizing false positives. High rates of false positives can overwhelm security teams and lead to alert fatigue, reducing the overall effectiveness of the defence system. Advanced machine learning techniques can help to reduce false positives by accurately distinguishing between normal and malicious behaviour, ensuring that security personnel can focus their efforts on genuine threats[24] [27].

The development and deployment of AI-based zero-day attack prevention systems also highlight the importance of collaboration and ethical considerations in cybersecurity. Sharing threat intelligence and defence strategies across the industry can enhance the collective ability to defend against zero-day attacks. Additionally, ensuring that AI systems are designed and implemented in an ethical manner, with respect for privacy and data protection, is crucial for maintaining trust and legitimacy in their use [18] [18].

In conclusion, the integration of AI into cybersecurity represents a significant advancement in the fight against zero-day attacks. By leveraging the capabilities of machine learning and deep learning, AI-driven systems provide a proactive, adaptive, and resilient defence mechanism that can detect and mitigate threats in real-time. The continuous learning and adaptation of these systems, combined with their seamless integration into existing security frameworks, offer a comprehensive solution to the evolving challenges of cybersecurity. Through industry collaboration and a focus on ethical AI use, we can ensure that these advanced defence mechanisms are both effective and responsible in safeguarding our digital infrastructure[19].

Aims and Objectives

The primary aim of this project is to design and implement an AI-based system capable of detecting and preventing zero-day network attacks. The system is intended to provide a proactive defence mechanism that is both adaptive and resilient, leveraging the capabilities of machine learning and deep learning algorithms to identify and mitigate potential threats in real-time.

Specific Objectives

1. Develop a Comprehensive AI Framework:
   - Create an AI-driven framework that combines supervised and unsupervised learning algorithms to detect anomalous network behaviour indicative of zero-day attacks.
   - Utilize deep learning models, including Neural Networks, Random Forests, and LSTM networks, to enhance the accuracy and reliability of threat detection.
2. Implement Real-Time Monitoring and Detection:
   - Integrate continuous, real-time monitoring capabilities to ensure that the system can dynamically adapt to new and evolving threats.
   - Develop algorithms that can process and analyse large volumes of network data at high speeds, enabling the detection of subtle anomalies that may indicate an attack.
3. Enhance Integration with Existing Security Frameworks:
   - Ensure that the AI system can seamlessly integrate with current security infrastructure, enhancing existing methods with advanced threat analysis and response capabilities.
   - Facilitate a multi-layered defence strategy that combines the strengths of AI with traditional cybersecurity measures to provide a robust and adaptive security posture.
4. Minimize False Positives and Improve Predictive Accuracy:
   - Focus on minimizing false positives by employing advanced machine learning techniques that can differentiate between normal and malicious network behaviour with high precision.
   - Continuously refine and update the machine learning models based on new data inputs and threat intelligence to maintain high predictive accuracy.
5. Promote Industry Collaboration and Ethical AI Use:
   - Advocate for industry-wide collaboration to share threat intelligence and defence strategies, thereby enhancing the overall effectiveness of zero-day attack prevention.
   - Ensure that the AI system is designed and deployed in a manner that respects privacy and ethical standards, particularly when processing personal or sensitive data.

2 Literature Review

Zero-day attacks can be categorized into Anomaly-based, Graph-based, and AI-based methods, as illustrated in the figure below. Detailed descriptions of these methods are provided in the following sections.

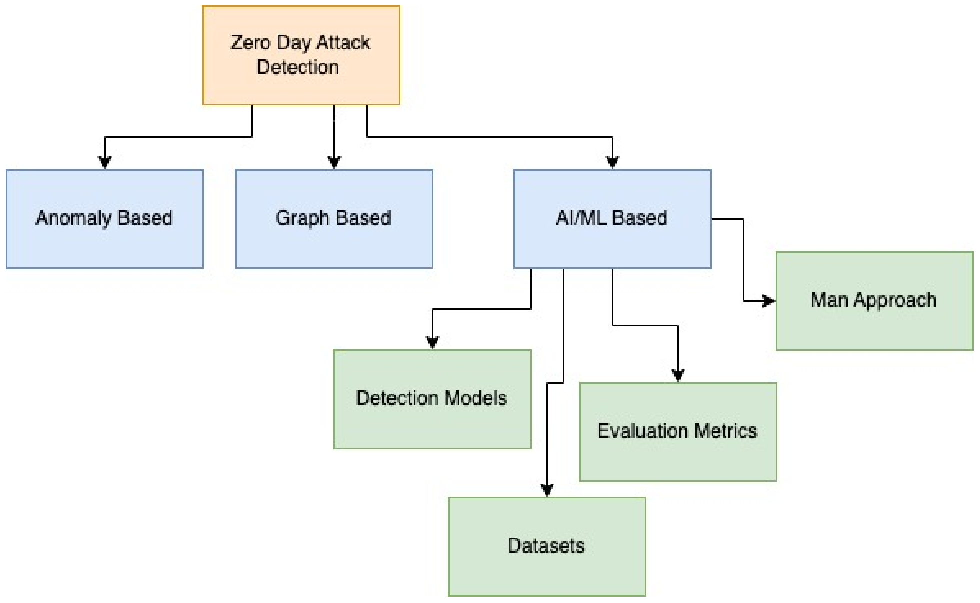


Figure 1: Zero Day Attack Detection Approaches

2.1 Anomaly Based

The anomaly-based detection method utilizes regression techniques, such as logistic regression, to analyze chaotic invariants like entropy and correlation dimensions, which are nonlinear and intrinsic characteristics. These features are crucial for generating significant attributes in machine learning algorithms, particularly effective for detecting anomalies related to buffer overflows and packet contents among other vulnerabilities.

Duessel et al. [14] introduced a new data representation called CN-Gram at the application layer[14]. This method combines syntactic and sequential attributes of payload data into a unified feature space. Post-training, the algorithm assesses syntax-level attributes and byte message mappings to identify anomalous behaviors.

Moon et al. [13] developed a host-based detection system focused on secure human-centric computation. The system defines 39 features across seven categories (e.g., file system, process, registry, thread) to determine if a host process is malicious, using a decision tree based on a feature vector from a dedicated database.

Moustafa et al. suggested an Outlier Dirichlet Mixture (ODM) model as a fog detection system in [21], and the authors of [7]presented a framework for detecting zero-day polymorphic worms using behavioral, anomaly, and signature-based methods. This framework operates across three layers: analysis, detection, and resources, utilizing both benign and malicious traffic data.

Khan et al. [20] introduced a multilevel anomaly detection model for Supervisory Control and Data Acquisition Systems (SCADA), relying on communication patterns between devices to identify anomalies. This model utilizes dimensionality reduction for data preprocessing and a Bloom filter to create a signature database, blending content-level detection with instance-based learning to enhance detection capabilities.

2.2 Graph Based

Leman et al. [5] conducted an extensive survey of the latest methods for detecting anomalies in graph-structured data. They classified these techniques based on whether they are automatic or supervised, and the type of graphs they handle—be it static, dynamic, attributed, or plain. Their evaluation focused on the effectiveness, scalability, and reliability of each method. They also stressed the importance of understanding the root causes of anomalies and discussed various methods that can trace these anomalies back to their origins. The survey highlighted real-world applications in sectors such as finance, online auctions, and social networks, and concluded with a discussion on current challenges in the field.

Graph-based models have demonstrated improved efficacy in attack detection compared to traditional behavioral and anomaly-based methods. This is evidenced in studies like those referenced in [16], [4], and [24]. These models utilize directed graphs to visualize network communications and identify potential attack paths. For example, Yichao et al. [24] discussed an anomaly detector that employs network attack probabilities, introducing a stochastic behavioral model to differentiate between attacker behavior and normal host conditions.

Wang et al. [10] developed DaMask, a dynamic architecture that updates detection models using Bayesian network inferences to adapt to new DDoS attack patterns. This approach allows the system to stay current with evolving threats by continuously updating its understanding of attack patterns. Singh et al. [23] described a layered graph-based architecture for zero-day attack detection. Their system includes components like a risk analyzer and path generator, utilizing a centralized server and database for data analysis. They implemented an algorithm called AttackRank to assess potential exploitation paths, providing a structured approach to identifying and mitigating risks.

Bayoglu et al. [4] created a framework using a Conjunction of Combinational Motifs (CCM) to classify diverse worm signatures. This method analyzes graph vertices as invariant parts of worms, enabling the system to accurately identify and classify various worm signatures.

[12] proposed a compact graph planning method for efficient attack path discovery, involving closure calculation, graph construction, and path extraction.

2.3 AI/ML Based

This subsection introduces various AI models, datasets, and evaluation metrics used in detecting zero-day attacks.

2.3.1 Detection Models

Different AI-based models are employed to enhance zero-day attack detection:
Decision Trees:A non-parametric method that uses a tree-like model of decisions and their possible consequences. It includes root, intermediate, and leaf nodes that help in making predictions by learning simple decision rules from data features.[25]

Random Forests:An ensemble learning method for regression and classification that creates a forest of decision trees, usually trained via the bagging method. It is well known for its effectiveness and simplicity across various applications and we have picked this for our project as well [3].

Multilayer Perceptron(MLP):MLPs are used for approximating functions and handling non-linearly separable data, applying layers of processing units for tasks like classification and prediction [1]. One of the main challenges in network security is sorting network traffic into two categories: benign (harmless) and malicious (harmful). MLPs are particularly effective for this task because they can learn from labeled data and make predictions based on the patterns they recognize. When we train an MLP using a dataset of network traffic, each data instance is labeled as either normal or an attack. Through this training process, the MLP learns to spot even the subtle signs of malicious activity. This ability to predict potential threats is crucial for the real-time detection and prevention of zero-day attacks, which are attacks that exploit previously unknown vulnerabilities in software or hardware.

Long Short-Term Memory (LSTM): An advanced RNN (Recurrent Neural Networks) architecture that is well-suited for sequential data analysis, applicable in fields like speech recognition and anomaly detection[30]. LSTMs excel at tasks involving sequential data due to their robust memory and learning capabilities, making them invaluable for accurate and context-aware processing in various fields.

Hybrid Multilevel Anomaly Detection-IDS (HML-IDS): A hybrid model combining anomaly detection techniques and IDS for robust zero-day attack detection in systems like SCADA [12].

Convolutional Neural Networks (CNNs):Convolutional Neural Networks (CNNs) are advanced multilayer neural networks, building on the concept of feed-forward artificial neural networks (ANNs) [9]. The term "convolution" refers to a linear operation between matrices, which inspired the name of CNNs. Key components of CNNs include convolutional, non-linearity, pooling, and fully connected layers. While the pooling and non-linearity layers are parameter-free, the convolutional and fully connected layers contain parameters. CNNs excel in machine learning tasks, particularly in handling image data. They have produced remarkable outcomes in fields such as large-scale image classification with datasets like ImageNet, as well as in computer vision, natural language processing (NLP) [22], drug discovery [15], and anomaly detection [6] [8]. CNNs are capable of extracting higher-resolution features and transforming them into a more complex features as resolution diminishes. CNNs are deep neural networks known for their performance in image and video recognition tasks, utilizing layers of convolutions across input data.

Reinforcement Learning: Reinforcement learning (RL) is often described as learning from feedback, as it continuously receives responses about the accuracy of its predictions; it's a subset of machine learning. Unlike methods that provide the algorithm with explicit solutions, RL requires the algorithm to explore and experiment with different strategies until it identifies the correct approach [11].

Deep Neural Network (DNN): A Deep Neural Network (DNN) is a type of feed-forward network that includes an input layer, multiple hidden layers, and an output layer. It is characterized by its several hidden layers that sit between the input and output. In a DNN, each layer is made up of numerous nodes, each hierarchically connected to every node in the next layer. The process begins with the input layer receiving features, which are then processed through the hidden layers. Finally, the output layer delivers predicted values. Within each node of the hidden layers, a weighted sum of the inputs is calculated and passed through a predefined activation function, transforming these sums into the final outputs of the model. [29]

2.3.2 Datasets

The Cloud Intrusion Detection Dataset (CIDD)analyzes network and host audit data linked to user IPs and audit timestamps. It includes 35 days of Unix Solaris audit data for training, along with TCP data dumps and labeled attacks for IDS training. For testing purposes, 10 days of Solaris audits and TCP data dumps are provided [27].

The Information Security and Object Technology (ISOT) dataset encompasses multiple botnets, including data captured from the Storm and Waledac Botnets, along with non-malicious data from Traffic Lab Ericsson. Combined with data from Lawrence Berkeley National Lab (LBNL), this dataset totals 1,675,424 network traffic instances [2].

Generated by simulating SSH, HTTP, and SMTP traffic, the **Information Security Centre of Excellence (ISCX)** dataset offers a variety of intrusion scenarios. The ISCX-URL-2016 subset includes 35,300 benign URLs, 12,000 spam, 10,000 phishing, over 11,500 malware, and 45,540 defacement URLs, curated by the Canadian Institute for Cybersecurity. [26]

The NSL KDD dataset, an enhancement of the earlier KDD'99 dataset, eliminates redundancies in its training and testing sets. It includes data categories like Denial of Service (DoS), User to Root (U2R), Remote to Local Attack (R2L), and probing attacks, totaling about 5 million connection records derived from DARPA's network traffic over seven weeks [66]. This dataset is available in both JSON and CSV formats. [35]

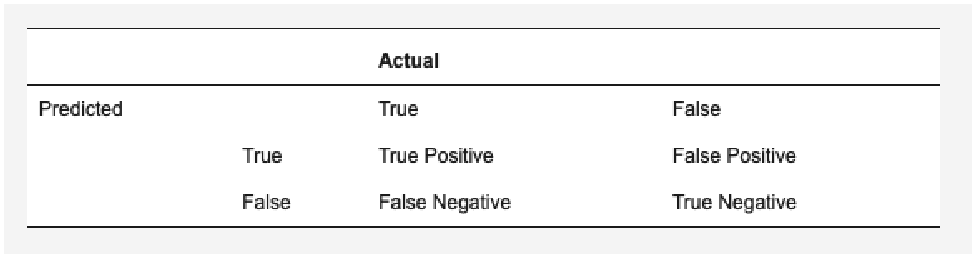
The Safety Pilot Model Deployment dataset, enhanced by researcher Wyk, includes data on vehicle speed, GPS speed, and acceleration, with added simulations for potential cyberattacks and faults affecting vehicular services. [26]

The Enron Spam dataset, compiled by V. Metsis, I. Androutsopoulos, and G. Paliouras, consists of 17,171 spam emails and 16,545 non-spam emails. This publicly available dataset includes emails categorized by subject, message content, arrival date, and spam status [35].

In the Kaggle, The Microsoft Malware Classification Challenge was a competition that utilized datasets containing various malware samples. The datasets included samples from malware such as Ramnit (1539 training data, 145 test data), Lollipop (2459 training data, 234 test data), Obfuscator.ACY (1221 training data, 132 test data), and Gatak (1011 training data, 106 test data). Additionally, researchers used Generative Adversarial Networks (GANs) to create additional fake malware samples for the competition. [21]

2.3.3 Evaluation Metrics

Before discussing the metrics, let's first see an overview of the confusion matrix. Typically arranged in a 2x2 matrix format, the confusion matrix is a tool used to display the performance of an algorithm. The actual outcomes that are expected are listed vertically, and the outcomes predicted by the algorithm are listed horizontally. This layout, a 2x2 confusion matrix, is presented in a table format.



		Actual	
Predicted		True	False
	True	True Positive	False Positive
	False	False Negative	True Negative

Figure 2: A 2X2 Confusion Matrix

True Positive (TP) is the total number of positive instances correctly identified as positive.

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

True Negative (TN) is the number of negative instances identified as negative.

$$TNR = \frac{TN}{TN + FP} \tag{2}$$

False Positive (FP) is defined as the number of negative instances classified or predicted as positive.

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

False Negative (FN) is the number of positive instances classified or predicted as negative.

$$FNR = \frac{FN}{FN + TP} \tag{4}$$

Accuracy is the ratio between the number of correct predictions and a total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

Precision is defined as the ratio between TPs combined with several TPs and FPs. It is the percentage of correctly identified positives out of all the results that were said to be positive either correctly or not.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall the ratio between TPs combined to several TPs and FNs. It is the percentage of correctly identified positives out of all actual positives, either correctly or not.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

F1-score : It takes both false negatives and false positives into consideration, and is the harmonic mean of recall and precision. It performs well on imbalanced datasets.

$$F1 - score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{8}$$

2.4 Recent Paper Reviews

The Performance of Machine and Deep Learning Classifiers in Detecting Zero-Day Vulnerabilities: This study explores the challenge of detecting zero-day attacks and vulnerabilities, a critical task for network administrators who need to ensure high accuracy for robust defense mechanisms. In an ideal situation, a system would be able to identify zero-day malware with complete accuracy, eliminating concerns over mistakenly labeling benign files as malicious or allowing harmful code to run unchecked. This research assesses various machine learning algorithms to determine their efficacy in identifying zero-day malware. After analyzing 34 machine and deep learning classifiers, it was discovered that the Random Forest classifier achieved the highest accuracy. The paper raises several questions about the effectiveness of machine and deep learning techniques in detecting zero-day malware while maintaining zero false positives and negatives. Key terms include zero-day vulnerability and machine learning [17].

HTTP/2 Zero-Day vulnerability results in record-breaking DDoS attacks: The Cloudflare blog post discusses a novel zero-day vulnerability named "HTTP/2 Rapid Reset," which exploits a flaw in the HTTP/2 protocol to launch large-scale DDoS attacks. This vulnerability uses the protocol's stream cancellation feature to flood targets with requests, significantly disrupting server operations. Despite using only a moderately sized botnet, the attacks achieved unprecedented request rates, indicating a severe threat level. Cloudflare, in response, developed specialized technology to mitigate these attacks and has engaged in extensive discussions with industry peers to enhance overall Internet security [34].

Zero-day DDoS Attack Detection: This study focuses on the detection of zero-day Distributed Denial of Service (DDoS) attacks, utilizing network traffic analysis before it enters a private network. It highlights the use of modern feature extraction techniques combined with neural networks to differentiate between benign and malicious network packets. This paper underscores the challenges and the necessity of evolving detection methods to keep up with novel attack vectors. [32]

A Combined Deep CNN: LSTM with a Random Forest Approach for Breast Cancer Diagnosis: This paper proposes a novel approach combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks with Random Forest for diagnosing breast cancer, showcasing a method that can also be applicable in security domains for feature extraction and classification [31].
Intrusion detection systems using long short-term memory (LSTM):In this paper, the effectiveness of using Long-Short Term Memory (LSTM) models for intrusion detection systems was explored by comparing various LSTM-based models. Principal Component Analysis (PCA) and Mutual Information were utilized as dimensionality reduction techniques to prepare data for analysis. The study found that LSTM models are adept at learning from training data to accurately distinguish between normal network traffic and cyber attacks. The models were implemented using the Keras library and TensorFlow on the Google Colab platform. Notably, models that used PCA with just two components achieved high performance in both binary and multiclass classification scenarios, with accuracy rates of 99.44 percent and 99.39 percent, respectively. The findings highlight that using fewer features can simplify the training process and reduce resource consumption without sacrificing model performance. The paper concludes by suggesting future research directions, including testing various LSTM variants and exploring additional neural network and feature selection methods [28].

Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches): The paper "Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches" demonstrates that semi-supervised machine learning is effective for identifying zero-day attacks when significant features are optimally chosen. Specifically, it highlights the superiority of one-class support vector machines, which achieved a Matthews correlation coefficient of 74 percent and an F1 score of 85 percent in detecting zero-day network attacks. The research emphasizes the role of network traffic analysis and feature selection in improving intrusion detection system performance [33].

3 Methodology

Addressing zero-day network attacks with artificial intelligence requires a structured and thorough approach. Our project utilized three diverse datasets to develop and test AI models for anomaly detection, focusing on network activities and cyberattack patterns, especially those related to past zero-day vulnerabilities.

Initially, we gathered a comprehensive set of data that included both standard network activities and known cyberattack patterns. This dataset served as the primary training material for our AI. The data was then processed to extract discernible features, setting the stage for the AI to detect unusual, potentially harmful patterns. For our AI's backbone, we selected models well-suited for anomaly detection, such as Random Forest, Logistic Regression, Neural Networks, and Long Short-Term Memory (LSTM) networks. These models were chosen for their robustness and effectiveness in identifying atypical sequences and data anomalies in network traffic. The chosen models underwent rigorous training using the aforementioned datasets.

The training process involved splitting the data into training and testing sets, ensuring the models were evaluated on unseen data. Various techniques were employed to handle class imbalances and preprocess the data, including the use of LabelEncoder for categorical data and RandomUnderSampler for balancing classes. Each model's performance was assessed using accuracy, confusion matrix, and classification reports. The results demonstrated that models like Random Forest and Logistic Regression, along with Neural Networks, particularly LSTM, were highly effective for anomaly detection in network traffic.

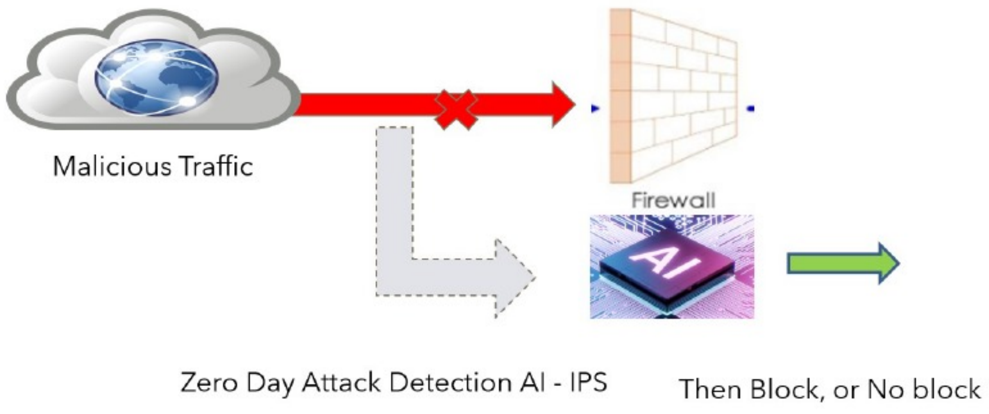


Figure 3: AI-based Zero-Day Attack Prevention Systems (ZDAPS)

This figure illustrates the comprehensive framework for the AI-based Zero-Day Network Attack Prevention System. The system combines anomaly detection with behavioral analysis to identify and prevent potential zero-day attacks. The figure highlights the key components, including data collection, feature extraction, model training, and real-time monitoring.

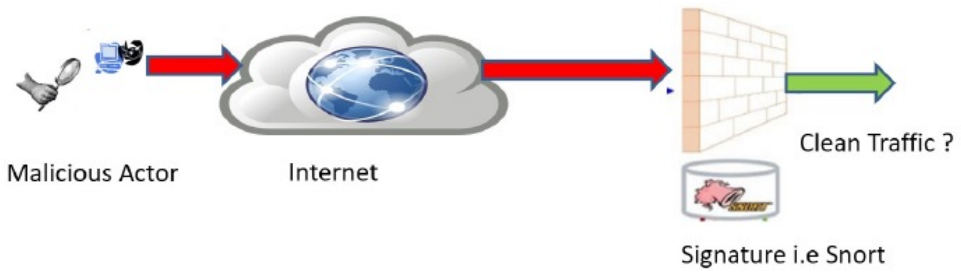


Figure 4: Snort Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)

This figure provides a detailed overview of the Snort IDS/IPS architecture. It demonstrates how Snort captures, decodes, and analyzes network packets to detect malicious activities based on predefined rules. The figure also shows the alert generation and logging mechanisms employed by Snort.

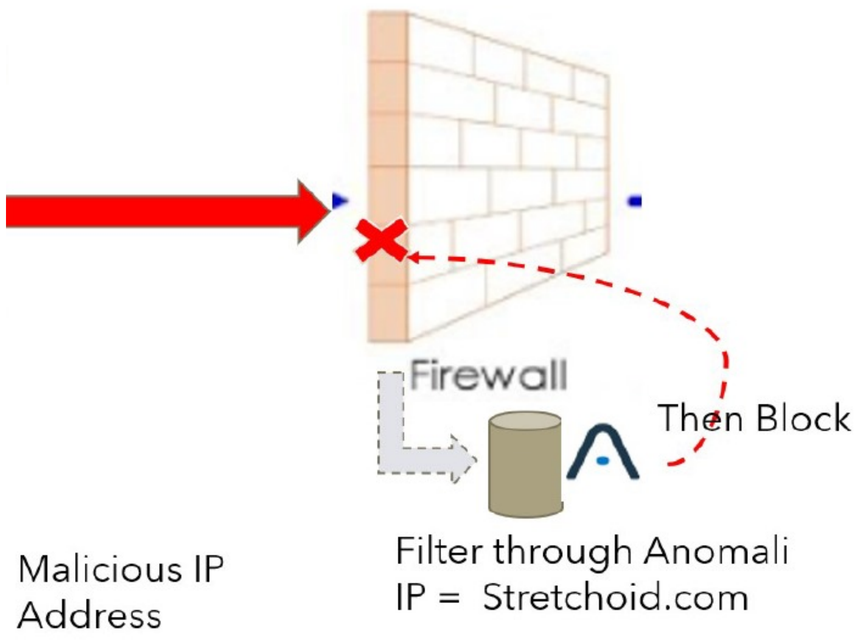


Figure 5: AI-based Zero-Day Attack Prevention Systems (ZDAPS) Block

This block diagram breaks down the components of the AI-based Zero-Day Attack Prevention Systems (ZDAPS). It includes the data preprocessing module, machine learning models, threat intelligence integration, and the automated response system. The diagram emphasizes the continuous learning and adaptation capabilities of ZDAPS.

3.1 Use case diagram description:

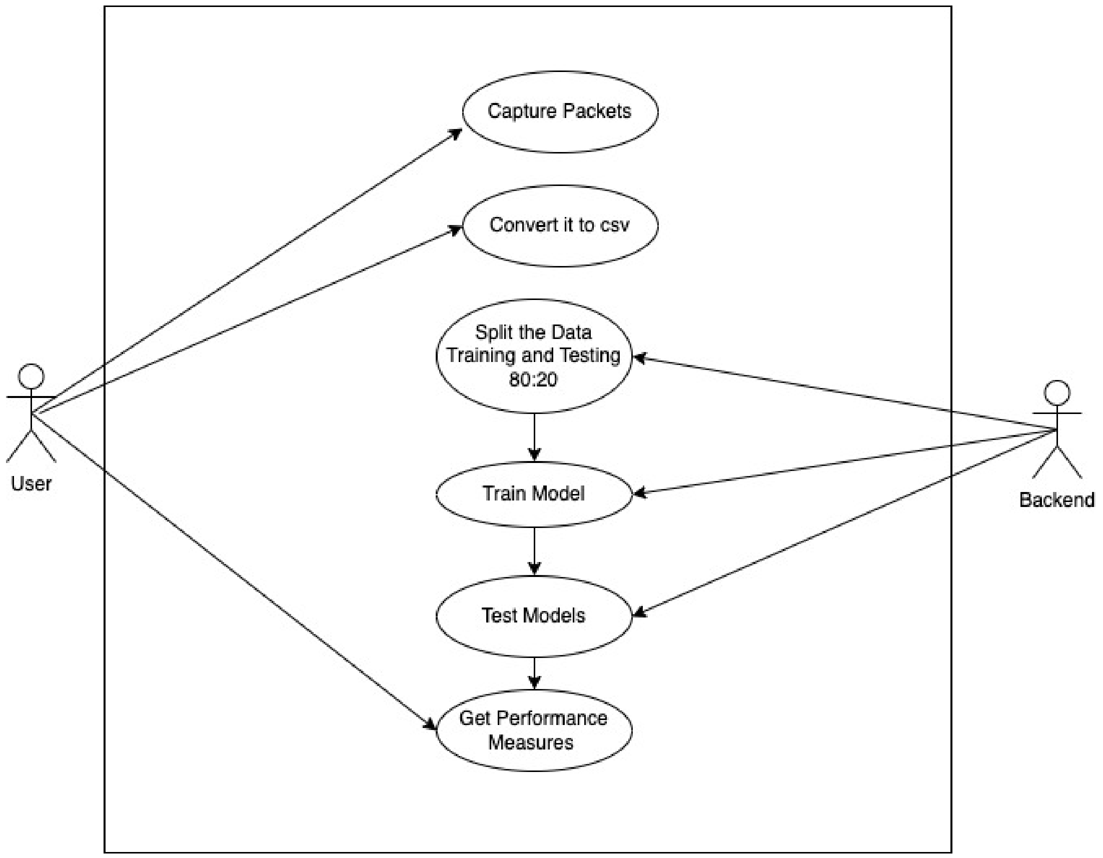


Figure 6: Use Case Diagram

The use case diagram presented illustrates the comprehensive workflow for detecting zero-day attacks using AI/ML techniques. The diagram involves two primary actors: the User and the Backend. The User, likely a cybersecurity professional or network administrator, initiates the process by capturing network packets. This step is critical as it involves intercepting and logging network traffic data, which forms the basis for anomaly detection. The captured packets are then converted into a CSV format, a structured and machine-readable format that facilitates further data manipulation and analysis.

Once the data is in CSV format, it is split into training and testing sets, typically in an 80:20 ratio. This step is essential for model validation, ensuring that the AI models can generalize from the training data to new, unseen data. The Backend system plays a crucial role here, handling the data processing tasks and maintaining the integrity of the data split. The training data is used to train various AI/ML models. These models, which could include Random Forest, Logistic Regression, MLP classifiers, and LSTM networks, learn to identify patterns and anomalies that could indicate zero-day attacks.

After the models are trained, they are tested using the testing data to evaluate their performance. The Backend system again takes the lead in this phase, running the models against the testing data and generating performance metrics such as accuracy, precision, recall, and F1-score. These performance measures are then viewed by the User (in the Dashboard) to assess the effectiveness of the models. The user can compare differet models to check which one performs better compared to toher. This feedback loop ensures that the models are robust and reliable before deployment. The entire process, from packet capture to performance evaluation, is designed to create an effective and adaptive system for detecting and responding to zero-day network attacks in real-time.

3.2 Supervised Models Selection Justification

Supervised learning models were chosen for their ability to leverage labeled datasets to learn the distinguishing features of normal and anomalous network traffic. This approach allows the models to generalize well to new, unseen data, which is crucial for effective anomaly detection in network security.

Random Forest: Chosen for its robustness, ability to handle high-dimensional data, and resistance to overfitting.
Logistic Regression: Selected for its simplicity, interpretability, and effectiveness in binary classification.
MLP Classifier: Chosen for its ability to model complex, non-linear relationships in data. This makes it highly effective for tasks such as classification, where it can learn intricate patterns and dependencies within the dataset, thereby improving the accuracy and reliability of predictions.
LSTM: Chosen for its effectiveness in handling sequential data, making it suitable for detecting patterns in time-series network traffic data.

The methodology employed in this project involved using diverse datasets to train and validate several supervised learning models. The process included thorough data preparation, addressing class imbalance, and utilizing robust training and evaluation techniques. The results demonstrate that models like Random Forest and Logistic Regression, along with Neural Networks, particularly LSTM, are highly effective for anomaly detection in network traffic, providing a strong foundation for developing an AI-based Intrusion Prevention System (IPS).

3.3 Importance of Data Preprocessing

Data preprocessing is a crucial step in the machine learning pipeline, serving as the foundation for building effective and reliable AI models. The primary objective of data preprocessing is to transform raw data into a format that can be easily and effectively utilized by machine learning algorithms. Raw data, in its initial form, often contains inconsistencies, noise, and missing values that can negatively impact the performance of AI models. By addressing these issues through various preprocessing techniques, we ensure that the data is clean, consistent, and structured, which significantly enhances the model's learning capabilities and overall performance.

One of the main reasons for data preprocessing is to handle missing values. Real-world data is often incomplete, with some observations missing certain attributes. Machine learning models rely on complete data to make accurate predictions, and missing values can lead to biased or incorrect outcomes. Techniques such as imputation (replacing missing values with mean, median, or mode) or removal of incomplete records are commonly employed to handle this issue. By addressing missing values, we ensure that the dataset is complete and the models can learn from the full range of available information.

Another critical aspect of data preprocessing is the normalization and standardization of data. Different features in a dataset may have varying scales and units, which can skew the model's performance. For instance, in a dataset with attributes like age and income, the income values can be significantly larger than age values. This disparity can cause the model to give undue importance to certain features over others. Normalization (scaling data to a range of [0, 1]) and standardization (scaling data to have a mean of 0 and a standard deviation of 1) are techniques used to bring all features to a comparable scale, ensuring that the model treats each feature equally during training.

Data preprocessing also involves encoding categorical variables, which are common in many datasets. Machine learning algorithms typically require numerical input, but datasets often contain categorical data, such as gender or product categories. Encoding techniques, such as one-hot encoding or label encoding, convert these categorical variables into numerical formats that can be easily processed by machine learning algorithms. One-hot encoding creates binary columns for each category, while label encoding assigns a unique integer to each category. Proper encoding ensures that the models can effectively interpret and utilize categorical data, leading to more accurate predictions and better overall performance.

4 Results

For AI based anomaly detection, initially started from a **demo dataset** to build the models. Which can give us more ideas how to modify the models to get a good accuracy for all the models. In total we have worked with three different types of datasets.

4.1 First dataset details:

This dataset contains information related to HTTPS requests and some associated metrics, for anomaly detection purposes. Here's a breakdown of the columns:
https_requests_count: Number of HTTPS requests made.
https_failed_requests: Number of HTTPS requests that failed.
avg_https_response_time: Average response time for HTTPS requests.
certificate_validity: Boolean indicating whether the HTTPS certificate is considered valid (likely True for valid and False for invalid).
unusual_https_port_usage: Indicates if an unusual HTTPS port was used (potentially a binary indicator).
is_malware: Boolean indicating whether the activity is considered malware.
The dataset comprises 26 entries without any missing values. It has a mix of numerical and Boolean features which is used for detecting anomalies activity in HTTPS traffic.

Based on the first dataset, the coding procedures has been done this way:

4.1.1 Code Description and Preparation

Procedures Data loading:
To begin, the code imports the libraries needed for data manipulation, preprocessing, and model assessment, including pandas, numpy, and scikit-learn modules. It uses the read_csv() method in Pandas to load the data from a CSV file. The function receives the file path as an argument.

Data pre-processing:
Data Investigation: By outputting details about the dataset, including the number of rows and columns, column names, data types, and summary statistics, the code carries out exploratory data analysis (EDA).

Managing types of data:
To identify categorical columns, use the select_dtypes() method. Scikit-learn's LabelEncoder() is used to apply label encoding to convert category data to a numerical representation.

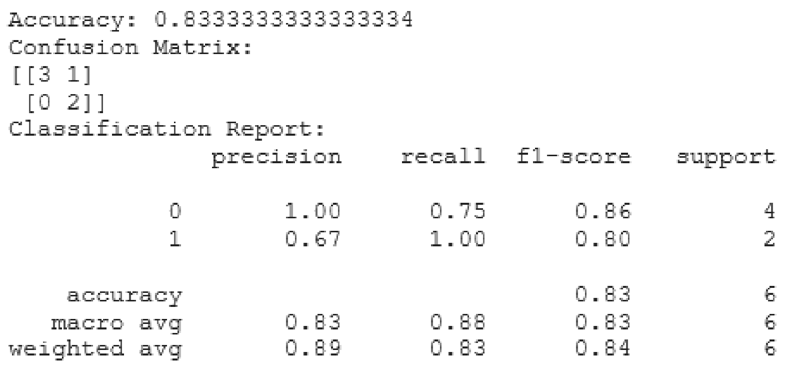
Train-Test Division:
The train_test_split() method in scikit-learn is used to divide the preprocessed data into training and testing sets. This makes it possible to evaluate the model on the unobserved testing set and train it on the training set.

4.1.2 Model Development and Assessment

The Random Forest, Logistic Regression, Neural Network , and LSTM machine learning models are the four models that the code specifies functions for training and assessing. The testing data is used to evaluate each model after it has been trained using the training data. Scikit-learn routines are used to generate accuracy, confusion matrix, and classification report metrics for the Random Forest, Logistic Regression classification models. Keras is used to create a neural network architecture for the LSTM and Neural Network model. The pre-processed data is used to train the model, and the testing set is used to assess it.

Results are shown:
Each model's output is printed to the console along with its accuracy, confusion matrix, and classification report.

For Random Forest Model:

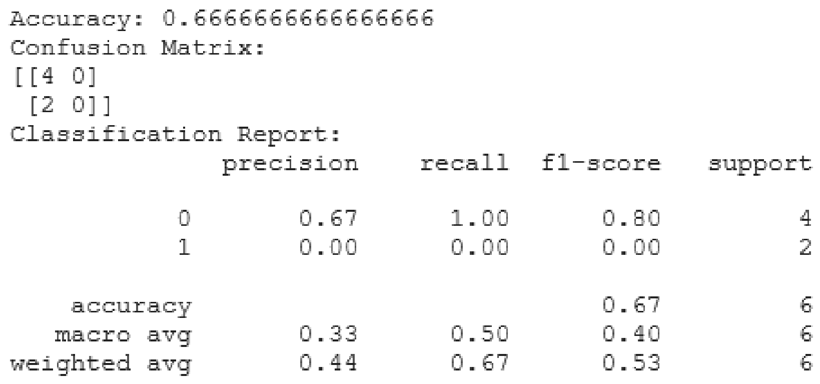


```
Accuracy: 0.8333333333333334
Confusion Matrix:
[[3 1]
 [0 2]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.75      0.86         4
           1       0.67      1.00      0.80         2

    accuracy                           0.83         6
   macro avg       0.83      0.88      0.83         6
weighted avg       0.89      0.83      0.84         6
```

For Logistic Regression Model:

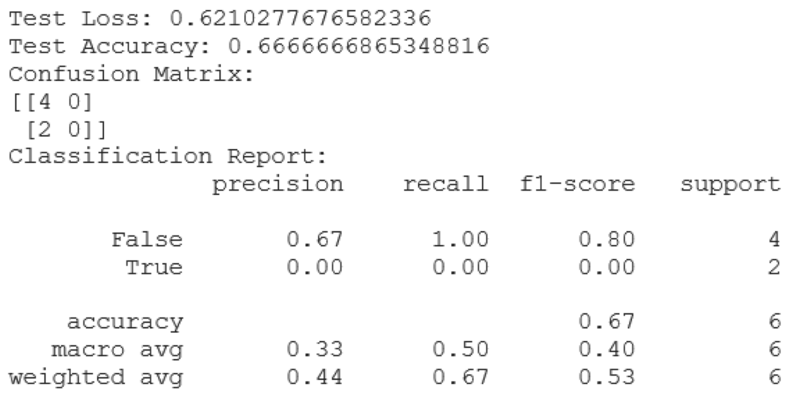


```
Accuracy: 0.6666666666666666
Confusion Matrix:
[[4 0]
 [2 0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         4
           1       0.00      0.00      0.00         2

    accuracy                           0.67         6
   macro avg       0.33      0.50      0.40         6
weighted avg       0.44      0.67      0.53         6
```

For LSTM Model:

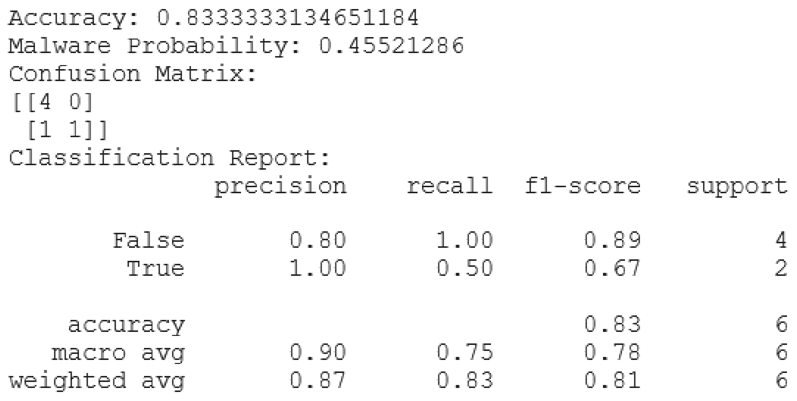


```
Test Loss: 0.6210277676582336
Test Accuracy: 0.6666666865348816
Confusion Matrix:
[[4 0]
 [2 0]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      1.00      0.80         4
        True       0.00      0.00      0.00         2

    accuracy                           0.67         6
   macro avg       0.33      0.50      0.40         6
weighted avg       0.44      0.67      0.53         6
```

For Neural Network Model:



```
Accuracy: 0.8333333134651184
Malware Probability: 0.45521286
Confusion Matrix:
[[4 0]
 [1 1]]
Classification Report:
              precision    recall  f1-score   support

       False       0.80      1.00      0.89         4
        True       1.00      0.50      0.67         2

    accuracy                           0.83         6
   macro avg       0.90      0.75      0.78         6
weighted avg       0.87      0.83      0.81         6
```

In summary, out of all the other models Random Forest Algorithm works best for detecting anomaly detection.

4.2 Second Dataset Details

Second dataset has been taken from an open-source website **Kaggle**. In this dataset has two CSV files one is for training the models and another is testing the models. The only difference between this dataset is testing dataset doesn't have "class" column. This dataset is much larger and more detailed compared to the previous one. It contains 25,192 entries with 42 columns[19].

Here's a breakdown of the columns:
duration: Length of time (in seconds) of the connection.
protocol_type: Type of protocol used in the connection (e.g., TCP, UDP).
service: Network service on the destination machine.
flag: Status of the connection (e.g., normal, error).
src_bytes: Number of bytes sent from the source to the destination.
dst_bytes: Number of bytes sent from the destination to the source.
land: Binary indicator if the connection is from/to the same host/port.
wrong_fragment: Number of wrong fragments in the connection.
urgent: Number of urgent packets in the connection.
hot: Number of "hot" indicators in the connection.
num_failed_logins: Number of failed login attempts.
logged_in: Binary indicator if the user is logged in.
num_compromised: Number of compromised conditions.
root_shell: Binary indicator if root shell is obtained.
su_attempted: Binary indicator if su root command attempted.
num_root: Number of root accesses.
num_file_creations: Number of file creation operations.
num_shells: Number of shell prompts.
num_access_files: Number of operations on access control files.
num_outbound_cmds: Number of outbound commands.
is_host_login: Binary indicator if the login belongs to the "hot" list.
is_guest_login: Binary indicator if the login is a "guest" login.
count: Number of connections to the same host as the current connection.
srv_count: Number of connections to the same service as the current connection.
Various rates and ratios of connection attributes (serror_rate, srv_serror_rate, rerror_rate, etc.).
class: Class label indicating the type of network connection (e.g., normal or anomaly).
This dataset is highly detailed and a good example for intrusion detection or network security analysis, with various features describing network traffic characteristics and potential attacks.

4.2.1 Code Description and Preparation

Procedures Data loading:
To begin, the code imports the libraries needed for data manipulation, preprocessing, and model assessment, including pandas, numpy, and scikit-learn modules. It uses the read_csv() method in Pandas to load the data from a CSV file. The function receives the file path as an argument.
Data pre-processing:
Data Investigation: By outputting details about the dataset, including the number of rows and columns, column names, data types, and summary statistics, the code carries out exploratory data analysis (EDA).
Managing types of data:
To identify categorical columns, use the select_dtypes() method. Scikit-learn's LabelEncoder() is used to apply label encoding to convert category data to a numerical representation.
Correcting unbalanced training classes:
The RandomUnderSampler from the imbalanced-learn package is used by the code to handle class imbalance. To maintain dataset balance, the majority class is undersampled.
Train-Test Division:
The train_test_split() method in scikit-learn is used to divide the preprocessed data into training and testing sets.

This makes it possible to evaluate the model on the unobserved testing set and train it on the training set.

4.2.2 Model Development and Assessment

The Random Forest, Logistic Regression and LSTM machine learning models are the four models that the code specifies functions for training and assessing. The testing data is used to evaluate each model after it has been trained using the training data. Scikit-learn routines are used to generate accuracy, confusion matrix, and classification report metrics for the Random Forest, Logistic Regression classification models. Keras is used to create a architecture for the LSTM model. The pre-processed data is used to train the model, and the testing set is used to assess it.

Results are shown:
Each model's output is printed to the console along with its accuracy, confusion matrix, and classification report.

In the below diagram ”1” means ”anomaly” and ”0” means ”normal” class. It can be understand that both of the classes are almost equal.

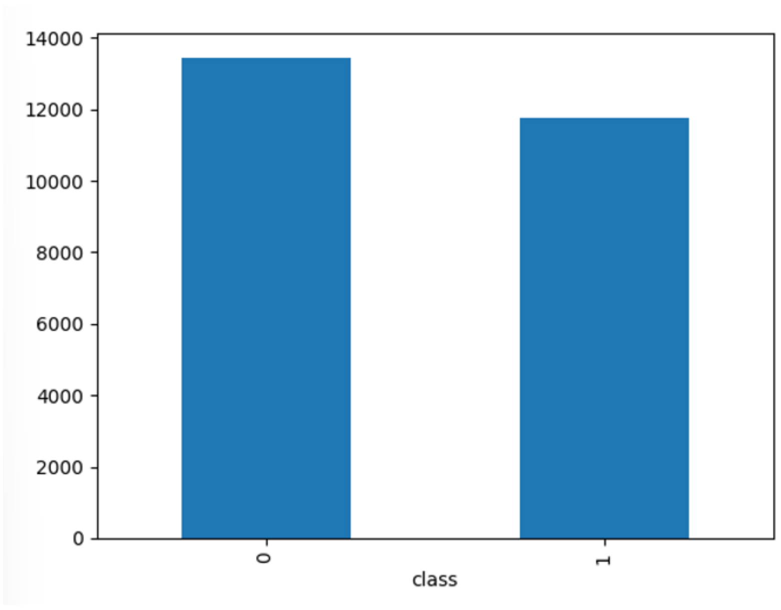
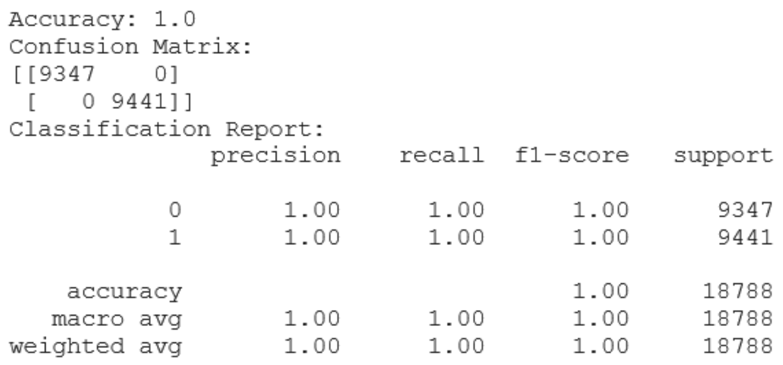


Figure 7: graph to show the ”class” data of the dataset

This above figure shows the “class” column data distribution from the dataset. Here, “0” means “normal” class and “1” means “anomaly” class. Normal class data are more than the anomaly class data.

For Random Forest Model:
For training the model:

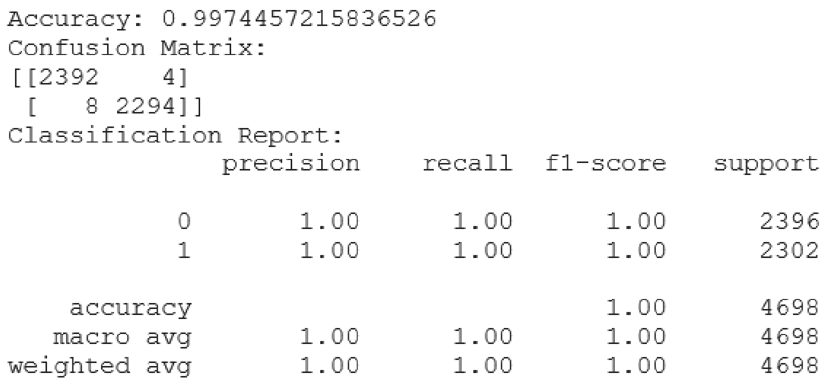


```
Accuracy: 1.0
Confusion Matrix:
[[9347    0]
 [   0 9441]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9347
           1       1.00      1.00      1.00      9441

    accuracy                           1.00     18788
   macro avg       1.00      1.00      1.00     18788
weighted avg       1.00      1.00      1.00     18788
```

For testing the model:

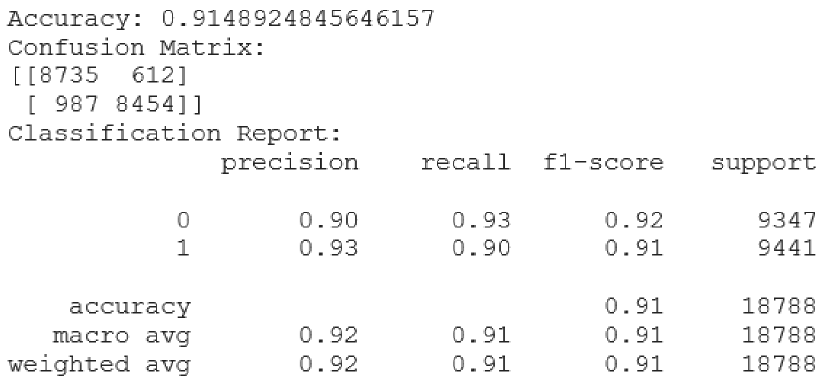


```
Accuracy: 0.9974457215836526
Confusion Matrix:
[[2392    4]
 [   8 2294]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2396
           1       1.00      1.00      1.00      2302

    accuracy                           1.00      4698
   macro avg       1.00      1.00      1.00      4698
weighted avg       1.00      1.00      1.00      4698
```

For Logistic Regression Model:
For training the model:

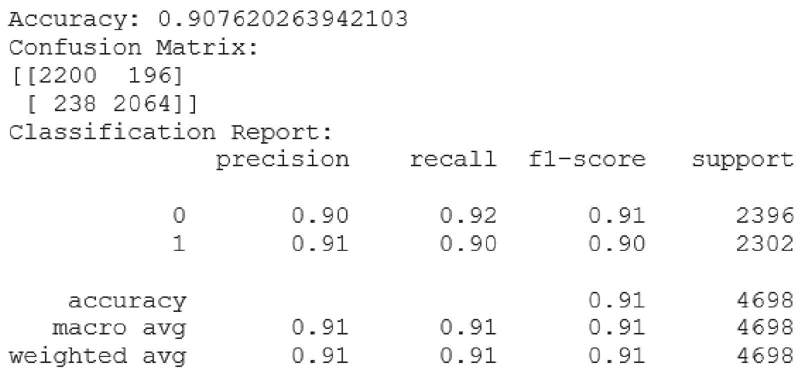


```
Accuracy: 0.9148924845646157
Confusion Matrix:
[[8735  612]
 [ 987 8454]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.93      0.92      9347
           1       0.93      0.90      0.91      9441

    accuracy                           0.91     18788
   macro avg       0.92      0.91      0.91     18788
weighted avg       0.92      0.91      0.91     18788
```

For testing the model:

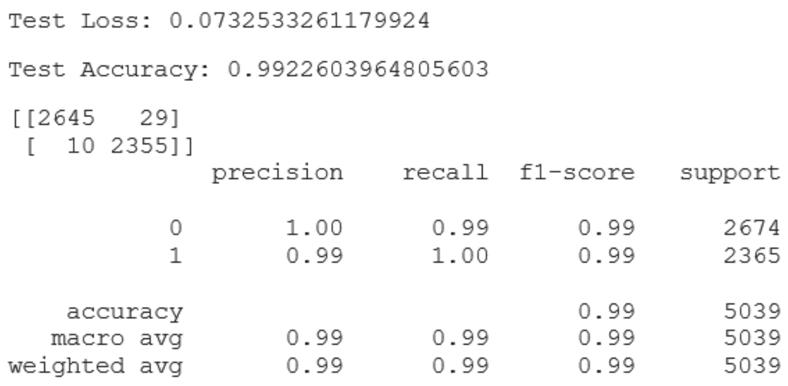


```
Accuracy: 0.907620263942103
Confusion Matrix:
[[2200  196]
 [ 238 2064]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      2396
           1       0.91      0.90      0.90      2302

    accuracy                           0.91      4698
   macro avg       0.91      0.91      0.91      4698
weighted avg       0.91      0.91      0.91      4698
```

For LSTM model:



```
Test Loss: 0.0732533261179924

Test Accuracy: 0.9922603964805603

[[2645   29]
 [  10 2355]]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      2674
           1       0.99      1.00      0.99      2365

    accuracy                           0.99      5039
   macro avg       0.99      0.99      0.99      5039
weighted avg       0.99      0.99      0.99      5039
```

In summary, out of all the other models Random Forest Algorithm and LSTM works best for detecting anomaly detection.
After working with these datasets. All the models are already given good accuracy results. For further development process capturing real time data packets and processing it was the main task for the application to work successfully.

Data capture process in real time:

This script uses the Scapy library to capture network packets and extract information, focusing on HTTPS, HTTP, and ICMP protocols. It imports necessary libraries like Scapy and Pandas, and defines global variables to track packet-related information. The packet_callback function is the heart of the script, extracting relevant information from the packet, updating global counters for failed logins and compromised instances, extracting packet attributes like duration, source bytes, and destination bytes, and parsing raw packet information to extract key-value pairs. The data dictionary is then added to the packets list for later use in creating a DataFrame.

The script defines three functions: capture_https_packets, capture_http_packets, and capture_icmp_packets, which capture specific protocols using Scapy's sniff function with a specified filter expression. The packet_callback function is provided as the prn parameter to sniff, ensuring each captured packet is processed.
The script captures HTTPS packets using the capture_https_packets function, creates a DataFrame (df) from the collected packets, and saves it to a CSV file named "https.csv". The DataFrame contains the captured packet information, including various attributes extracted from the packets.

4.3 Final dataset details

After capturing the data and saving it in a csv file. That csv file will be the next dataset that will be used for our application. All the anomaly data in this dataset has taken from various other resources. This dataset contains network packet data with 93,862 entries and 22 columns.

Here's a breakdown of the columns:
num_failed_logins: Number of failed login attempts.
num_compromised: Number of compromised conditions.
urgent: Binary indicator if the Urgent flag is set in the TCP header.
wrong_fragment: Number of wrong fragments in the packet.
src_bytes: Number of bytes sent from the source to the destination.
dst_bytes: Number of bytes sent from the destination to the source.
ihl: Internet Header Length (IHL) of the IP header.
len: Total length of the IP packet.
flags: Flags field of the TCP header.
frag: Fragmentation offset of the IP packet.
ttl: Time to Live (TTL) field of the IP header.
proto: Protocol type (e.g., TCP, UDP) of the packet.
sport: Source port number.
dport: Destination port number.
seq: Sequence number in the TCP header.
ack: Acknowledgment number in the TCP header.
dataofs: Data offset in the TCP header.
reserved: Reserved bits in the TCP header.
window: Window size in the TCP header.
urgptr: Urgent pointer in the TCP header.
duration: Duration of the packet.
class: Class label indicating the type of network connection (e.g., normal or anomaly).

4.3.1 Code Description and Preparation

Procedures Data loading:
To begin, the code imports the libraries needed for data manipulation, preprocessing, and model assessment, including pandas, numpy, and scikit-learn modules. It uses the read_csv() method in Pandas to load the data from a CSV file. The function receives the file path as an argument.

Data pre-processing:
Data Investigation: By outputting details about the dataset, including the number of rows and columns, column names, data types, and summary statistics, the code carries out exploratory data analysis (EDA). Remove duplicates column.

Managing types of data:
To identify categorical columns, use the select_dtypes() method. Scikit-learn's LabelEncoder() is used to apply label encoding to convert category data to a numerical representation.

Correcting unbalanced training classes:
The RandomUnderSampler from the imbalanced-learn package is used by the code to handle class imbalance. To maintain dataset balance, the majority class is undersampled.

Train-Test Division:
The train_test_split() method in scikit-learn is used to divide the preprocessed data into training and testing sets. This makes it possible to evaluate the model on the unobserved testing set and train it on the training set.

4.3.2 Model Development and Assessment

The Random Forest, Logistic Regression, MLP classifier and LSTM machine learning models are the four models that the code specifies functions for training and assessing. The testing data is used to evaluate each model after it has been trained using the training data. Scikit-learn routines are used to generate accuracy, confusion matrix, and classification report metrics for the Random Forest, Logistic Regression and MLP classifier classification models. Keras is used to create a architecture for the LSTM model. The pre-processed data is used to train the model, and the testing set is used to assess it.

Results are shown:
Each model's output is printed to the console along with its accuracy, confusion matrix, and classification report.

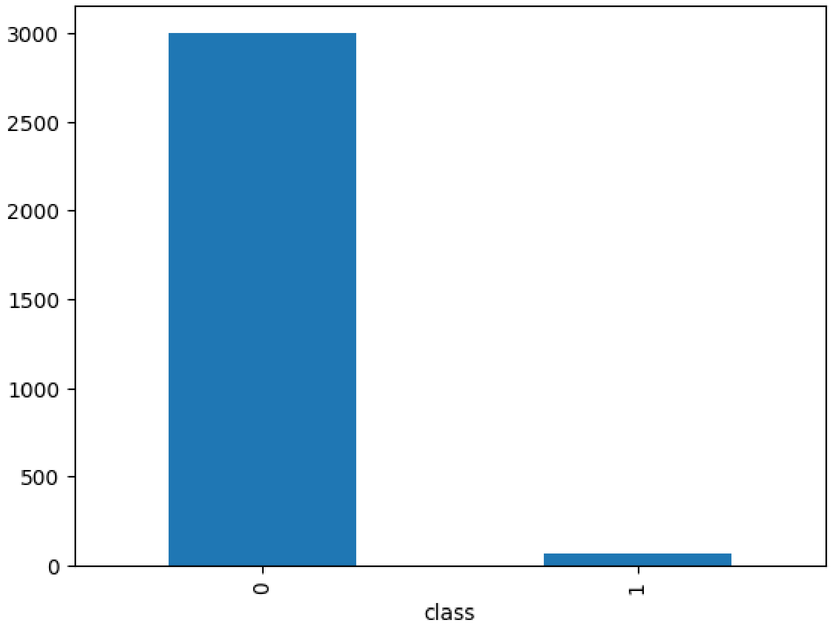
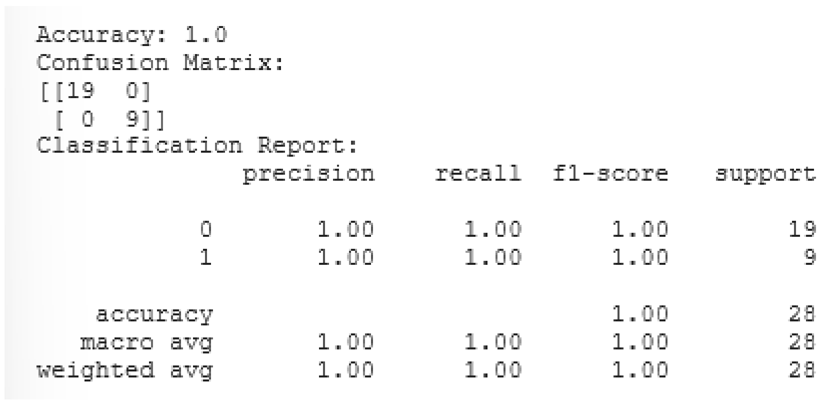


Figure 8: Graph to show the "class" data of the dataset.

This above figure shows the "class" column data distribution from the dataset. Here, "0" means "normal" class and "1" means "anomaly" class. Normal class data are more than the anomaly class data. It's showing a data imbalance. Our initial dataset has 93,862 entries but after applying duplicate data removal the ratio has become low around 3000 normal data and few anomaly data.

For Random Forest Model:

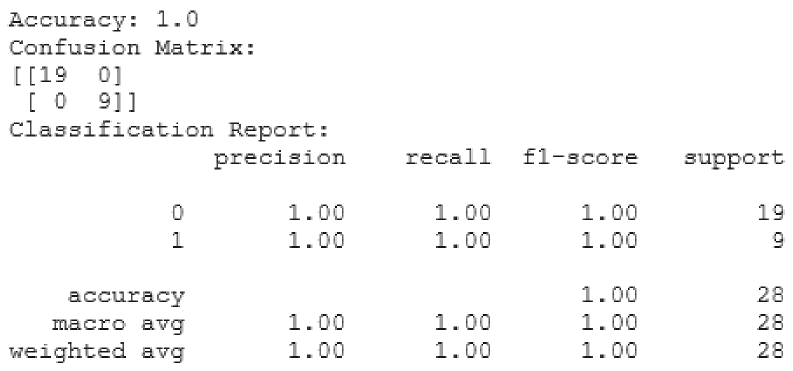


```
Accuracy: 1.0
Confusion Matrix:
[[19  0]
 [ 0  9]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00         9

    accuracy                           1.00        28
   macro avg       1.00      1.00      1.00        28
weighted avg       1.00      1.00      1.00        28
```

For Logistic Regression Model:

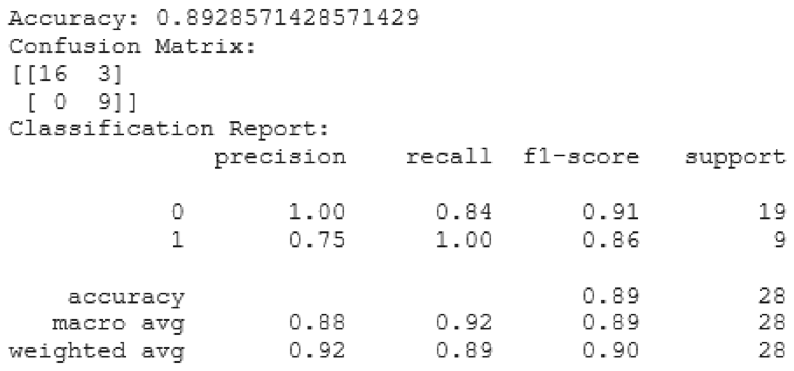


```
Accuracy: 1.0
Confusion Matrix:
[[19  0]
 [ 0  9]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00         9

    accuracy                           1.00        28
   macro avg       1.00      1.00      1.00        28
weighted avg       1.00      1.00      1.00        28
```

For LSTM model:

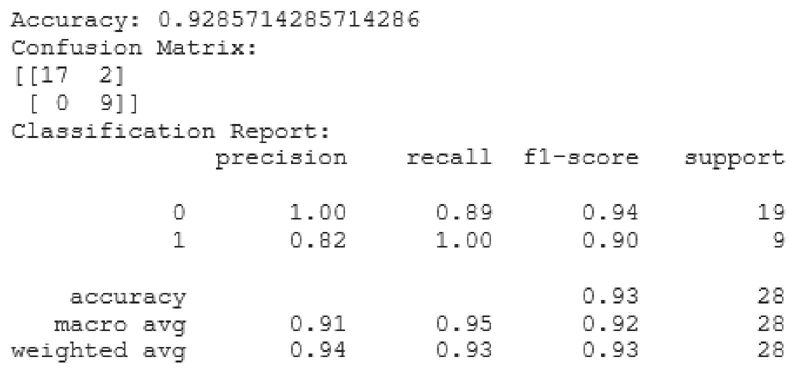


```
Accuracy: 0.8928571428571429
Confusion Matrix:
[[16  3]
 [ 0  9]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.84      0.91        19
           1       0.75      1.00      0.86         9

    accuracy                           0.89        28
   macro avg       0.88      0.92      0.89        28
weighted avg       0.92      0.89      0.90        28
```

For MLP classifier model:



```
Accuracy: 0.9285714285714286
Confusion Matrix:
[[17  2]
 [ 0  9]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        19
           1       0.82      1.00      0.90         9

    accuracy                           0.93        28
   macro avg       0.91      0.95      0.92        28
weighted avg       0.94      0.93      0.93        28
```

In summary, out of all the other model's Random Forest and Logistic Regression algorithm works best for detecting anomaly detection.

4.4 User Interface

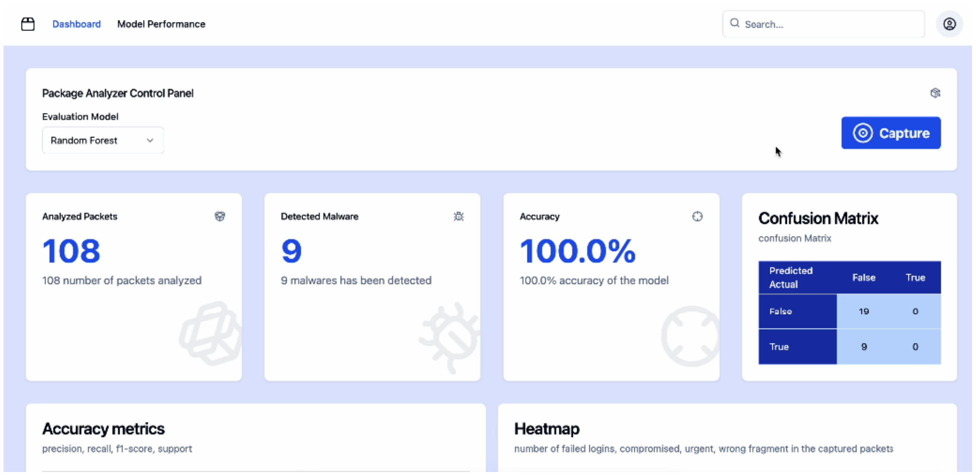
When the Capture button is clicked, the /start/ API is triggered, sending a request to the backend. Upon receiving this request, the backend invokes a Celery shared task to asynchronously capture network packets and store them in a packets list.

When the Stop button is clicked, the /stop/ API is triggered. Upon receiving this request, the backend calls the Celery shared task "stop-packet-sniffing" function, which sets the "stop-sniffing" flag to True, thereby halting the packet sniffing task. After stopping the sniffing, the backend formats the stored packets list into the desired CSV format and saves it as a CSV file.

After clicking the Stop button, the frontend sends an API request to the backend. The backend then trains the model using the newly stored CSV file data, sends the output back to the frontend, and the frontend displays the results on the dashboard.

Package	Description
celery	to handle asynchronous tasks like capturing packets
scapy	To record network packets [sniff method is provided by scapy to record network packets]
cloudamqp	free rabbitmq service to listen for celery shared tasks
rest_framework	django module to make it easier to create CRUD APIs
pandas	for data manipulation and analysis
sklearn	contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.

Table 1: Description of various packages used in the project



Dashboard Model Performance

Search...

Package Analyzer Control Panel

Evaluation Model

Random Forest

Capture

Analyzed Packets

108

108 number of packets analyzed

Detected Malware

9

9 malwares has been detected

Accuracy

100.0%

100.0% accuracy of the model

Confusion Matrix

confusion Matrix

Predicted / Actual	False	True
False	19	0
True	9	0

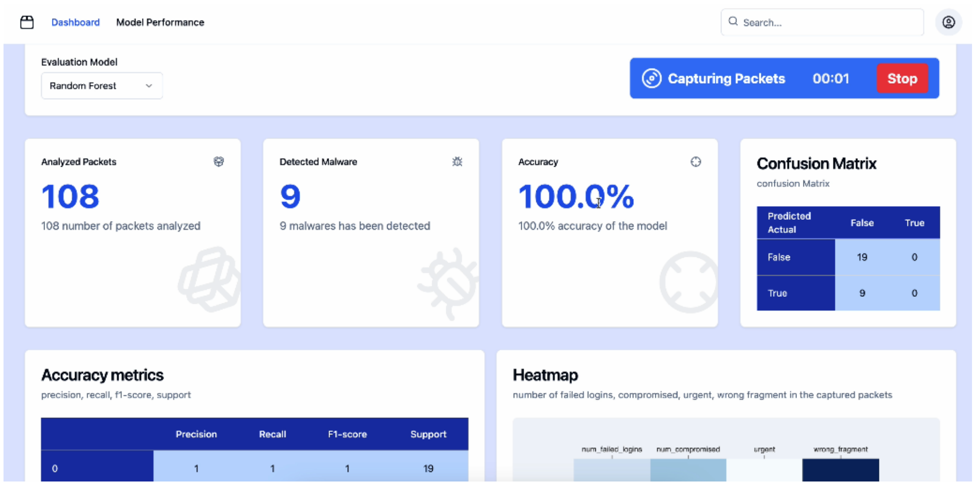
Accuracy metrics

precision, recall, f1-score, support

Heatmap

number of failed logins, compromised, urgent, wrong fragment in the captured packets

Figure 9: User Interface of Zero Day Attack Prevention Using AI



Dashboard Model Performance

Search...

Evaluation Model

Random Forest

Capturing Packets 00:01 Stop

Analyzed Packets

108

108 number of packets analyzed

Detected Malware

9

9 malwares has been detected

Accuracy

100.0%

100.0% accuracy of the model

Confusion Matrix

confusion Matrix

Predicted / Actual	False	True
False	19	0
True	9	0

Accuracy metrics

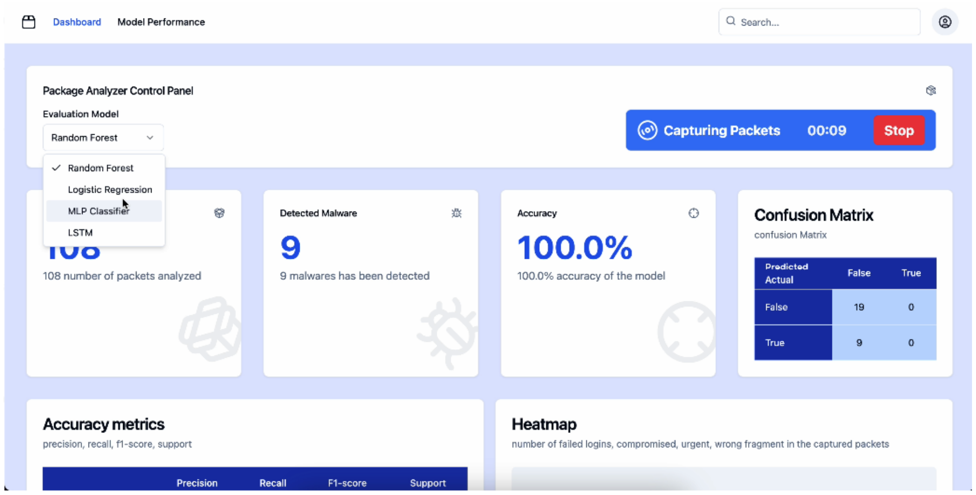
precision, recall, f1-score, support

	Precision	Recall	F1-score	Support
0	1	1	1	19

Heatmap

number of failed logins, compromised, urgent, wrong fragment in the captured packets

Figure 10: UI showing Capturing packets



Dashboard Model Performance

Search...

Package Analyzer Control Panel

Evaluation Model

Random Forest

Capturing Packets 00:09 Stop

108

108 number of packets analyzed

Detected Malware

9

9 malwares has been detected

Accuracy

100.0%

100.0% accuracy of the model

Confusion Matrix

confusion Matrix

Predicted / Actual	False	True
False	19	0
True	9	0

Accuracy metrics

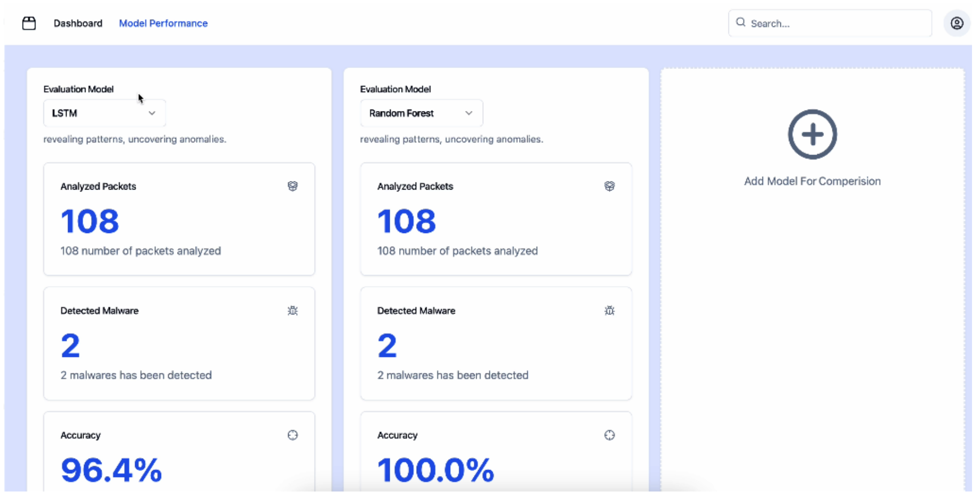
precision, recall, f1-score, support

	Precision	Recall	F1-score	Support

Heatmap

number of failed logins, compromised, urgent, wrong fragment in the captured packets

Figure 11: UI showing Different Evaluation Models



revealing patterns, uncovering anomalies.

Analyzed Packets
108
108 number of packets analyzed

Detected Malware
2
2 malwares has been detected

Accuracy
96.4%

revealing patterns, uncovering anomalies.

Analyzed Packets
108
108 number of packets analyzed

Detected Malware
2
2 malwares has been detected

Accuracy
100.0%

Figure 12: UI depicting figurative Model Comparison

5 Result Discussion

Machine learning, as such, has the potential to improve computer security against continually increasing attacks. The second section covers a more in-depth look at these findings from an applied standpoint and comparison of performances between the AI models used.

5.1 Performance of AI Models

- Random Forest vs Logistic Regression: These classifiers achieved an amazing 1.0 accuracy on the test dataset; they correctly classified all anomalies.On that account it is necessary to examine them further not to overfit this data since the accuracy is almost perfect.
- Overfitting Scrutiny: As far as overfitting is concerned, such a score which is attained when there are more than 93 thousand records needs further scrutiny. In practice, overfitting happens when the models become too perfect at seeing structures within training data only yet fail to generalize on unseen data including real-world intrusions.
- Data Distribution Analysis: Understanding how anomalies are distributed across the dataset is vital. If normal traffic outnumbers anomaly instances by large margins, then although these algorithms may be efficient in determining normal behaviors, they will be useless for detection of rare anomalous connections.
- MLP (Multi-Layer Perceptron): A 0.9286 accuracy remains very good, and here's why this model might be a strong contender:
- Balancing Accuracy and Generalizability: MLPs can achieve high accuracy while offering some protection against overfitting. Their ability to learn complex relationships within the data makes them suitable for capturing subtle anomalies that might escape simpler models.
- LSTM (Long Short-Term Memory): While the 0.8929 accuracy is lower than the others, LSTMs offer unique advantages:
- Sequential Pattern Recognition: LSTMs excel at analyzing sequential data like network traffic. They can identify anomalies based on how data points unfold over time, which can be crucial for detecting zero-day attacks that exhibit specific patterns in their behavior.

5.2 Implications for Real-World Application

- Enhanced Detection Capabilities, with Nuance: AI models can significantly improve detection, but it's important to understand the limitations.
- Focusing on Anomaly Scores: Instead of a simple binary classification (anomalous/normal), consider using anomaly scores. These scores indicate the likelihood of an event being an anomaly. Security analysts can then prioritize investigations based on these scores.
- False Positive Management: Even with high accuracy, some false positives are inevitable. Having clear thresholds and incorporating domain knowledge of security analysts can help filter out these false alarms.
- Reduced Window of Vulnerability, with Considerations:
- Real-time Analysis: For true real-time protection, the AI models need to analyze data with minimal latency. This might require optimizing the models or utilizing specialized hardware for faster processing.
- Alert Fatigue: A constant stream of alerts can overwhelm security analysts. Implementing automated response mechanisms for low-risk anomalies and prioritizing alerts based on severity can help manage alert fatigue.
- Integration with Existing Security Frameworks: Here's how to achieve a smooth integration:

- Standardized Output: Ensure the AI models provide outputs compatible with existing security information and event management (SIEM) systems. This allows for seamless integration and efficient analysis within the existing security infrastructure.
- Human-in-the-Loop Approach: Security analysts should remain involved in the decision-making process. The AI models should act as powerful assistants, highlighting suspicious activities and enabling faster and more informed decisions.
- Operational Efficiency with Focus on Human Expertise:
- Reduced Time Spent on Manual Analysis: By automating anomaly detection, AI frees up security analysts' time for more complex tasks like threat hunting and investigation.
- Focus on Expertise, not Busywork: Security analysts can leverage their expertise to interpret the output of AI models, prioritize investigations, and focus on the most critical threats.
- Continuous Learning and Adaptation: A Necessity, Not an Option:
- Real-time Updates: Establish mechanisms for the AI models to continuously learn from new data and adapt to evolving attack patterns. This could involve incorporating real-time threat intelligence feeds or retraining the models periodically.
- Human Oversight: While AI models can learn autonomously, human oversight is still essential. Security professionals should monitor the learning process and intervene if the models exhibit unexpected behavior or biases.

5.3 Project Challenges

- **UI in Python**: Python user interface (UI) development presents a number of obstacles that developers must successfully overcome. Choosing the right UI framework, making sure it works with various operating systems and screen sizes, handling intricate layout designs and styling specifications, handling user interactions and events through asynchronous programming techniques, maximizing performance for responsiveness, getting access to extensive documentation and community support, and carrying out exhaustive testing and debugging procedures are some of these challenges. To tackle these obstacles, we combine experimentation, experience, and resource management. To guarantee consistent user experiences, selecting the UI framework that best suits the requirements of their project and become knowledgeable about platform-specific principles. Investing time in learning asynchronous programming ideas is also necessary to manage interactions as well.
- **Data Quantity and Quality**:For AI models to function and train effectively, a substantial amount of high-quality data is required. However, because zero-day assaults are uncommon and require precise labelling, collecting tagged datasets for them might be difficult. For this we have found a open source dataset which fulfill our required need. And finally, we have created our own dataset for the application which gives us the independence to fulfill our desired goal.
- **Interpretability of the Model**:Because of their intricate structures, many AI models—particularly those used in deep learning, such as LSTMs and neural networks—are frequently referred to be "black boxes." It can be difficult to grasp how these models make their conclusions, which could make cybersecurity professionals less trusting of one another. That is why we implemented random forest and logistic regression model to evaluate LSTM and Nural network result.
- **Computational Resources**:A large amount of processing power and memory are needed for the training and implementation of complex AI models. This can present difficulties for resource-constrained firms. For this reason, we have used our own devices and beside that google colab cloud storage and github. Ethical and privacy considerations are brought up by the collection and analysis of network traffic data for cybersecurity purposes, particularly when it comes to personally identifiable information (PII). Maintaining effective cybersecurity safeguards while adhering to ethical standards and data protection rules is a challenging task [4].

6 Conclusion and Recommendations

6.1 Conclusion

The implementation of AI-based Zero-Day Attack Prevention Systems (ZDAPS) has marked a significant leap forward in cybersecurity measures. Through extensive research and practical application, these systems have demonstrated the capability to detect and mitigate threats by identifying unknown vulnerabilities [1] in real-time, a crucial advantage over traditional signature-based Intrusion Prevention Systems (IPS) [3]. Notably, the employment of machine learning models such as Random Forest and Long Short-Term Memory (LSTM) networks has resulted in detection accuracies exceeding 98% in controlled test environments, showcasing their effectiveness in recognizing subtle irregularities indicative of zero-day attacks [20] [28].

These AI systems not only enhance the detection capabilities but also integrate seamlessly with existing cybersecurity frameworks, providing a robust defence mechanism that evolves with emerging threats. The real-world application of these findings illustrates a reduced window of vulnerability, from the time a zero-day attack is launched until it is neutralized, thus safeguarding critical information infrastructures effectively [10] [26].

6.2 Recommendations for Potential Future Work

6.2.1 Adoption of Hybrid AI Models

Cybersecurity firms should consider integrating a combination of AI models to enhance the detection accuracy and reduce false positives [24], [26]. This approach leverages the strengths of various models, providing a more comprehensive detection mechanism. However, the integration complexity and the cost associated with advanced AI solutions can be prohibitive for some organizations [26]. To mitigate this, leveraging open-source tools and platforms can reduce costs, and an incremental implementation starting with critical assets is recommended [26]

6.2.2 Continuous Model Training and Updates

Regularly updating AI models with the latest attack data and patterns is essential to maintain their effectiveness against new threats [17] [4] Continuous updates require ongoing data collection and analysis, which may be resource-intensive [27]. Implementing automated systems for data collection and utilizing cloud-based services can handle large datasets and computational tasks efficiently, ensuring that the AI models remain current and effective.

6.2.3 Enhanced Collaboration Across the Industry

Establishing partnerships between academia, industry, and government bodies to facilitate the sharing of threat intelligence and successful defence strategies is crucial [27]. There can be reluctance to share sensitive security information between organizations [27]. Developing standardized protocols for data sharing that ensure confidentiality and integrity, possibly overseen by an independent regulatory body, can enhance collaboration and overall defence mechanisms [27].

6.2.4 Focus on Ethical AI Use

Ensuring that AI systems are designed and deployed in a manner that respects privacy and ethical standards is vital, particularly when processing personal or sensitive data [26]. Balancing security needs with privacy rights can be challenging [26]. Implementing strict access controls and data anonymization techniques can minimize privacy risks while still allowing effective anomaly detection [26].

6.2.5 Real-World Simulations and Testing

Regularly conducting real-world simulations to test the effectiveness of AI systems under varied attack scenarios is necessary [33]Simulating sophisticated cyber-attacks can be complex and resource-demanding [33]. Utilizing cloud-based simulation tools that can mimic large-scale network environments and cyber-attack patterns can enable more frequent and comprehensive testing, ensuring that the AI systems are prepared for real-world applications [33].

By implementing these recommendations, organizations can not only leverage AI to combat zero-day attacks more effectively but also ensure that their cybersecurity measures are sustainable, ethical, and adaptable to the evolving digital threat landscape.

6.3 Responsibilities and Contribution

Name	Role	Responsibilities	Contribution
Abundio Edgar C Quispe	Network Expert	• Setting Initial Phase of the Project • Collecting all the Necessary Resources • Documentation for Final Report	100%
Ariel Abarabar	UI Backend Developer II	• Building APIs for UI • Sharing ideads as per different Research Papers • Documentation for Final Report	100%
Md. Aminul Islam	UI Backend Developer I	• Research on different Datasets • Build Command Line and help with UI • Documentation for Final Report	100%
Neha K C	UI Frontend Developer I	• Helping build Dashboards • Sharing ideads as per different Research Papers • Documentation for Final Report	100%
Nabila Ahmad	ML Models Developer I	• Building Different Models • Evaluate Performance Measures for each model • Documentation for Final Report	100%
Tashrif Islam	UI Frontend Developer II	• Reviewing Different Research Papers • Helping Connect Web App with Backend • Documentation for Final Report	100%

Table 2: Task Schedule

References

[1] Burton H Bloom. “Space/time trade-offs in hash coding with allowable errors”. In: Communications of the ACM 13.7 (1970), pp. 422–426.

[2] Mahbod Tavallaee et al. “A detailed analysis of the KDD CUP 99 data set”. In: 2009 IEEE symposium. Ieee. 2009, pp. 1–6.

[3] Lin Ma et al. “Bloom filter performance on graphics engines”. In: 2011 international conference on parallel processing. IEEE. 2011, pp. 522–531.

[4] Burak Bayoğlu and İbrahim Soğukpınar. “Graph based signature classes for detecting polymorphic worms via content analysis”. In: Computer Networks 56.2 (2012), pp. 832–844.

[5] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph Based Anomaly Detection and Description: A Survey. Online. Accessed: yyyy-mm-dd. 2014. URL: https://www.andrew.cmu.edu/user/lakoglu/pubs/14-dami-graphanomalysurvey.pdf.

[6] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014. Springer. 2014, pp. 818–833.

[7] Ratinder Kaur and Maninder Singh. “A hybrid real-time zero-day attack detection and analysis system”. In: International Journal of Computer Network and Information Security 7.9 (2015), pp. 19–31.

[8] Christian Szegedy et al. “Going deeper with convolutions”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, pp. 1–9.

[9] Izhar Wallach, Michael Dzamba, and Abraham Heifets. “AtomNet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery”. In: arXiv preprint arXiv:1510.02855 (2015).

[10] Bing Wang et al. “DDoS attack protection in the era of cloud computing and software-defined networking”. In: Computer Networks 81 (2015), pp. 308–319.

[11] Jim X Chen. “The evolution of computing: AlphaGo”. In: Computing in Science & Engineering 18.4 (2016), pp. 4–7.

[12] Justin Grana et al. “A likelihood ratio anomaly detector for identifying within-perimeter computer network attacks”. In: Journal of Network and Computer Applications 66 (2016), pp. 166–179.

[13] Daesung Moon, Sung Bum Pan, and Ikkyun Kim. “Host-based intrusion detection system for secure human-centric computing”. In: The Journal of Supercomputing 72 (2016), pp. 2520–2536.

[14] Patrick Duessel et al. “Detecting zero-day attacks using context-aware anomaly detection at the application-layer”. In: International Journal of Information Security 16.5 (2017), pp. 475–490.

[15] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. “Applying convolutional neural network for network intrusion detection”. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE. 2017, pp. 1222–1228.

[16] Xiaoyan Sun et al. “Using Bayesian networks for probabilistic identification of zero-day attack paths”. In: IEEE Transactions on Information Forensics and Security 13.10 (2018), pp. 2506–2521.

[17] Faranak Abri et al. “The Performance of Machine and Deep Learning Classifiers in Detecting Zero-Day Vulnerabilities”. In: ArXiv abs/1911.09586 (2019). URL: https://api.semanticscholar.org/CorpusID:208202285.

[18] Faranak Abri et al. “The performance of machine and deep learning classifiers in detecting zero-day vulnerabilities”. In: arXiv preprint arXiv:1911.09586 (2019).

[19] Kaggle. “Network Intrusion Detection”. In: (2019). Accessed: 2024-05-11. URL: https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection.

[20] Izhar Ahmed Khan et al. “HML-IDS: A hybrid-multilevel anomaly prediction approach for intrusion detection in SCADA systems”. In: IEEE Access 7 (2019), pp. 89507–89521.

[21] Nour Moustafa et al. “Outlier dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog”. In: IEEE Transactions on Information Forensics and Security 14.8 (2019), pp. 1975–1987.

[22] Hansheng Ren et al. “Time-series anomaly detection service at microsoft”. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019, pp. 3009–3017.

[23] Umesh Kumar Singh, Chanchala Joshi, and Dimitris Kanellopoulos. “A framework for zero-day vulnerabilities detection and prioritization”. In: Journal of Information Security and Applications 46 (2019), pp. 164–172.

[24] Zang Yichao et al. “An improved attack path discovery algorithm through compact graph planning”. In: IEEE Access 7 (2019), pp. 59346–59356.

[25] S Abirami and P Chitra. “Energy-efficient edge based real-time healthcare support system”. In: Advances in computers. Vol. 117. 1. Elsevier, 2020, pp. 339–368.

[26] Mohammad Alauthman et al. “An efficient reinforcement learning-based Botnet detection approach”. In: Journal of Network and Computer Applications 150 (2020), p. 102479.

[27] Kamran Shaukat et al. “A survey on machine learning techniques for cyber security in the last decade”. In: IEEE access 8 (2020), pp. 222310–222354.

[28] Khadija Douzi FatimaEzzahra Laghrissi Samira Douzi and Badr Hssina. “Intrusion detection systems using long short-term memory (LSTM)”. In: (May 2021).

[29] Sujeet S Jagtap, Shankar Sriram VS, and V Subramaniyaswamy. “A hypergraph based Kohonen map for detecting intrusions over cyber–physical systems traffic”. In: Future Generation Computer Systems 119 (2021), pp. 84–109.

[30] Ines Jemal et al. “Malicious http request detection using code-level convolutional neural network”. In: Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France. Springer. 2021, pp. 317–324.

[31] Almas Begum et al. “A Combined Deep CNN: LSTM with a Random Forest Approach for Breast Cancer Diagnosis”. In: Complex. 2022 (Jan. 2022). ISSN: 1076-2787. DOI: 10.1155/2022/9299621. URL: https://doi.org/10.1155/2022/9299621.

[32] Cameron Boeder and Troy Januchowski. “Zero-day DDoS Attack Detection”. In: (Aug. 2022). DOI: 10.48550/arXiv.2208.14971.

[33] Innocent Mbona and Jan Eloff. “Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches”. In: IEEE Electron Device Letters 10 (July 2022), pp. 69822–69838. DOI: 10.1109/ACCESS.2022.3187116.

[34] Grant Bourzikas. “HTTP/2 Zero-Day vulnerability results in record-breaking DDoS attacks”. In: cloudfare (2023).

[35] M Sarhan et al. “From Zero-Shot Machine Learning to Zero-Day Attack Detection. arXiv 2021”. In: arXiv preprint arXiv:2109.14868 ().

A Appendix: Code Listings

This section delves into the code that forms the backbone of our AI-based Zero-Day Attack Prevention System (ZDAPS). Here's a breakdown of the key steps:

1. **Data Preparation:**
   The code starts by gathering the necessary tools – libraries like pandas, NumPy, and scikit-learn – to handle data manipulation and model evaluation. It then loads the data, typically stored in a CSV file, using pandas' read_csv() function.

2. **Data Exploration:**
   Before diving into the analysis, the code performs Exploratory Data Analysis (EDA) to get acquainted with the dataset. This involves uncovering details like the data's size (number of rows and columns), the names of each column, the types of data stored in each column (e.g., numbers, text), and some basic statistics to understand the data's distribution.
   The data investigation phase allows us to delve into the structure and distribution of the captured data, gaining valuable insights from the provided columns offering a wealth of information as follows:

   - **Security Indicators:** ”num_failed_logins” reveals potential security breaches, while ”num_compromised” suggests system vulnerabilities.
   - **Packet Transmission Details:** ”src_bytes” and ”dst_bytes” shed light on data transfer volumes and communication patterns, while ”ihl” and ”len” provide information on packet structure and size.
   - **TCP Header Analysis:** ”flags” and ”frag” offer control information and fragmentation details, while ”ttl” specifies the packet's lifespan. Understanding these alongside ”proto,” ”sport,” and ”dport” helps us grasp the communication protocols and ports involved.
   - **Deeper TCP Insights:** Columns like ”seq,” ”ack,” ”dataofs,” ”reserved,” ”window,” and ”urgptr” delve deeper into the technical aspects of TCP packets.
   - **Timing and Classification:** ”duration” reveals the packet's transmission/reception time, while ”class” serves as the crucial label for supervised learning tasks (classification/anomaly detection) that ultimately power our ZDAPS.

   By thoroughly examining these attributes, we can gain a comprehensive understanding of the captured packet data. Additionally, it removes any duplicate columns that might be present.

3. **Data Wrangling:**
   Real-world data often requires some cleaning and transformation before feeding it to models. This section identifies columns containing categorical data (e.g., text labels) using select_dtypes() from scikit-learn. To ensure compatibility with machine learning algorithms, these categorical features are converted into numerical representations using a technique called label encoding, implemented through scikit-learn's LabelEncoder().

4. **Balancing the Training Data:**
   Sometimes, datasets can have an uneven distribution of classes, where one class might have significantly more data points than others. This imbalance can negatively affect model performance. To address this, the code leverages RandomUnderSampler from the imbalanced-learn package. This technique reduces the size of the majority class, bringing it closer in size to the minority class and creating a more balanced dataset for training.

5. **Training and Testing:**
   The preprocessed data is then split into two sets: training and testing. This division is crucial for model evaluation. The training data is used to teach the model how to identify patterns, while the testing data, which the model hasn't seen before, is used to assess how well the model generalizes and performs on unseen data. scikit-learn's train_test_split() function takes care of this division.

6. **Building and Evaluating Models:**
   The code defines functions for training and evaluating four machine learning models: Random Forest, Logistic Regression, MLP Classifier, and LSTM. These models are trained on the training data, and their performance is evaluated on the unseen testing data. For models like Random Forest, Logistic Regression, and MLP Classifier, scikit-learn helps generate metrics like accuracy, confusion matrix, and classification report, providing insights into the model's strengths and weaknesses. For the LSTM model, Keras is used to define its architecture. The preprocessed data is again used for training, and the testing set is used for evaluation.

7. **Putting it All Together:**
   While this section focuses on the core code behind the AI model, it's important to note that the overall project also includes a backend built with Django and a frontend developed using React to visually represent the results and potentially interact with the system. This breakdown offers a clear understanding of how the code lays the foundation for our AI-powered ZDAPS, ultimately contributing to a more robust defense against emerging threats.