**LOW LEVEL DESIGN(HLD)**

**Network Security With Machine Learning**

**ABHIJIT SEN**

**iNeuron**

Contents:

Document Version Control………………………………………………………………………………….. 3

1. Introduction……………………………………………………………………………………………………. 4

1.1 What is Low-Level Design Document?.......................................................... 4

1.2 Scope…………………………………………………………………………………………………………. 4

2. Architecture……………………………………………………………………………………………………. 4

3. Architecture description…………………………………………………………………………………. 5

3.1 Data description………………………………………………………………………………………….. 5

3.2 Data transformation……………………………………………………………………………………. 5

3.3 Data pre-processing……………………………………………………………………………………. 5

3.4 Data visualization………………………………………………………………………………………… 5

3.5 Feature Selection………………………………………………………………………………………… 6

3.6 Modeling…………………………………………………………………………………………………….. 6

3.7 Data Validation…………………………………………………………………………………………… 6

3.8 Deployment……………………………………………………………………………………………….. 6

**Document Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Issued** | **Version** | **Description** | **Author** |
|  | **V1.0** | **Initial LLD-V1.0** | **ABHIJIT SEN** |

1. Introduction
   1. What is Low-Level Design Document?

The goal of LLD or Low Level Design document is to give the internal logical design of the actual program code. Low level design is created based on the High Level Design. LLD describes the class diagrams with the method and relations between classes and program specs. It describes the modules so that the programmer can directly code from the document.

* 1. Scope

Low Level Design (LLD) is a component level design process that follows a step by step refinement process. This process can be used for designing data structures, required software architectures, source code, and ultimately performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

1. Architecture:

Data Transformation

Data Acquisition

Data Pre-processing

Business Problem

Data Visualization

Feature selection

Modeling

Evaluation

Deployment

1. Architecture description:

3.1 Data description:

The primary source of data for this project is the [NSL-KDD dataset](https://www.unb.ca/cic/datasets/nsl.html). This is an open source benchmark dataset for testing machine learning algorithms curated for designing Intrusion Detection System using ML/DL techniques. This dataset is prepared by the Canadian Institute of Cybersecurity. The dataset consists of Train and Test dataset having filenames as **KDDTrain+.ARFF and KDDTest+.ARFF. The test dataset has distribution slightly different from the train dataset which results in lower test accuracy without feature selection or feature engineering. Both the train and test dataset consists of various parameters of a network packet and all the datapoints are labeled as either normal (meaning safe) or anomaly (meaning unsafe) network packet. The data distribution in the test dataset is intentionally kept different from that of the train in order to make the model zero day attack proof.**

The train dataset consists of 125972 rows and test dataset consists of 22543 rows. Both of them have 41 different attributes. Both the datasets are given in .arff file format.

3.2 Data transformation:

The original dataset is given in .arff file format. In this transformation step we convert the .arff file to .csv file for our working convenience. Further, we don’t store the transformed data in any database as there is only a single tabulated data which gets easily fit into the memory.

3.3 Data pre-processing:

In this particular step we check for the class distribution to verify if the distribution is balanced or skewed. Further we check for any data duplication and remove them if any. We also check for any null values and impute them if any. We check the unique values in each of the columns and remove columns which have only a single value in all of the rows. The class in the original dataset is labeled as either **normal** and **anomaly** which is transformed as 0 for normal and 1 for anomaly.

3.4 Data visualization:

In the dataset there are primarily two types of features, categorical and numerical. In the categorical feature, we check the count for each of the categories against each class which gives us an idea about the distribution of various categories in the normal and anomaly class. Here we conclude that wider the difference in count better is the category.

However, for the numerical features, we plot the violin plot, box plot, pdf and cdf plots. The violin and box plot gives the distribution of the mean value for the various classes and the pdf/cdf plot gives us an idea of data distribution for the various classes. Here from the violin plot we say wider the mean for a feature against the two classes better is the feature and wider the distribution in pdf better is the feature.

3.5 Feature Selection:

As we are already given with the fact that the distribution of the train and test dataset is different so modeling with all the features is surely not going to work as good as one would expect. Hence, we apply various types of feature selection process like, manual, statistical, ML based and Auto-encoder based to get the most relevant features in distinguishing between a normal and anomalous traffic.

3.6 Modeling:

We train various classification based ML models for doing binary classification. After getting the features against various types of feature selection techniques, we train various ML/DL models for each type of dataset. While training, we do hyper parameter tuning with 10 folds cross validation with grid search. For each of the model, we calculate the accuracy as a KPI for model performance also keeping an eye on the False positive and False Negative rates.

3.7 Data Validation:

Here Data validation is done on the test set.

3.8 Deployment:

The best model will be deployed in AWS EC2 instance using Flask API.