github repo link

The hands-on implementation of the Intrusion Detection System (IDS) is being developed in VS Code. Here is the GitHub repository link for the project.: <https://github.com/Itzswika/Intrusion-detection-2>

March: 17th - 23rd

17

* Completed the assigned topics in the computer networks book by kurose and ross
* Started research on building an intrusion detection system

KEY LEARNINGS:

1. Got familiar with computer networks.

* How does the internet work?
* Service model
* Protocols
* Transport layer and network layer
* IP address
* SDN

1. INTRUSION DETECTION SYSTEM

* Started going through the articles, research papers, and a few videos.

18

PROGRESS ON INTRUSION DETECTION SYSTEM.

1. Understanding the key features in the dataset.

[CIC-IDS2017](https://www.unb.ca/cic/datasets/ids-2017.html) : This dataset has the most common up to date attacks.

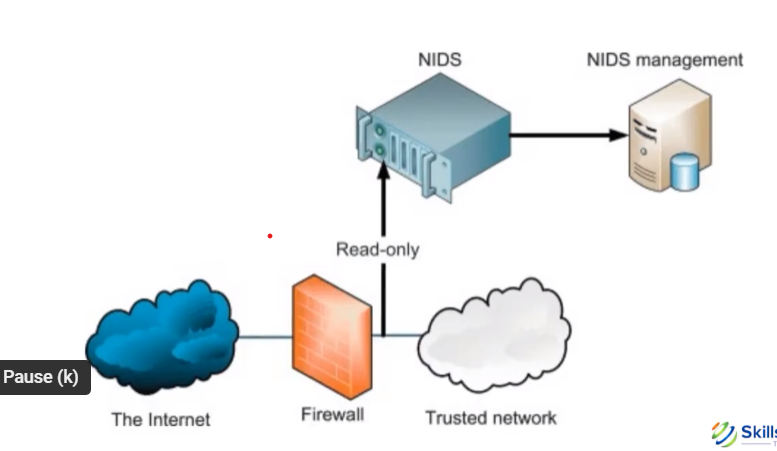
19,20,21

IDS: HELPS TO DETECT POTENTIAL CYBER THREATS AND SECURITY BREACHES.

Types :

1. Network based IDS: monitors network traffic for suspicious activity, entire network.

* Packet inspection, traffic analysis\
* Sees all devices and network activity
* Detects network wide attacks
* Deployed on firewalls and routers
* Tools: suricata, zeek,



1. Host based IDS: detects attacks targeting a single system

* Detection type: file integrity, log analysis and process monitoring.
* Limited to a specific host
* Installed on each device.

1. Signature based IDS: detects known attacks, it comes under supervised learning.
2. Anomaly based IDS: detects malicious activities by analysing the outliers, it comes under unsupervised learning.
3. Hybrid IDS: customized using more than one type of IDS.

* A hybrid based model combining all the layers to increase the security.

22

IDS:

Layers :

1. Data capture layer:

* A dataset will be taken

1. Preprocessing layer

* Once captured, they undergo filtering, normalization and reassembly to remove noise and structure the data. - numpy

1. Detection engine layer

* Signature based detection: uses database of known attack patterns - snort, suricata, ossec
* Anomaly based detection: detects deviations from normal traffic behaviour using ml methods - scikitlearn (ml models). tensorflow(deep learning), pacaret(auto ml library)

1. Logging and alerting layer:

* After a potential intrusion, the system logs the event and triggers an alert for admin (flask, fastapi)

1. Response and mitigation layer:

* Response mechanism :

1. Passive response - only logs and alerts, does not interfere
2. Active response - blocks traffic, resets connections, triggers firewall rules.
3. Integration with firewalls & SIEM for automated mitigation

* Tools:
* **iptables (Linux):** Blocks suspicious IPs using firewall rules.
* **Fail2Ban:** Bans IPs after repeated malicious activity.
* **Snort/Suricata Inline Mode:** Can drop malicious packets.
* **SOAR (Security Orchestration, Automation, and Response):** Automates threat responses.
* **SIEM (Security Information & Event Management):** Automates mitigation workflows.

1. Management and reporting layer:

* Admin control, report generation and system tuning

1. Defining and updating detection rules
2. Generating periodic security reports
3. Monitoring IDS health and performance
4. Fine tuning detection accuracy and reduce false positives.

23

* I went through a github repository : <https://github.com/noushinpervez/Intrusion-Detection-CICIDS2017>

1. This repository describes the structure and contents of the cicids2017 dataset.
2. It analyzes characteristics like feature importance, balancing the data, attack types etc
3. The cleaned data is further used to train and evaluate ML models for IDS

MY NOTES:

* The CIC-IDS2017 dataset contains network traffic data for the development and evaluation of intrusion detection system
* The dataset is quite huge and has more than 70 features, so PCA is performed to reduce dimensions and preserve the important features.
* The goal is to successfully develop an IDS using the dataset with performance metrics accuracy, precision, recall and f1 score.
* Accuracy: measures the overall correctness of the model
* Precision: measures how many of the predicted attacks are actually attacks
* Recall: measures how many actual attacks were correctly detected.
* F1 score: balances precision and recall, useful for imbalanced datasets. (these are the common performance metrics for most of the models, i would be using in developing the IDS.\

WORKING WITH THE DATASET:

* Libraries used :

1. Numpy, pandas, seaborn, missingno, matplotlib

* These libraries were used to analyze the data characteristics, analyze and visualize the dataset.

IDS\_Paper

An Efficient Intrusion Detection Solution for Near-Real-Time Open-RAN

* The paper explores model poisoning attacks within ensemble learning systems and proposes an IDS that is integrated into the near RT–RIC.
* The paper gives a brief idea about the architecture and challenges faced while performing the architecture and solutions used to mitigate the challenges.
* Workflow :

1. Th traffic collected from the user equipment is stored in shared data layer
2. The IDS access this traffic through API
3. The data is preprocessed
4. Trains a supervised ensemble classifier and learns to identify the patterns.
5. The trained model i deployed back into the nearRTC-RIC
6. It monitors traffic, detects threats and flags on intrusions in real time.
7. If malicious activity is detected, the system can trigger automated mitigation

* Dimensionality reduction

1. Extra tree classifier is chosen as feature selection technique
2. Explainable ai is used to interpret why the SEL took a certain decision.

Future work

So the whole idea is to do Intrusion detection in an O-RAN dataset

So a high level roadmap is below

1. Learn about the basics of networks (UDP, TCP, OSI model, packets etc…)
2. Learn about the basics of intrusion detection system
3. Pick a Dataset for doing Intrusion Detection (<https://www.unb.ca/cic/datasets/ids-2017.html>)
4. Then learn about the dataset (feature set) on which intrusion detection has to be performed
5. Then learn (mostly implementation part) about all the ML/ DL tools and techniques required for the IDS
6. **Milestone 1**: Perform Intrusion detection on a small Dataset with a good accuracy
7. Then learn the basics of O-RAN (Architecture, CU, DU, Interfaces).
8. Then get the O-RAN dataset and study that
9. **Milestone 2**: Then perform Intrusion detection on that Dataset and achieve a good accuracy.

March: 24th - 31st

24

Learnings:

* Going through the github repository.
* Implemented the code from the github repository.
* DATASET

1. The dataset consisted of two different sets: one was raw network capture, and the other was Machine Learning CVE, which had csv files that could be used to train machine learning models.
2. The data was structured but had missing, duplicate values
3. They were first resolved to ensure dataset is clean for further processing
4. Understanding the dataset characteristics and visualized the missing values using seaborn and matplotlib
5. There were 79 columns with 78 numerical features and a categorical label column in the dataset, correlation analysis (measures the dependence of variables with each other) was performed to identify features with positive correlation.
6. 32 features were found to be positive and can be considered for the model training.

IMPLEMENTATION:

* Identified duplicate rows and missing or invalid values.
* The duplicated rows were dropped
* The data description indicated that the dataset has infinity values.
* They were counted as missing values.
* The no. of missing values is 1564
* In the dataset, flow bytes and flow packets contain missing values.
* It is observed that flow bytes and flow packets are continuous variables and the data is not normally distributed. It has extreme values
* The option is to fill in missing values with median value.
* Filling the missing values does not disrupt the distribution of the data.

25

VISUALISING THE ANALYSED PATTERNS:

* In the following data visualization, heatmap is used in understanding the large datasets efficiently.
* Each attack is mapped to their attack type for a better understanding.
* Here's an example:
* BENIGN': 'BENIGN',
* 'DDoS': 'DDoS',
* 'DoS Hulk': 'DoS',
* 'DoS GoldenEye': 'DoS',
* 'DoS slowloris': 'DoS',
* 'DoS Slowhttptest': 'DoS',
* 'PortScan': 'Port Scan',
* 'FTP-Patator': 'Brute Force',
* 'SSH-Patator': 'Brute Force',
* 'Bot': 'Bot',
* 'Web Attack � Brute Force': 'Web Attack',
* 'Web Attack � XSS': 'Web Attack',
* 'Web Attack � Sql Injection': 'Web Attack',
* 'Infiltration': 'Infiltration',
* 'Heartbleed': 'Heartbleed'
* Correlation analysis is applied to identify relationships between different features.
* Then, heatmap is used to visualize the relation between the features.
* It is observed that many pairs are highly correlated and it can cause overfitting.
* NaN correlation coefficient occurred as there is no meaningful relationship with any other columns. They appear blank in the heatmap.
* Further linear relationship analysis is performed to check how one feature changes with respect to another.
* It helps in removing unnecessary features.

26

Network traffic analysis:

1. The goal is to perform classification on the network dataset, the model needs to identify whether the incoming network traffic is malicious or benign (traffic that does not pose any security threat.
2. The dataset has a wide range of attacks.

* Attacks are mapped to the attack type for generalisation
* The network dataset contains a total of nine distinct types of attacks.

1. Benign
2. DDoS
3. DoS
4. Port scan
5. Brute force
6. Web attack
7. Bot
8. Infiltration
9. Heartbleed

Importance of network traffic analysis in CICIDS 2017 for intrusion detection

* It contains real world normal and attack traffic.
* It provides flow based, packet based and behavioural features for ml models
* Helps in improving detection accuracy and reduce false positives
* It is a standard dataset for comparing intrusion detection techniques
* Helps detect evolving threats
* Supports building a real time threat detection system.

Note: the github repo has my updated implementation of the intrusion detection system for CICIDS 2017

27

Implementing the github repo :

Furthering with linear relationship analysis:

* To visualize linear relationship of columns for continuous numerical variables, the dataset is assessed whether the sample dataset is representative of the original dataset by comparing the mean values of numerical features.
* And why is this important? The rep sample ensures that analysis and conclusions apply to the entire population.
* Feature variation is performed to identify the features with high variations (a significant amount of difference between mean, median and mode).
* Why? If the features are highly variant, it will affect the further modeling process.
* Perform unique value count to understand the features after cleaning the data

Interpreting the below graph:

1. Categorical columns are shown using bar plots, the count of each unique value
2. Numerical columns are shown using histograms, how values are distributed.
3. Both of them show one dominant value with a very high count, while other values are less frequent. This suggests that most traffic dataset share similar flag behaviour.
4. Certain flags are rarely used, which may help in detecting attacks if a particular flag becomes more common.
5. Min\_seg\_size\_forward has extreme values, indicating possible outliers.
6. The attack type distribution is highly skewed, might require balancing techniques like oversampling or weighted loss functions
7. Outliers should be treated using techniques like scaling, transformation, and capping extreme values.

* In the attack type plot, it shows a high imbalance where one attack type such as benign is dominating, meaning the dataset has skewed class distribution.
* The other categorical features also show a concentration towards a few specific values, suggesting that most records belong to limited categories.
* And we infer that it requires resampling techniques or it could lead to biased model training.
* Histograms represent numerical data, in the plot, it shows a strong left skewed distribution.

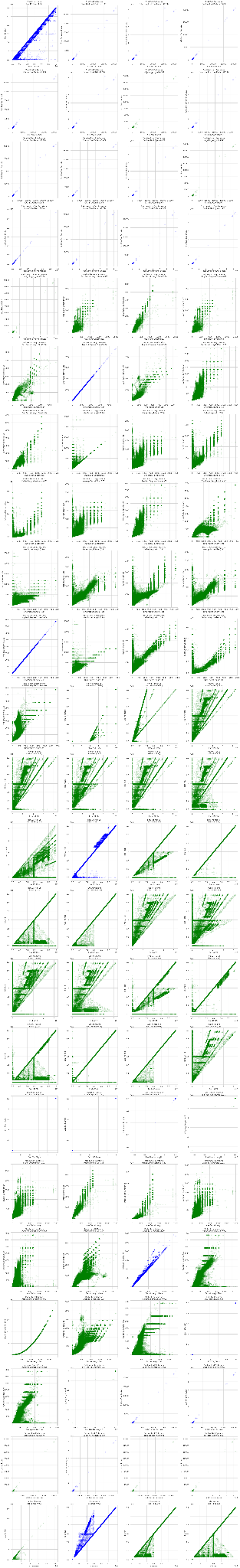


28

UNDERSTANDING THE VARIABLE RELATIONSHIP USING CORRELATION MATRIX:

* It helps identify relationships between features, detect multicollinearity, and select important variables for modeling.
* Essential step in data analysis and feature selection to understand the variable relationships before applying the machine learning models.
* 1 represents perfect positive correlation
* Strong is 0.7 to 0.9
* Moderate is 0.4 to 0.6
* Weak is 0.1 to 0.3
* 0 represents no correlation
* And -1 indicated perfect negative correlation
* -0.7 to -0.9 indicates strong negative correlation
* -0.4 to -0.6 indicates moderate negative correlation
* -0.1 to -0.3 indicated weak negative correlation.

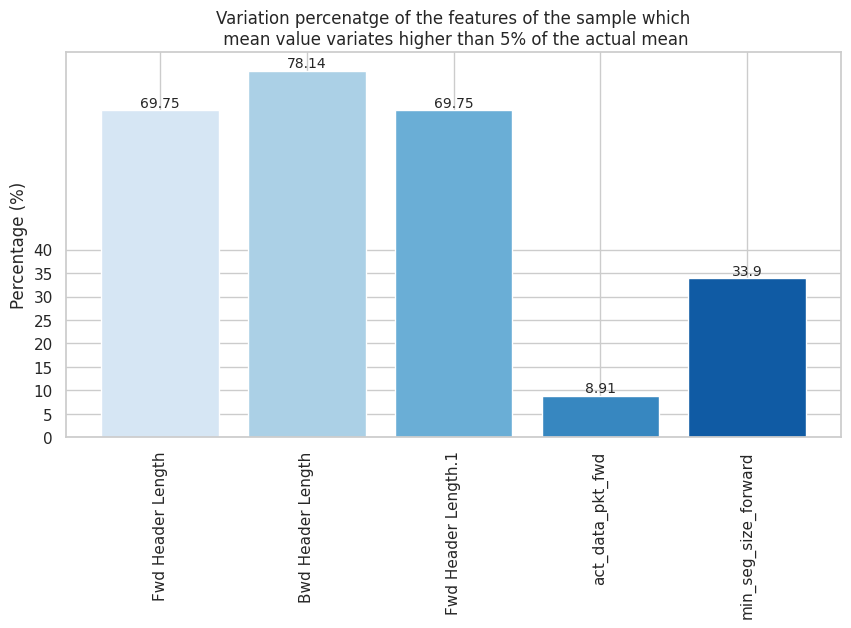
Here's a graph representing strongly positively correlated features in the sampled data that have a correlation coefficient of 0.85 or higher: refer the line 106 in github repo



29

Interpreting the graph :

* The graph shows features with high variation, features with higher percentage of variation are inconsistent between the sampled data and original data
* This could suggest potential bias in the sampling process
* Unnecessary features are dropped from the sampled data, so that the model trains and learns the attack patterns.
* For ex: attack number is dropped from the sampled data ensuring the model learns the attack patterns from features like packet sizes and flag counts.
* Bias can affect the model learning, the model may memorize the important labels ignoring the real attacks.

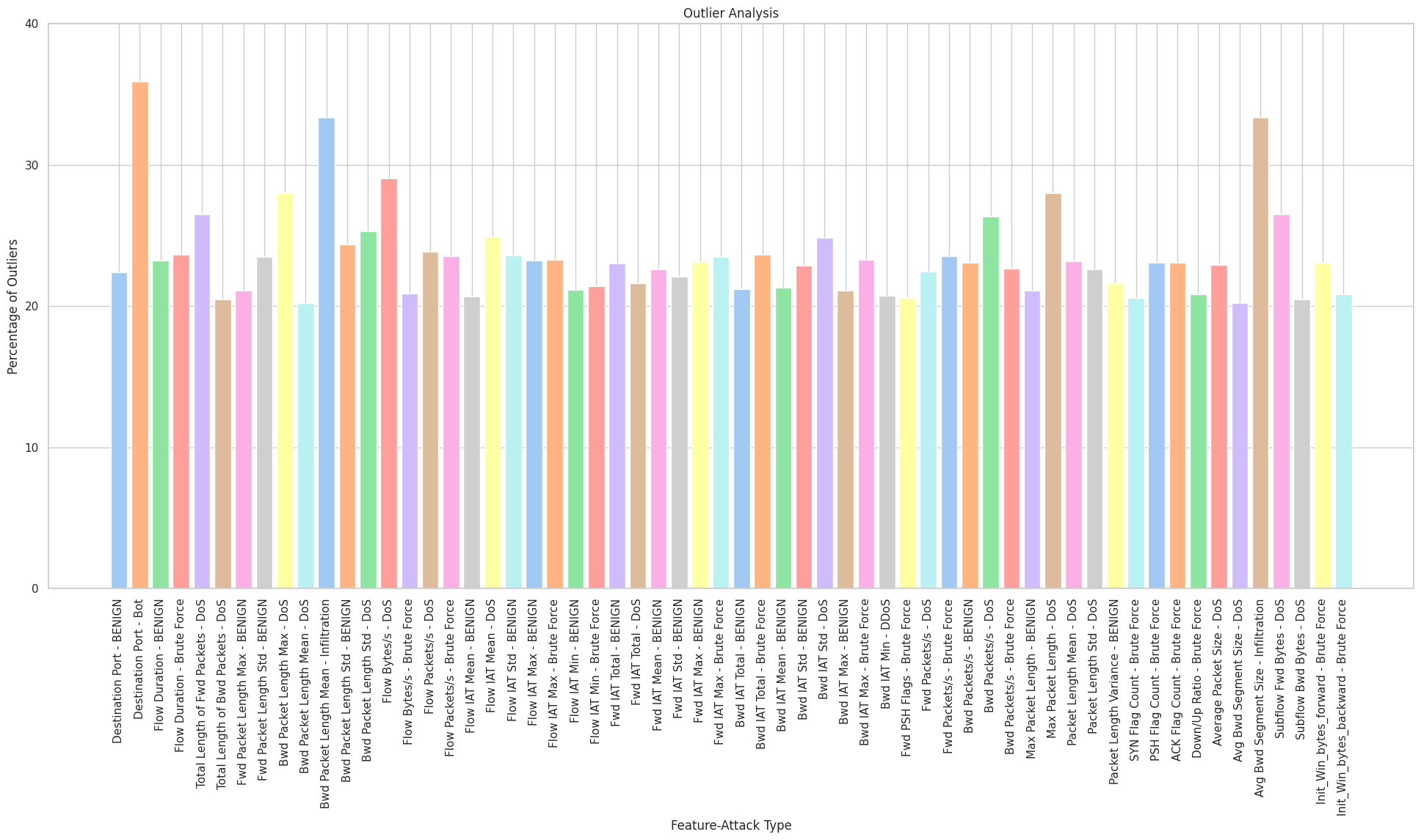


30

OUTLIERS IN THE DATASET:

* Outliers are identified in numerical data, they are data points which are significantly different from the rest of the data points. Their presence can greatly affect the model's performance.
* Method used : interquartile range method
* Why? It is a robust statistical method, resistant to extreme values and effective for skewed data.
* For example: network traffic features are not normally distributed, it works well for both left and right skewed data.
* Features with strong correlation and high variance are considered for further model training as they possess strong deviation (potential attack)
* Features with weak correlation and high variance should be dropped as they only cause overfitting and unnecessary noise in the model training.
* Identifying outliers based on attack types can help us find extreme values
* Which with further analysis, will find out whether to keep them, transform or remove.
* Each outlier based on attack type has been counted.

Interpretation of the graph:

* The below graph visualizes outliers for each attack type, if a feature consistently has a higher percentage of outliers then it has extreme values, which can be used for further processing.
* 

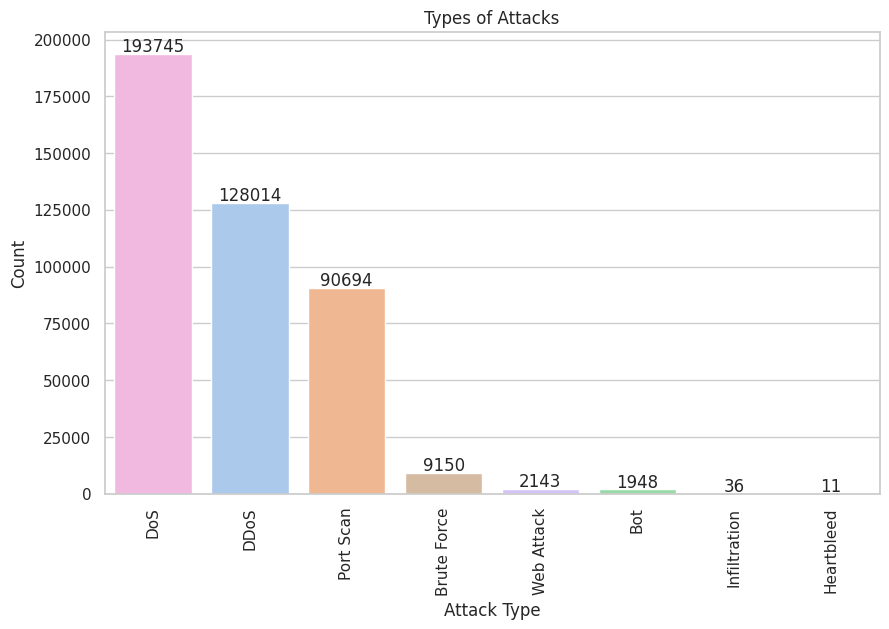
31

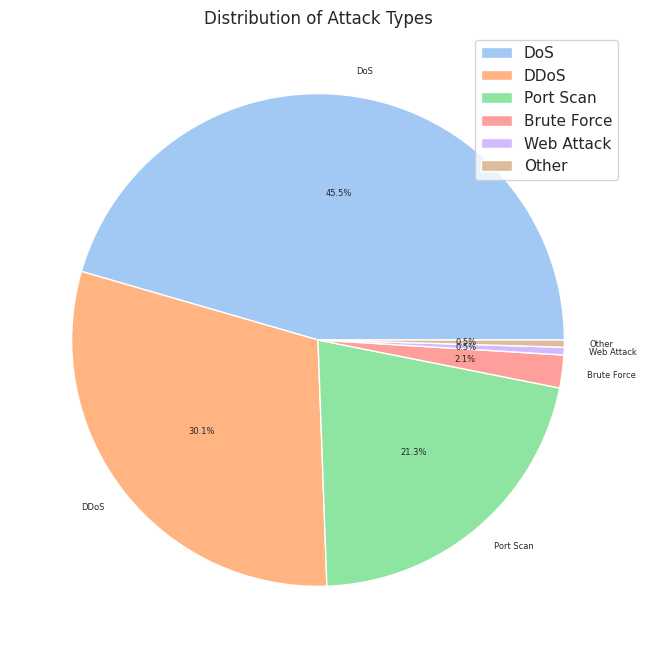
PROGRESS IN A NUTSHELL:

* Due to the large dataset, 20% of the dataset was sampled out from the original dataset for analysis.
* The sampled data is successfully assessed whether it belongs to the original dataset through descriptive statistics
* Histogram and bar plots : used to understand data distribution, detect patterns and spot anomalies
* Scatter plots for highly correlated features: identified features with high correlation, helping detecting multicollinearity, which can affect the machine learning models
* Outlier analysis: identified outliers by attack type and found many extreme values
* Since outliers can indicate network intrusion attempts, they were analysed rather than directly removed.

Interpretation of the graph:

1. Dos, DDos and port scan are the highest type of attack so the dataset is highly skewed, rare attacks like infiltration and heartbleed need special handling while training the models.





Interpreting the above graph:

* Distribution of attack types while grouping smaller percentages into other categories.
* A pie chart is used to visualize major attack types clearly.

April: 1st - 7th

1

LEARNINGS:

GITHUB REPO:

* Through hands-on implementation of the github repository, I gained an understanding that -

1. The primary objective of the repository is to implement, compare and evaluate different ML models for intrusion and detection systems.
2. It provides a structured pipeline that can be replicated on other datasets for traffic analysis and data visualization.
3. It analyzes the impact of preprocessing and class balancing (addressing the issues of imbalanced dataset - one class significantly has more samples than others) on model performance
4. It can also help identify the best performing model for the real world IDS deployment.

HOW THIS IMPLEMENTATION CAN HELP IN MY RESEARCH ON THE O-RAN DATASET:

* The approach can be adapted in the following ways:

1. Dataset analysis and preprocessing:

* The preprocessing techniques (handling missing data, standardization, PCA, SMOTE) can be applied to O-RAN
* Class imbalance techniques are crucial if the dataset has more normal traffic than malicious traffic leading to bias towards normal traffic. Balancing the dataset is crucial to ensure proper recognition of minority classes.

1. Feature engineering

* Can analyze which features are most relevant for further data processing.
* Network flow characteristics might be relevant in O-RAN as well.

1. Building a base model

* The binary classification model can be trained on O-RAN to detect intrusions.
* The multi-class model can classify different network anomalies
* Feature selection methods such as PCA, correlation analysis can help refine which aspects are most important.

1. Model comparison and optimization

* The models trained on CIC-IDS 2017 dataset can be fine tuned.
* The same evaluation techniques can be used to check which model performs the best.

2

DATA PREPROCESSING:

* It is a crucial step as it ensures the dataset is clean, structured and optimized for machine learning models.
* Drop redundant and unnecessary features.
* The data with zero standard deviation were dropped as they do not have any variance. It means there is no meaningful relationship with any other columns
* Memory related errors were addresses to improve the performance
* Visualizations for attack types were performed, check the github repo for the hands on implementation.

3

PCA: PRINCIPAL COMPONENT ANALYSIS.

* Reducing the number of features while preserving most important information is a crucial task in improving the performance of machine learning models.
* PCA is a technique that transforms correlated features into a set of uncorrelated components.
* Use case - feature selection and faster computation.
* IncrementalPCA was performed to retain information. 99.23% got retained.
* Why was it performed? It processes the data into smaller chunks, making it scalable for large datasets.
* It can be updated incrementally
* The goal of pca is to keep the most important information while removing redundancy.
* It may be slightly less accurate than standard PCA but works for large datasets and real time processing.

4

LOGISTIC REGRESSION FOR BINARY CLASSIFICATION:

* It is a supervised machine learning algorithm used for binary classification
* It predicts probabilities for categorical outcomes, such as spam or not spam
* Two logistic regression models are trained based on the chosen parameters

1. The number of max iterations : max\_iter : why? - it controls the max number of iterations the algorithm is allowed to converge to a solution.
2. Lr used iterative optimization algorithms to find the best weights that minimize the cost function.
3. The algo update weights step by step getting closer to the optimal solution with each iteration.

5

Implementing support vector machine:

* Supervised machine learning model used for classification and regression analysis.
* It finds the best boundary to separate classes
* It uses kernels, in this model kernel = poly, it uses a polynomial kernel, which means it will try to separate the data with a curved decision boundary
* Regularization parameter - it balances margin width with classification accuracy.

6

Overall progress report: milestone 1:

* Goal : to implement a simple intrusion detection system on CICIDS-2017 dataset using machine learning techniques.
* Focus : it has been on understanding the dataset, preparing it through EDA and getting it ready for model training

LEARNINGS:

* The original size of the dataset has more 2.5 million observations or network traffic records/rows and 78 features or columns.

Process:

* EDA is used to visualize the data
* Statistics – to understand the data better, to know what is typical and what is not.
* Charts are used to visualize how the data is distributed in a data range.
* Median is the typical flow rate – it is not affected by both extreme and lower values
* After inspecting the data range, dropping the duplicates and filling out the missing values
* Correlation analysis is performed to understand which features are strongly correlated to each other – to mainly avoid duplication and focusing on features that truly adds value.
* Box plots are used to spot the outliers whereas histogram is used to understand the data distribution within the data range.
* Basically boxplots spots outliers in flow bytes and histogram shows the traffic volume.
* Data was sampled to reduce the computational costs and prototyping, this is an efficient way for future scaling.
* Outlier analysis:

1. Extreme values – either has normal with extreme or actually malicious
2. So mapping it with attack label is crucial to identify the patterns

* Performed logistic regression and support vector machine model with performance metrics.

Updates 🎉

* Need to implement the rest of the machine learning models.
* Deep dive into the dataset again
* Interpret the graphs better.
* Correlation matrices
* Moved ids env to the intrusion detection 2 folder and activated the environment

7

Implementing the rest of the models:

* Implemented random forest classifier with its visualization of performance metrics
* Implemented decision tree classifier with its visualization of performance metrics
* Implemented k nearest neighbour with its visualization of performance metrics
* The github repository for intrusion detection system using CICIDS 2017 dataset is completed, with a few code changes from my end.

April: 8th- 15th

9

[Anya Forger](mailto:reachanya03@gmail.com)

Check and download these datasets

1. [RAN Performance measurements for security threats - Mendeley Data](https://data.mendeley.com/datasets/t2rzh9y4mp/1)
2. [Edge-IIoTset Cyber Security Dataset of IoT & IIoT](https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot)
3. [ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-006-2023-QuasarX: This repository is a solution by team QuasarX for the ITU-ML5G-PS-006: Intrusion and Vulnerability Detection in Software-Defined Networks (SDN) challenge that ran from 2023-05-29 +03:00 to 2023-08-26 +03:00](https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-006-2023-QuasarX)

Also check this Python Library

[DropCorrelatedFeatures — 1.8.3](https://feature-engine.trainindata.com/en/latest/user_guide/selection/DropCorrelatedFeatures.html)

10

* Started performing a classification model on users vs bot dataset available on kaggle.
* Link to the dataset : [users vs bot](https://www.kaggle.com/datasets/juice0lover/users-vs-bots-classification)
* The goal is bot vs human classification – teaching a model to look at a profile and decide if it is a real person or an automated bot

11

Research paper 👍

Robust intrusion detection system with explainable artificial intelligence.

* ML is useful for detecting threats but incase of adversarial attacks, the ML model can be tricked by feeding crafted fake data (adversarial input) to make it fail and bypass the system as benign traffic.
* XAI - explainable ai - understand why the model took a certain decision. This field is focused on making ai models transparent and understandable to humans.
* It aims to answer a question – why did the model predict this output.
* The model is designed to basically work with any machine model – in the experimental setup IDS was developed using autoencoders. – it is trained on normal traffic only so it learns normal patterns, during testing if it detects something odd leading to high reconstruction error.
* Working –

1. Data collected during normal operation of the 5g ran, without attacks
2. Only benign rrc signaling features are used.
3. The autoencoder is trained to minimize reconstruction error for these features
4. After training it understands the structure of normal rrc signaling
5. During testing – the trained model is used to evaluate the real time traffic.
6. If the UE behaviour is abnormal, the autoencoder cannot reconstruct it accurately, thus leading to high reconstruction error.
7. And if the threshold increases, the input is flagged as anomalous.

* Why is an autoencoder a good fit? – we do not need labelled dataset for training
* It generalizes to unknown attacks or unseen patterns.
* Shap : SHapely additive exPlanations.
* Shap was an existing explainable xai tool, it was newly performed on o-ran.

12

O-RAN survey 👍

* O-ran stands for open radio access network
* It is a component which is used to connect user equipment and core network
* It was mainly developed to overcome the limitations of radio access network
* It has three main parts

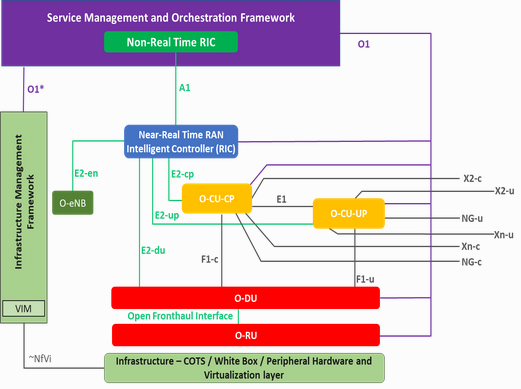
1. Distributed unit – which deals with lower layers of osi model
2. Ran intelligent controller – brain of the o ran which has ai/ml models integrated to make quick decisions. It has two subparts – 1. Near real time RIC 2. Non - real time RIC. it is a software defined platform, the programmability is done via xapps and rapps. It dynamically decides how the network behaves dynamically.
3. Centralized unit – handles higher layer protocols of the ran, manages user sessions and connects the radio network to the 5g core.
4. O ran architectural principles revolve around software defined networking, the key principles explained are

1. Disaggregation

2. Ran intelligent controller

3. Virtualization

4. Open interface.

1. 

13

Roadmap for integrating open RAN and AI for intrusion detection systems in 6G.

* It discussed the integration of O-RAN and artificial intelligence into 6g networks.
* AI for 6G : envisions a future where AI plays a pivotal role in securing and optimizing the operation of 6G networks.
* O-RAN consists of various components and open interfaces and ids is crucial for detecting malicious activities and vulnerabilities.
* Strategic placement : critical points to consider for the deployment – near real time RIC, open distributed units, open fronthaul interfaces.
* Federated learning for IDS – protecting data privacy like data traffic, private communication and device information.
* Strategic points for IDS deployment 👍

1. Near RT RIC

* Monitor e2 nodes and xapps
* Detects conflicts or malicious xapp behaviour

1. ODU AND OCU

* Monitor access attempts to protect the s-plane and c-plane

1. OFH

* Protects links between RU’S and DU’s

1. Xapp and rapp environments

* Detect misuse or overreach in control/data access

1. Management interfaces

* Prevent unauthorized configuration changes and Dos attempts

1. User plane

* Watch for rogue/malicious traffic and control messages

1. Integrated monitoring system

* A central analysis unit aggregates and analyzes alerts
* Coordinates system-wide response

1. Workflow

* Collect data – analyze with ml/rules – detect anomalies – alert or auto respond
* Suggested practice : blockchain + federated learning

1. Each ids sensor trains the model locally
2. Model updates are logged on a blockchain (immutability + transparency)
3. Updates are aggregated to form a global model
4. No raw data leaves the local node.

EXPERIMENTATION AND EVALUATION 👍

* Edge-IIoTset dataset was used. It consists of real attack data from iot and iiot environments – which are highly vulnerable and similar to o ran environments in complexity and distributed nature
* Features

1. 62 input features
2. It covers 15 different types of attacks

* Benign (normal traffic)
* DoS
* DDoS
* Man-in-the-Middle (MiTM)
* Reconnaissance
* Command Injection
* Backdoor\
* Password Attack
* Botnet
* Ransomware
* XSS
* SQL Injection
* Vulnerability Scan
* Spoofing
* Data Exfiltration

How federated learning is implemented 👍

* The model was trained in two different scenario using federated learning
* Scenario –1 : 5 clients were locally trained with their own dataset, each client sent gradients (model updates) to the central unit. The central unit aggregates all updates to refine the global model.
* Scenario – 2 : 10 clients were locally trained with their own dataset, each client sent gradients (model updates) to the central unit. The central unit aggregated all updates and refined the global model

1. Scenario –2 has better results compared to 1, the conclusion from this experimentation is, global model performs better when more clients are trained on the local dataset locally. It improves the global model performance, ultimately increasing the performance metrics. It performs just like the centralized learning.

14

AI driven network intrusion detection and resource allocation in real world O-RAN 5G networks.

* Anomaly traffic detector : IDS framework with dynamic resource allocation and user management within O-RAN architecture.
* It mitigates denial of service attack through an xapp that classifies network traffic in real time and dynamically adjusts network resources and user connections.
* The experimental evaluations show that the system effectively maintains low latency under attack conditions, doubled the throughput for legitimate users and reduces cpu usage by up to 15%.
* Dataset used – KDDCUP’99, the ml model was trained on this dataset (over 4 million for training and 3 lakhs for testing).
* The five features selected :

1. Protocol type
2. Service
3. Flag
4. Source bytes
5. Destination bytes

* The major objective is to DoS attack and allocate resources to legitimate users.
* Data preprocessing :

1. Label conversion : 0 and 1 (normal and any kind of attack)
2. Flag conversion :
3. Feature selection
4. Encoding and scaling

* Model training & evaluation :

1. Trained multiple ML models (random forest, one class SVM, LOF, KNN, autoencoder)
2. After evaluating accuracy, f1-score, precision, recall and inference speed
3. Random forest was selected for its high accuracy in classification attack vs normal.
4. Low training and prediction time.

* Detection logic : once the model is trained and deployed in an xapp

1. Real time packets are fed into the model
2. The model classifies each as 0 or 1
3. A burst of 1 from a single source, or a repeating attack pattern, triggers DoS detection logic.

* Dynamic resource allocation : the system dynamically allocates new resources to protect and prioritize normal users. (it will basically shift the bandwidth to legitimate users for a good throughput)

15

Feature engine : python library used in data science and machine learning for feature engineering. It makes pipelines cleaner, more reproducible and easier to debug.

How it can be helpful 👍

1. Transformers for common preprocessing tasks like

* Missing value imputation
* Encoding categorical variables
* Scaling numerical features
* Handling outliers

1. Scikit-learn compatible
2. Automates repetitive tasks
3. Model explainability

* To fill missing values with the median, we use:

from feature\_engine.imputation import MeanMedianImputer

* To fill missing values in categorical features, we use:

from feature\_engine.imputation import CategoricalImputer

* To encode rare labels (group infrequent categories), we use:

from feature\_engine.encoding import RareLabelEncoder

* To perform one-hot encoding on categorical features, we use:

from feature\_engine.encoding import OneHotEncoder

* To discretize continuous numerical features into bins, we use:

from feature\_engine.discretisation import EqualWidthDiscretiser

* To cap outliers in numerical features using the IQR method, we use:

from feature\_engine.outliers import Winsorizer

* To scale numerical features using a standard scaler, we use:

from feature\_engine.wrappers import SklearnTransformerWrapper

along with:

from sklearn.preprocessing import StandardScaler

* To drop highly correlated features, we use:

from feature\_engine.selection import DropCorrelatedFeatures

* To combine all steps into one reusable process, we use:

from sklearn.pipeline import Pipeline

April: 16th - 23rd

16

* Worked on making the PPT
* Going through the research papers for ppt

apr 17

* Worked on the ppt

apr 18

| **Model** | **Type** | **Used Datasets** | **Strengths** | **Notes** |
| --- | --- | --- | --- | --- |
| **SVM** (Support Vector Machine) | Traditional ML | NSL-KDD, UNSW-NB15 | Good for binary classification, small datasets | Can struggle with multiclass and large-scale data |
| **Random Forest (RF)** | Ensemble ML | NSL-KDD, CICIDS2017, UNSW-NB15 | High accuracy, handles imbalance | Slower for large feature sets |
| **KNN** (K-Nearest Neighbors) | Traditional ML | NSL-KDD, CICIDS2017 | Simple, interpretable | Slow at prediction, sensitive to noise |
| **Logistic Regression** | Traditional ML | NSL-KDD, TON\_IoT | Fast, explainable | Lower performance on complex patterns |
| **Naive Bayes** | Probabilistic ML | NSL-KDD, CICIDS2017 | Lightweight | Assumes feature independence |
| **LSTM** | Deep Learning (RNN) | CICIDS2017, TON\_IoT, Simulated RAN | Detects sequential/multi-step attacks | Needs lots of data and tuning |
| **GRU** | Deep Learning (RNN) | CICIDS2017, TON\_IoT | Similar to LSTM, but more efficient | Slightly lower performance than LSTM |
| **CNN** | Deep Learning | UNSW-NB15, CICIDS2017 | Good for spatial pattern detection | Not ideal for time-based attacks |
| **DNN (MLP)** | Deep Learning | NSL-KDD, CICIDS2017 | Flexible architecture | Requires tuning |
| **Autoencoder** | Unsupervised DL | TON\_IoT, CICIDS2017, Simulated | Detects unknown anomalies without labels | May produce false positives |
| **Isolation Forest** | Unsupervised ML | NSL-KDD, TON\_IoT, Open RAN Simulations | Good for anomaly detection | No training on attack data |
| **XGBoost** | Boosting ML | CICIDS2017, UNSW-NB15 | High performance, handles imbalance well | Needs tuning, compute-heavy |
| **GNN (GCN, GAT)** | Graph Neural Network | Simulated Open RAN (topology/UE-DU-RU graphs) | Captures relationships in node-based data | Needs structured graph input |
| **Hybrid/Ensemble** | Mixed | NSL-KDD, CICIDS2017, Custom RAN | Combine strengths of multiple models | Complex training, interpretability drops |

April: 24th - 30th

apr 23

apr 27

| FBS DETECTOR | EDGE - IIoT DATASET | KDDCUP 1999 DATASET |
| --- | --- | --- |

|  | Loading and cleaning the dataset   * Load the ml-edge\_iiot dataset for training the model * Check for missing, infinite values, dropping the duplicates. * Label encoding/ one-hot encoding if necessary for the categorical features   Feature scaling :   * Normalization/ standardization according to the needs * Feature selection using correlation analysis   Class balancing   * The dataset has major attack class, so smote can be used to oversample the normal traffic to train the models   train/test split   * The data is divided into training and testing sets.   Implementing the classic ml models   * Logistic regression, decision tree, random forest, KNN, naive bayes, XGBoost | Loading and cleaning the dataset.   * Load the dataset kddcup.data\_10\_percent.gz for training the model. * Check for missing or infinite values, dropping the duplicates. * Mapping labels to each class * One hot encode for categorical features   Feature scaling and normalization   * Manually dropping features based on domain knowledge * Standardization of the dataset for the linear models * Feature selection - using correlation analysis, feature importance (based on the dataset) * Feature extraction if necessary using PCA * Class weights or smote for balancing the dataset   Implementing the classic ml models   * Logistic regression, decision tree, random forest, KNN. naive bayes, SVM, XGBoost   And can experiment with ensemble learning techniques. |
| --- | --- | --- |

May: 1st - 7th