Optimized Intrusion Detection System with Temporal Feature Extraction for Network Traffic Classification

With the increase in accessibility of internet, it became possible to fetch and transfer data and information over multiple networks. But the spike in network related threats like worms, malware, viruses, etc. has demanded the necessity of a system which can properly monitor the network activity to detect threats that interrupt the security systems. Intrusion Detection System (IDS) is one such model that properly monitor and study the pattern of network traffic under malicious activity and makes the operators vigilant by alarms or alerts. This system works on the principle that the conduct of a normal user and an intruder is different which makes it possible to distinguish normal and malicious activities. So, an efficient IDS system which can accurately detect the temporally uncorrelated and correlated attacks with proper feature selection for minimizing False Alarm Rate (FAR) and thereby maximizing the detection rate is highly on demand.

Problems under analysis:

1. **Enhancing the detection rate of IDS with optimized FAR**

   An authentic IDS can be established with a reliable feature selection technique. Several features can be extracted from data packets. Efficient detection of attacks requires identification of relevant features. This can be extracted using machine learning algorithms. But in such cases, there is chance that some intrusion attacks are misclassified as normal, and some normal flows are misclassified as intrusions and thereby increasing the FAR.

   So, a technique based on Genetic search algorithm combined with heuristic evaluation is proposed. The basic idea is to figure out the relation between a feature and class, eventually utilizing maximum correlated features for further feature extraction with fitness calculation using Genetic algorithm. These features further serve as the input to the classification model.

2. **Detecting temporally uncorrelated and correlated attacks with Bi-directional Long Short-Term Memory (BiLSTM) and attention mechanism**

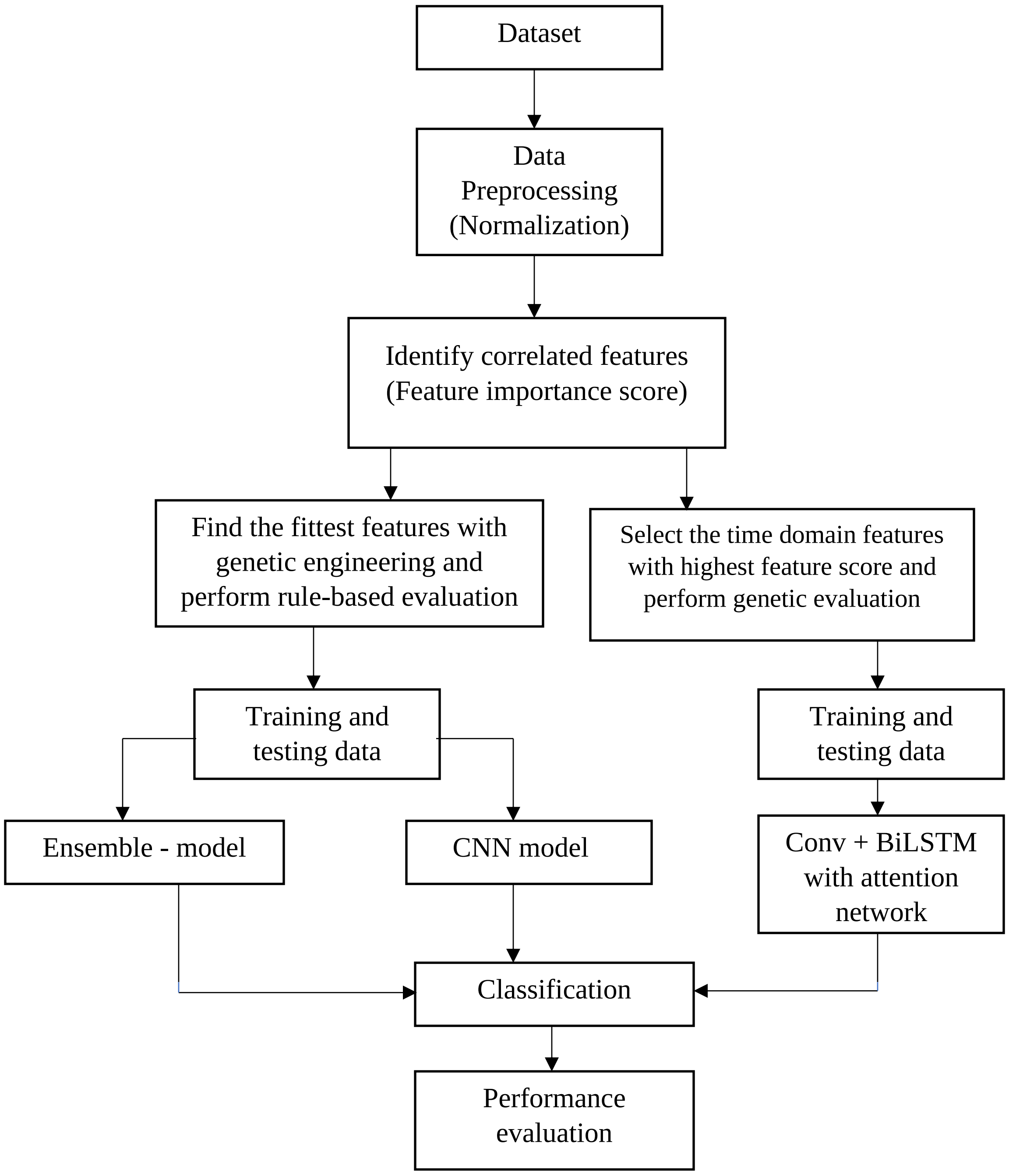
   Normal classification algorithms perform simplified classification of attacks as: normal and malicious. To predict temporally correlated attacks like Denial-of-Service (DoS) and Man-in-the-Middle (MITM) (variants of malicious attacks) it is necessary to consider some time domain features which is difficult to collect manually. Normal deep learning techniques when deployed independently consider each data packet as same, thereby ignoring the inter-relations among the multiple data packets of network traffic. So, inter-packet patterns in time domain i.e., sequential features are to be extracted, as features in different packets can be different.

A hybrid model with BiLSTM and attention mechanism is proposed which will utilize the time-domain and global features for predicting events in future time steps and thereby detecting temporally correlated threats by identifying the similarity in feature vectors.

Datasets (Collected both):

1) UNSW-NB15 network dataset
2) CIC DDOS 2019 dataset

Design Steps: -


Dataset
Data Preprocessing (Normalization)
Identify correlated features (Feature importance score)
Find the fittest features with genetic engineering and perform rule-based evaluation
Select the time domain features with highest feature score and perform genetic evaluation
Training and testing data
Training and testing data
Ensemble - model
CNN model
Conv + BiLSTM with attention network
Classification
Performance evaluation

- Data preprocessing (Normalization) for modelling different scales.
- Find the best correlations and develop the feature score graph.
- Find the fittest features with genetic engineering.
- Apply rules.
- Now, specifically select the time-domain features with highest feature score as random samples for genetic evaluation.
- Apply features to the classification model (Ensemble model, CNN, BiLSTM).
- Compare their performance and evaluate the False Alarm Rate.

Frameworks and Libraries

- Language: - Python
- Framework: - Keras and TensorFlow
- Library: - NumPy, Pandas, Matplotlib

Handson details: -

i. Handson 1: - Performed XGBoost classification on a rough dataset without any preprocessing.

ii. Handson 2: - Encoded the categorical data into numerical data using Ordinal Encoder.

iii. XGBoost feature selection and XGBoost-based classification

- Step 1: - Loaded both training and testing data with pandas.
- Step 2: - Performed **preprocessing with Ordinal Encoder**.
  - ➢ Dataset has 42 features out of which 3 features are categorical and 30 features are nominal (‘state’, ’proto’, ’service’).
  - ➢ Result: - Successfully converted ‘state’ and ‘proto’ into nominal instances.
  - ➢ Feature ‘service’ is categorical, and several missing data are present. So first ‘- ‘is converted to ‘NaN’ and then fillna(‘nodata’) function was used to convert all features to categorical and then Ordinal Encoder was applied.
  - ➢ Result: - Converted and when output was printed it showed numerical value. But, during training with this feature error occurred.
- Step 3: - Performed **preprocessing with MinMaxScaler** and transformed all data to the range [0,1].
- Step 4: - Performed **classification with XGBoost classifier before feature selection** for performance comparison.
- Step 5: - **Obtained the feature score graph** with the **plot_importance()** function.
- Step 6: - Since XGBoost classifier is used features with high scores were manually applied to the model.

- Step 7: - Performed **classification with XGBoost classifier before feature selection**.
- Learning rate: - 0.1

➢ **To be completed**

- Need to solve the problem with feature “service”.
- Look for techniques which can improve the accuracy.
- Perform for different parameters in XGBoost classifier like learning rate.

iv. Autoencoder-based feature selection and XGBoost-based classification

- Step 1: - Imported all the required packages.
- Step 2: - Loaded both training and testing data with pandas.
- Step 3**: -** Performed **preprocessing with Ordinal Encoder**.
  - ➢ Dataset has 42 features out of which 3 features are categorical and 30 features are nominal (‘state’, ’proto’, ’service’).
  - ➢ Result: - Successfully converted ‘state’ and ‘proto’ into nominal instances.
  - ➢ Feature ‘service’ is categorical, and several missing data are present. So first ‘- ‘is converted to ‘NaN’ and then fillna(‘nodata’) function was used to convert all features to categorical and then Ordinal Encoder was applied.
  - ➢ Result: - Successfully preprocessed the feature ‘service’.
- Step 3: - Performed **preprocessing with MinMaxScaler** and transformed all data to the range [0,1].
- Step 4: - Created input, encoder, bottleneck, decoder and output layer of autoencoder.
  - ➢ Created **input layer with 42 units**.
  - ➢ Created **2 encoder layer** each with a dense layer, batch normalization layer and leaky ReLU layer. **First layer contains 84 units, and second layer contains 42 units.**
  - ➢ Created the **bottleneck layer (dense layer) with 21 units.** (This is the actual output we want).
  - ➢ Created the **decoder part with 2 layers** each with a dense layer, batch normalization layer and leaky ReLU layer.
  - ➢ Created the **output layer with Linear activation function, optimizer = 'adam', loss='mse'.**
  - ➢ Train the model and then save the model with output as bottleneck.
- Step 5: - Performed **classification with XGBoost classifier before feature selection** for performance comparison.
- Step 6: - **Fit the saved autoencoder model data into XGBoost classifier model and perform classification.**
- Compare the performance.
- Learning rate: - 0.1.

➢ **To be completed**

- Perform same test with different optimizers and learning rate.

v. XGBoost feature selection and DNN-based classification

- Step 1: - Loaded both training and testing data with pandas.
- Step 2: - Performed **preprocessing with Ordinal Encoder (Same as in 'iii')**.
- Step 3: - Performed **preprocessing with MinMaxScaler** and transformed all data to the range [0,1].
- Step 4: - Performed **classification with DNN before feature selection** for performance comparison.
    - Created 5 dense layers with 42, 84, 168, 168, 336 layers respectively.
    - Dropout layers are added in between.
    - Activation function used is 'relu', loss="sparse_categorical_crossentropy", optimizer="adam".
    - Output layer has 2 units and sigmoid activation function is used.
- Step 5: - **Obtained the feature score graph** with the **plot_importance()** function.
- Step 6: - Since XGBoost classifier is used features with high scores were manually applied to the model.
- Step 7: - Performed **classification with DNN**.

➢ **To be completed**

- Accuracy obtained is very less and need to improve it by using accuracy improvement techniques.

vi. XGBoost feature selection and CNN-based classification

- Step 1: - Loaded both training and testing data with pandas.
- Step 2: - Performed **preprocessing with Ordinal Encoder (Same as in 'iii')**.
- Step 3: - Performed **preprocessing with MinMaxScaler** and transformed all data to the range [0,1].

➢ **To be completed**

- How to input the non-image data to CNN?

References: -

[1] Rajagopal S, Kundapur PP, Hareesha KS. A stacking ensemble for network intrusion detection using heterogeneous datasets. Security and Communication Networks. 2020 Jan 24;2020.

[2] Devan P, Khare N. An efficient XGBoost–DNN-based classification model for network intrusion detection system. Neural Computing and Applications. 2020 Jan 19:1-6.

[3] Aslahi-Shahri BM, Rahmani R, Chizari M, Maralani A, Eslami M, Golkar MJ, Ebrahimi A. A hybrid method consisting of GA and SVM for intrusion detection system. Neural computing and applications. Aug;27(6):1669-76.

[4] Hsu CM, Hsieh HY, Prakosa SW, Azhari MZ, Leu JS. Using long-short-term memory based convolutional neural networks for network intrusion detection. InInternational wireless internet conference 2018 Oct 15 (pp. 86-94). Springer, Cham.

[5] Gao J, Gan L, Buschendorf F, Zhang L, Liu H, Li P, Dong X, Lu T. Omni SCADA intrusion detection using deep learning algorithms. IEEE Internet of Things Journal. 2020 Jul 14;8(2):951-61.

Guide details:

Prof. Sumod Sundar
TKM College of Engineering, Kollam
Email: sumodsundar@tkmce.ac.in
Mob: 8086515716