Configuration Manual

DDoS Defence in IoMT: A Hybrid CNN-LSTM approach for SNORT based Intrusion Detection

Misha Rose Kambakaran Mathew
Student ID: 22159851

School of Computing
National College of Ireland

Supervisor: Joel Aleburu

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Misha Rose Kambakaran Mathew		
Student ID:	22159851		
Programme:	MSc Cybersecurity	**Year:**	2024
Module:	Research Project		
Lecturer:	Joel Aleburu		
Submission Due Date:	16/09/2024		
Project Title:	DDoS Defence in IoMT: A Hybrid CNN-LSTM approach for SNORT based Intrusion Detection		
Word Count:	1340 **Page Count:** 23		

Signature:	Misha Rose Kambakaran Mathew
Date:	16/09/2024

Configuration Manual

Misha Rose Kambakaran Mathew
Student ID: 22159851

1 Introduction

This configuration manual contains the information regarding tools, software and technologies used for hybrid CNN-LSTM approach for SNORT based intrusion detection. This guide includes sections where section 2 contains system specification describing the details of hardware and operation system, section 3 list the software tools and libraries used in software specification, section 4 list out the steps for the configuration of deep learning, section 5 include the instructions for the configuration of Virtual machines and snort, section 6 gives a detailed steps of deep learning procedures and steps 7 explain the steps for deploying the model and generation of SNORT rule.

2 System Specification

The experiment was conducted on personal computer on which the experiment setup was made, and implementation was done. Specification of the system as follows,

- HP SPECTRE x360
- RAM:16 GB
- System Type-64-bit OS, x64-based processor
- Operating System: Windows 11
- Experiment setup: Windows 11, Anaconda.Navigator, JupyterLab, VM ware, Ubuntu 24.04, Kali Linux 2024.2, SNORT VERSION 2.9.20, Python 3.9.12

3 Software Specification

This section discusses the software used to build this model.

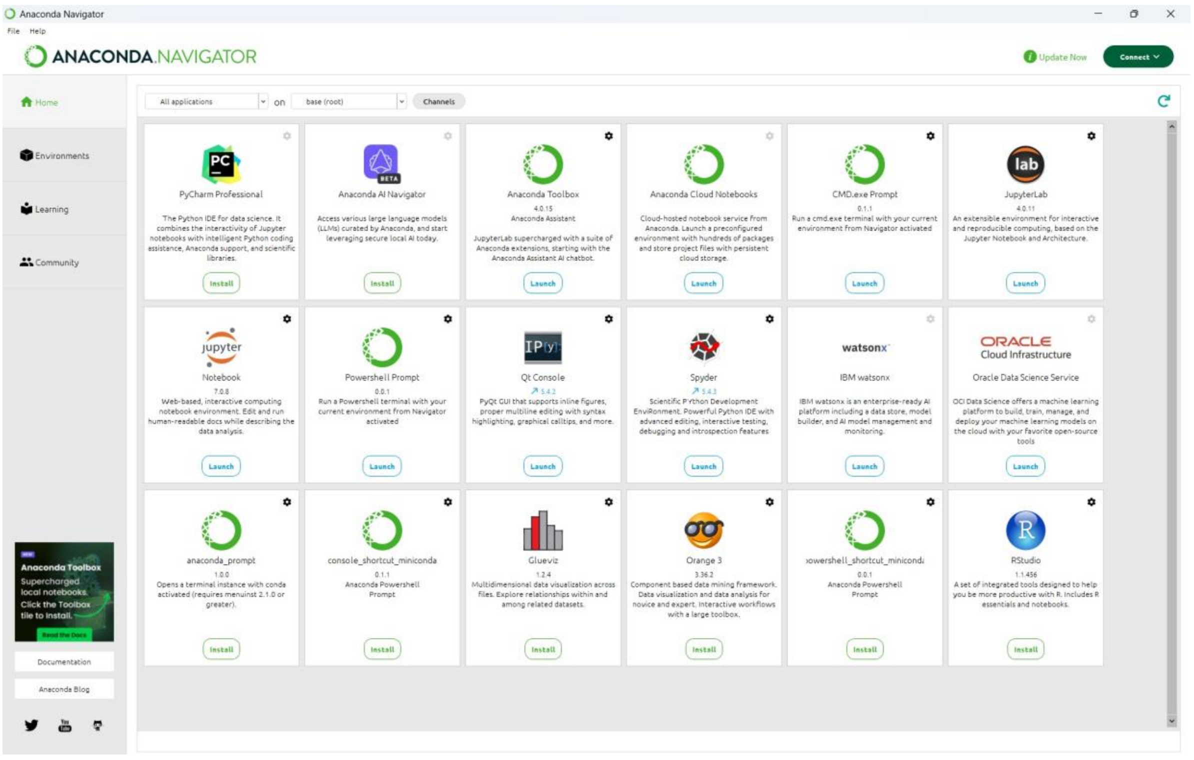
- **Anaconda** is used to do the deep learning; it manages the packages and environment.
- **Jupyter Notebook**: The python code for the deep learning is written here, which is an open-source web application to create live code, equations and visualizations implementation.
- **Python 3.9.12**: The programming language used for coding the model and to preprocess the data. Pythons consist of wide range of libraries which helps in deep learning tasks.
- **Pandas**: Python Library for data manipulation and analysis, provide data frames to analyse the structured data.
- **NumPy**: Library for numerical computation in Python, support arrays, matrices and mathematical functions to operate on data.
- **Scikit-learn** mainly used for data preprocessing, model evaluation and metrics calculation, it is an efficient tool for data mining and analysis.

- **TensorFlow**: Library for numerical computation and creation of complex deep learning model. This framework mainly used for building and training CNN-LSTM model.
- **Scapy**: It is an interactive packet manipulation program for packet generation, network discovery and network attacks. Used for creating custom network traffic for testing.
- **VMware**: Allows to run multiple operating system on single machine providing flexible environment for testing and deployment. SNORT installation, model deployment and testing are done using virtual machine setup.
- **Ubuntu**: Base operating system for servers and development environments, SNORT configuration, traffic capture and snort rule generation is carried out here.
- **Kali Linux**: Debian based Linux distribution aiming advanced penetration testing and secure auditing. In this project it is mainly used for generating customized traffic and sending packets to the SNORT machine.

4 Configuration for Deep Learning

1. Download and install Anaconda2.6.0
2. Open Jupyter notebook with Python version and create new notebook in '. pynb' file.

ANACONDA NAVIGATOR



PyCharm Professional
The Python IDE for data science. It combines the interactivity of Jupyter notebooks with intelligent Python coding assistance, Anaconda support, and scientific libraries.
Install

Anaconda AI Navigator
Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.
Install

Anaconda Toolbox
4.0.15
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.
Launch

Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.
Launch

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch

JupyterLab
4.0.11
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
7.0.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch

Qt Console
↗ 5.4.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Spyder
↗ 5.4.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

IBM watsonx
IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.
Launch

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools
Launch

anaconda_prompt
1.0.0
Opens a terminal instance with conda activated (requires menuinst 2.1.0 or greater).
Install

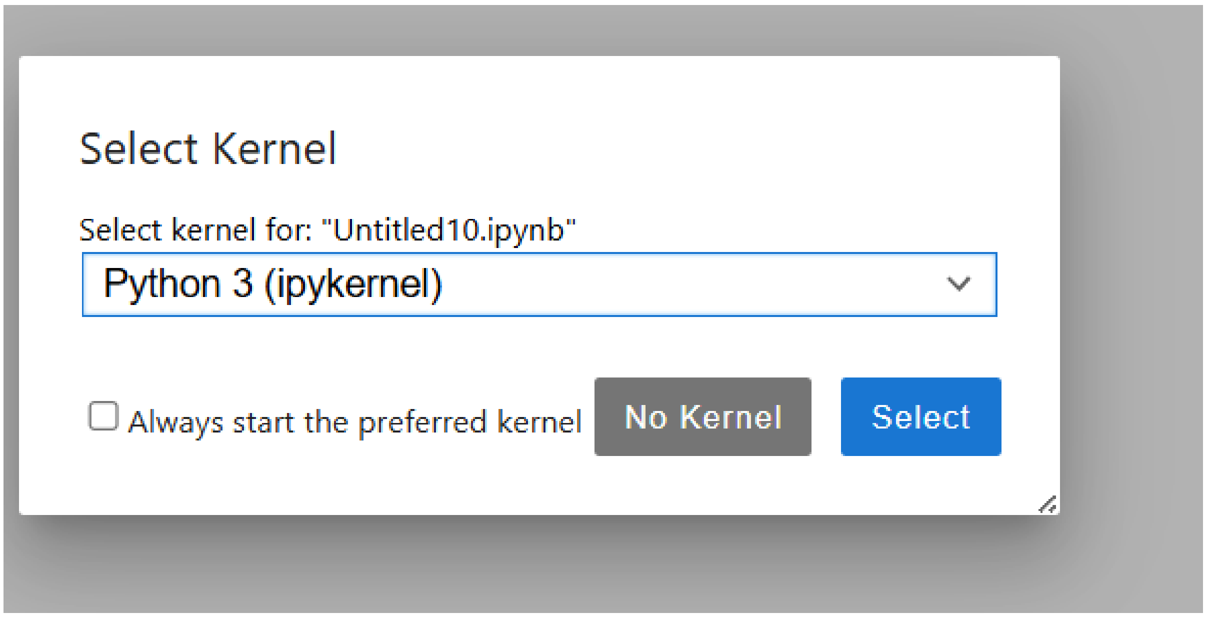
console_shortcut_miniconda
0.1.1
Anaconda Powershell Prompt
Install

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

Orange 3
3.36.2
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

›owershell_shortcut_minicond‹
0.0.1
Anaconda Powershell Prompt
Install

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install



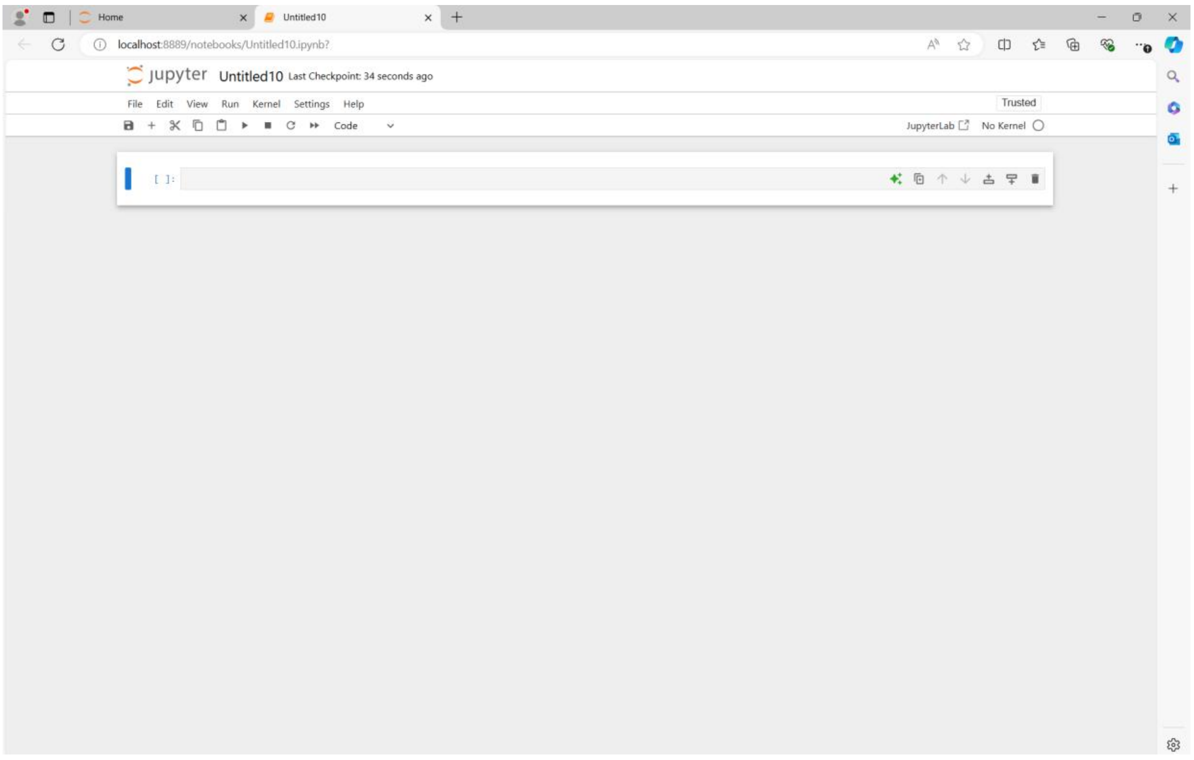
Select Kernel

Select kernel for: "Untitled10.ipynb"

Python 3 (ipykernel)

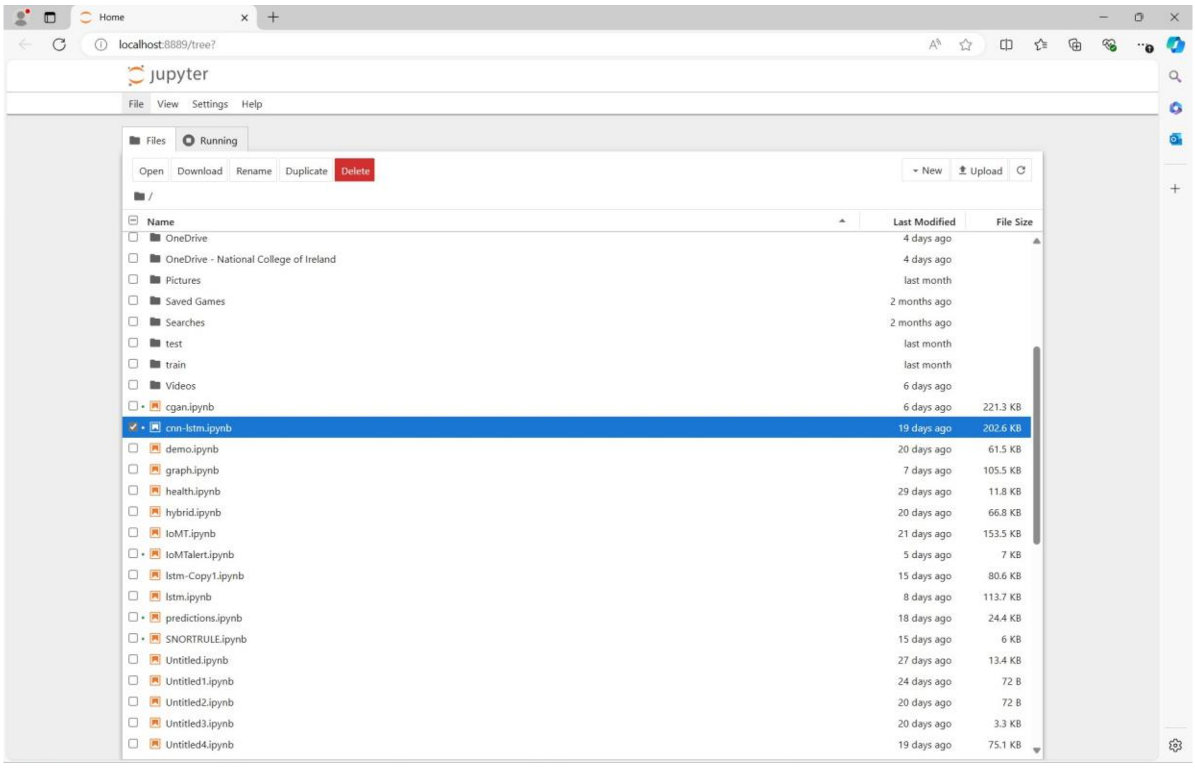
Always start the preferred kernel

Jupyter Notebook



localhost:8889/notebooks/Untitled10.ipynb?

jupyter Untitled10 Last Checkpoint: 34 seconds ago

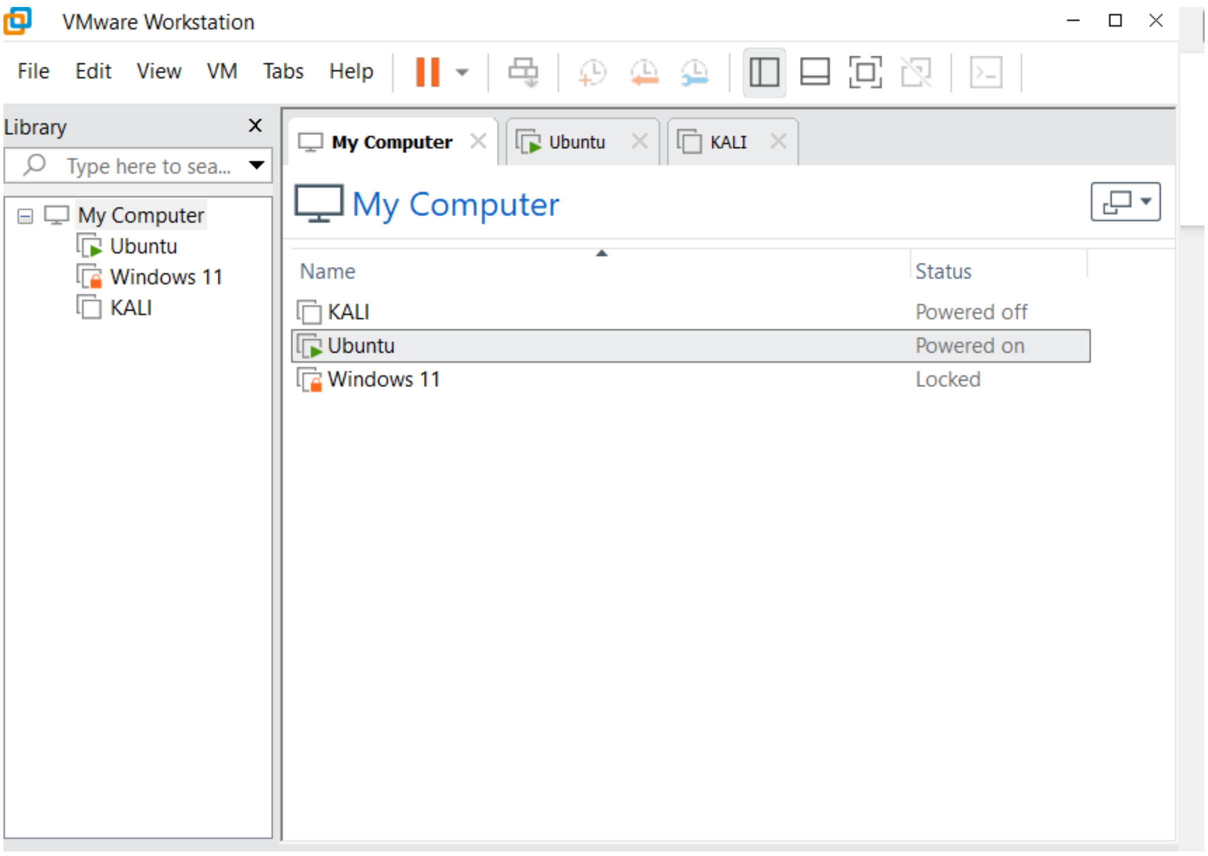


Name	Last Modified	File Size
OneDrive	4 days ago	
OneDrive - National College of Ireland	4 days ago	
Pictures	last month	
Saved Games	2 months ago	
Searches	2 months ago	
test	last month	
train	last month	
Videos	6 days ago	
cgan.ipynb	6 days ago	221.3 KB
cnn-lstm.ipynb	19 days ago	202.6 KB
demo.ipynb	20 days ago	61.5 KB
graph.ipynb	7 days ago	105.5 KB
health.ipynb	29 days ago	11.8 KB
hybrid.ipynb	20 days ago	66.8 KB
IoMT.ipynb	21 days ago	153.5 KB
IoMTalert.ipynb	5 days ago	7 KB
lstm-Copy1.ipynb	15 days ago	80.6 KB
lstm.ipynb	8 days ago	113.7 KB
predictions.ipynb	18 days ago	24.4 KB
SNORTRULE.ipynb	15 days ago	6 KB
Untitled.ipynb	27 days ago	13.4 KB
Untitled1.ipynb	24 days ago	72 B
Untitled2.ipynb	20 days ago	72 B
Untitled3.ipynb	20 days ago	3.3 KB
Untitled4.ipynb	19 days ago	75.1 KB

5 Configuration of Virtual Machines and SNORT

1. Download and install VMware
2. Setup an environment for SNORT configuration and testing
3. Download and install Kali Linux with NAT network
4. Download and Install Ubuntu 24.04 with Nat network.
5. Install SNORT packages using the command: sudo apt-get install snort and check the version using snort – version inside Ubuntu
6. Use the ‘nautilus’ command to obtain the manage file permissions.
7. Open the snort.conf file from the path /etc/snort/snort.conf inside ubuntu in otherlocations and make a copy of this file for retrieve the data in case of loss.

VMWARE

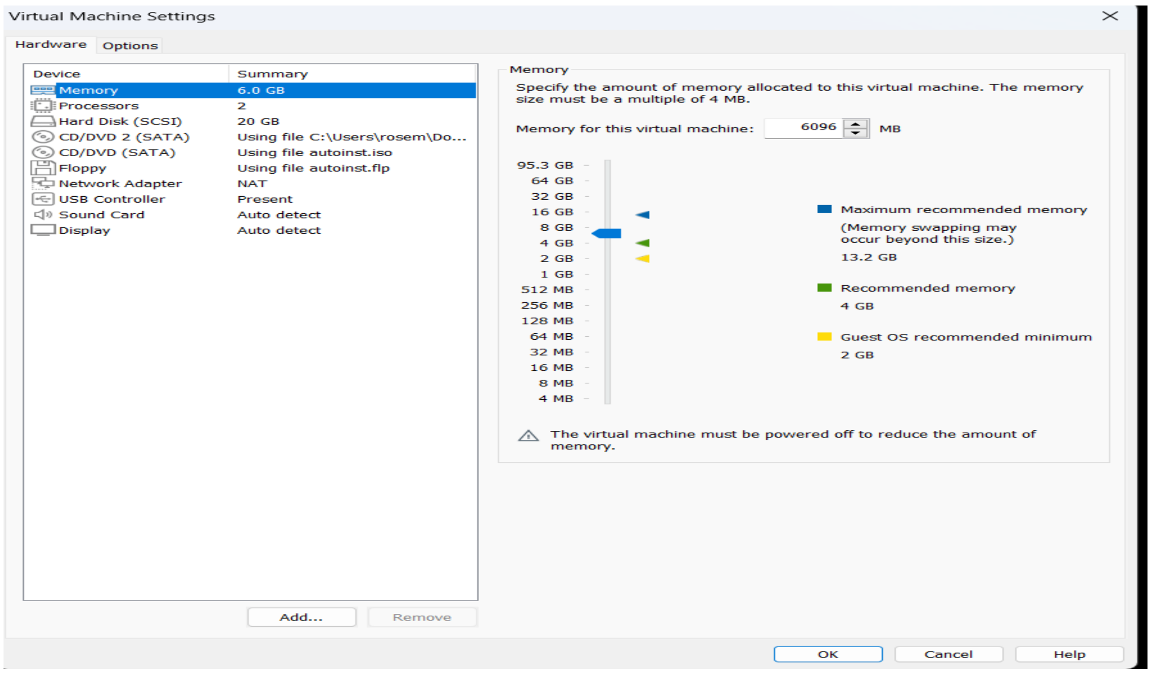


My Computer
- Ubuntu
- Windows 11
- KALI

My Computer

Name	Status
KALI	Powered off
Ubuntu	Powered on
Windows 11	Locked

Ubuntu settings



Device	Summary
Memory	6.0 GB
Processors	2
Hard Disk (SCSI)	20 GB
CD/DVD 2 (SATA)	Using file C:\Users\rosem\Do...
CD/DVD (SATA)	Using file autoinst.iso
Floppy	Using file autoinst.flp
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Display	Auto detect

Memory

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

Memory for this virtual machine: 6096 MB

95.3 GB
64 GB
32 GB
16 GB
8 GB
4 GB
2 GB
1 GB
512 MB
256 MB
128 MB
64 MB
32 MB
16 MB
8 MB
4 MB

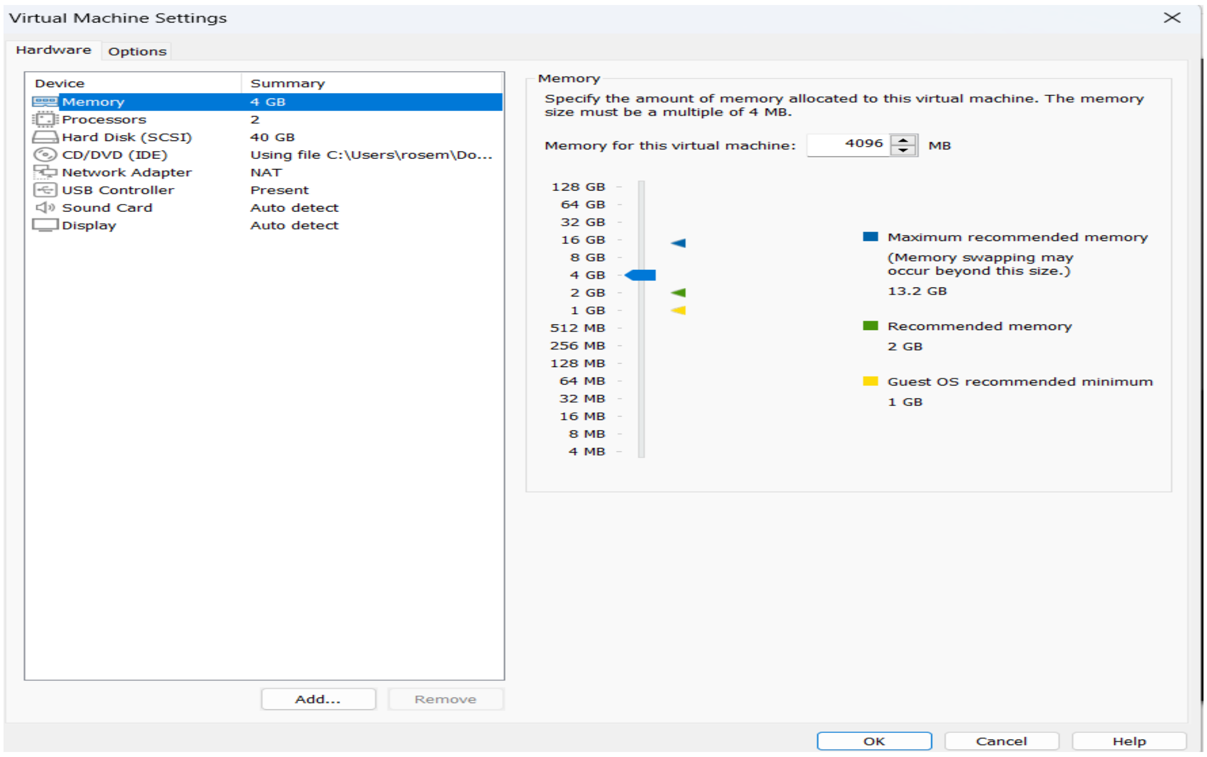
🟦 Maximum recommended memory
(Memory swapping may occur beyond this size.)
13.2 GB

🟩 Recommended memory
4 GB

🟨 Guest OS recommended minimum
2 GB

⚠ The virtual machine must be powered off to reduce the amount of memory.

Kali Linux Settings



Virtual Machine Settings

Hardware | Options

```
Device               Summary
Memory               4 GB
Processors           2
Hard Disk (SCSI)     40 GB
CD/DVD (IDE)         Using file C:\Users\rosem\Do...
Network Adapter      NAT
USB Controller       Present
Sound Card           Auto detect
Display              Auto detect
```

Memory

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

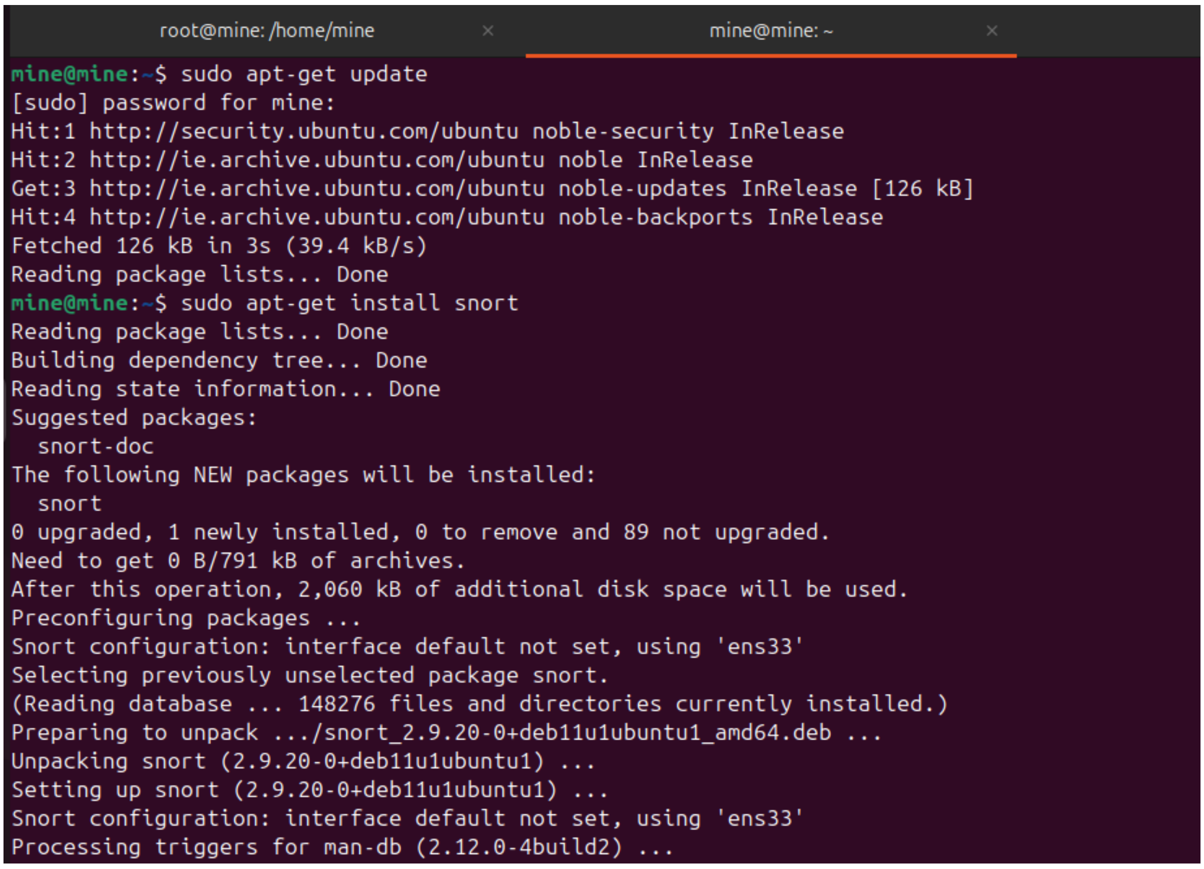
Memory for this virtual machine: 4096 MB

🟦 Maximum recommended memory (Memory swapping may occur beyond this size.) 13.2 GB

🟩 Recommended memory 2 GB

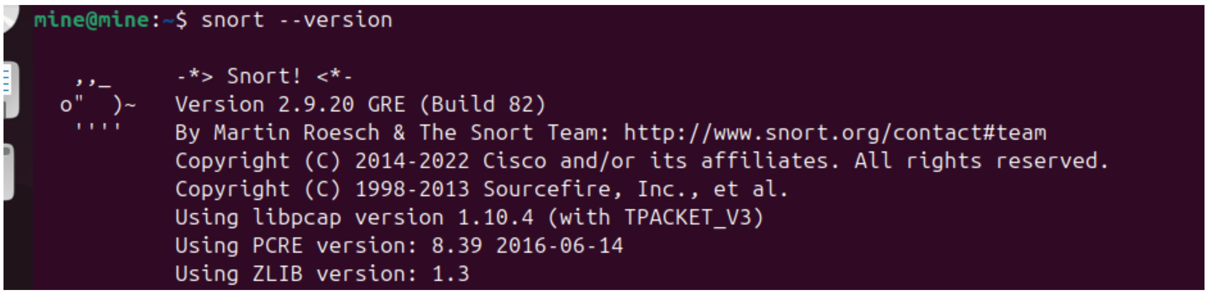
🟨 Guest OS recommended minimum 1 GB

SNORT INSTALLATION

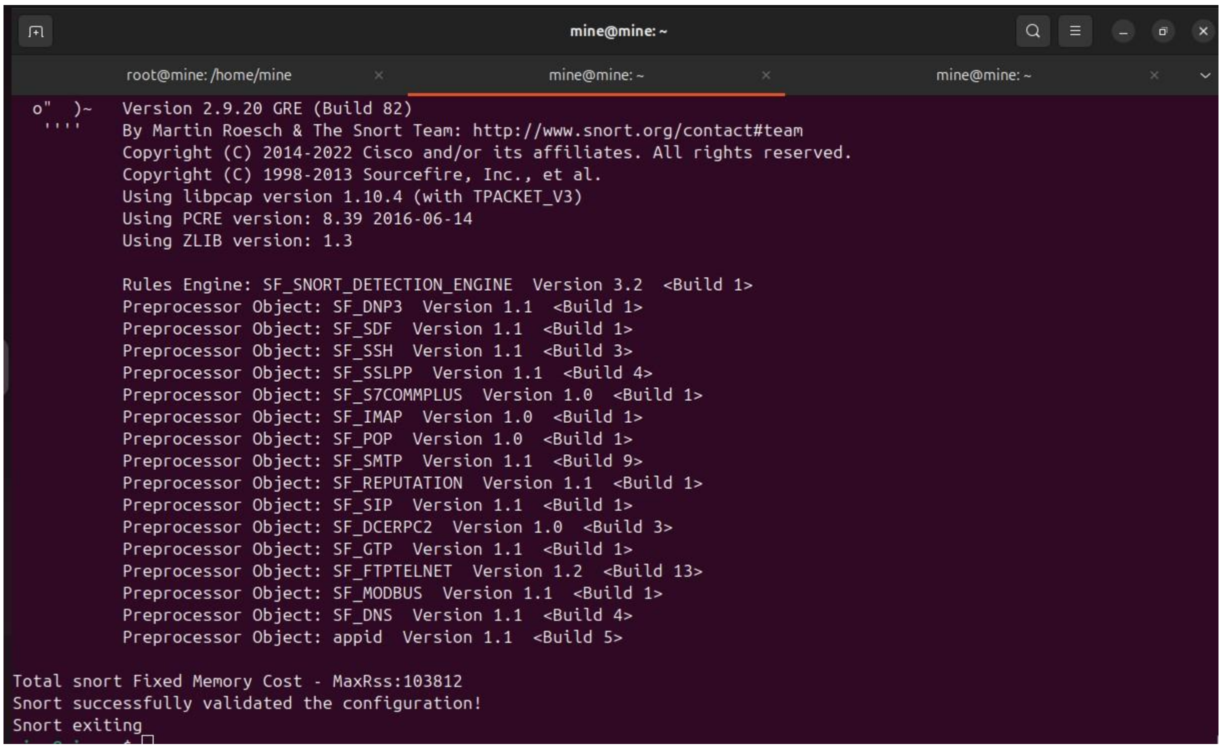


```
mine@mine:~$ sudo apt-get update
[sudo] password for mine:
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://ie.archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://ie.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:4 http://ie.archive.ubuntu.com/ubuntu noble-backports InRelease
Fetched 126 kB in 3s (39.4 kB/s)
Reading package lists... Done
mine@mine:~$ sudo apt-get install snort
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  snort-doc
The following NEW packages will be installed:
  snort
0 upgraded, 1 newly installed, 0 to remove and 89 not upgraded.
Need to get 0 B/791 kB of archives.
After this operation, 2,060 kB of additional disk space will be used.
Preconfiguring packages ...
Snort configuration: interface default not set, using 'ens33'
Selecting previously unselected package snort.
(Reading database ... 148276 files and directories currently installed.)
Preparing to unpack .../snort_2.9.20-0+deb11u1ubuntu1_amd64.deb ...
Unpacking snort (2.9.20-0+deb11u1ubuntu1) ...
Setting up snort (2.9.20-0+deb11u1ubuntu1) ...
Snort configuration: interface default not set, using 'ens33'
Processing triggers for man-db (2.12.0-4build2) ...
```

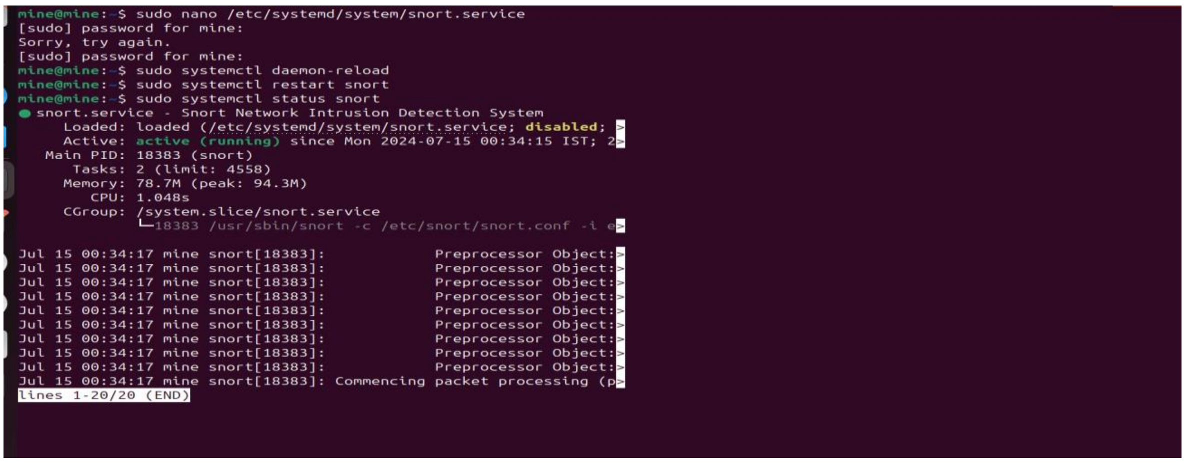
SNORT VERSION



```
mine@mine:~$ snort --version

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.3
```

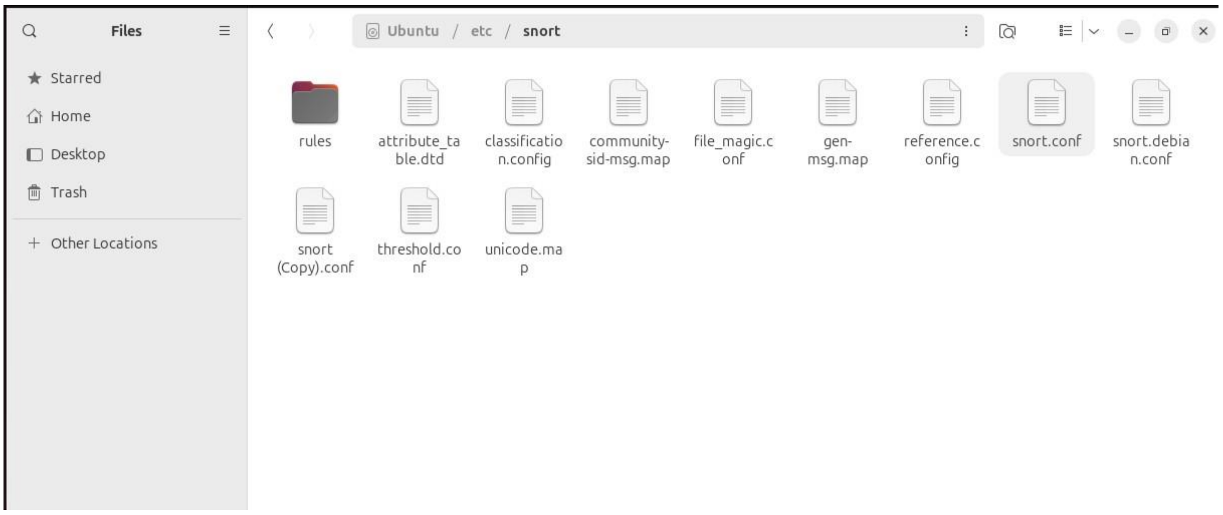


```
  o"  )~   Version 2.9.20 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.2  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: appid  Version 1.1  <Build 5>

Total snort Fixed Memory Cost - MaxRss:103812
Snort successfully validated the configuration!
Snort exiting
```



```
mine@mine:~$ sudo nano /etc/systemd/system/snort.service
[sudo] password for mine:
Sorry, try again.
[sudo] password for mine:
mine@mine:~$ sudo systemctl daemon-reload
mine@mine:~$ sudo systemctl restart snort
mine@mine:~$ sudo systemctl status snort
● snort.service - Snort Network Intrusion Detection System
     Loaded: loaded (/etc/systemd/system/snort.service; disabled; >
     Active: active (running) since Mon 2024-07-15 00:34:15 IST; 2>
   Main PID: 18383 (snort)
      Tasks: 2 (limit: 4558)
     Memory: 78.7M (peak: 94.3M)
        CPU: 1.048s
     CGroup: /system.slice/snort.service
             └─18383 /usr/sbin/snort -c /etc/snort/snort.conf -i e>

Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]:             Preprocessor Object:>
Jul 15 00:34:17 mine snort[18383]: Commencing packet processing (p>
lines 1-20/20 (END)
```

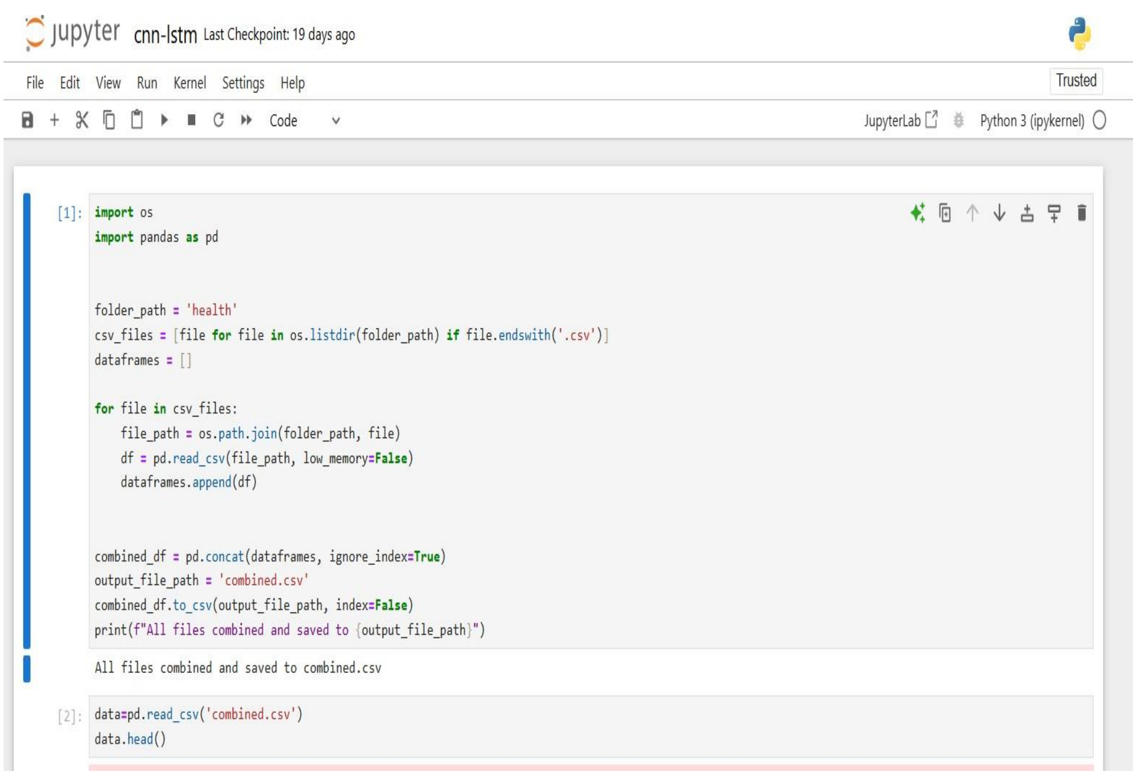
snort.conf FILE



6 Deep Learning Procedures

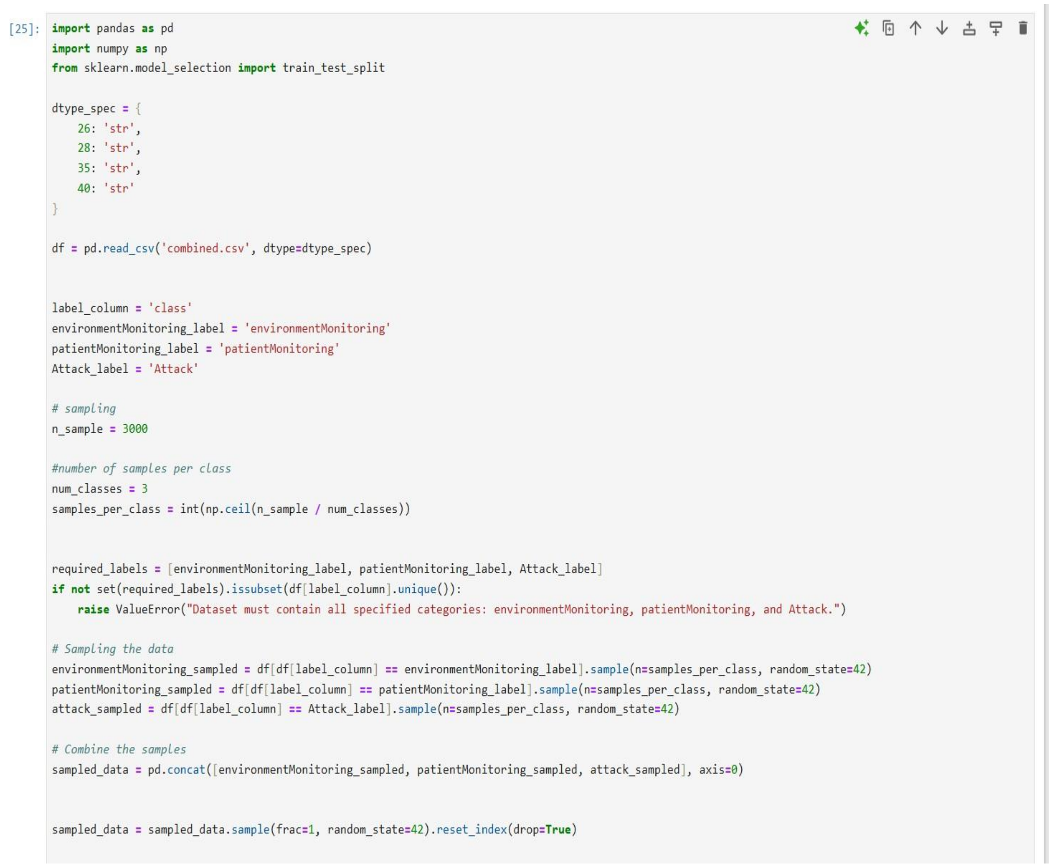
1. Download the dataset required for the deep learning.
2. Import the necessary libraries
3. Load and combine the dataset into a single csv file.
4. Sample the dataset to create balance subset with equal number of samples from the selected class and save the sampled dataset into a single csv file.
5. Preprocess the data, by defining available features, separating numeric and non-numeric features, preprocess numeric features using standardscaler () and non-numeric features using one-hot encoder and then combine processed features.
6. Encode the targeted label and split dataset into test train sets, later reshape the data to fit into the expected input shape of neural network.
7. Define CNN-LSTM model and train the hybrid model
8. Evaluate the CNN-LSTM model using metrices like accuracy, precision, recall, F-1 score, ROC AUC
9. Evaluate combined model using classification report, accuracy, precision, recall and ROC AUC.
10. Calculate Malicious Score to measure how perfectly the model identify the attacks.
11. Visualize confusion matrix to study distribution of true positives, false positives, true negatives and false negatives which will help to assess the model performance.
12. Compute and plot ROC curve
13. Compute and plot Precision-Recall curve

Loading the dataset

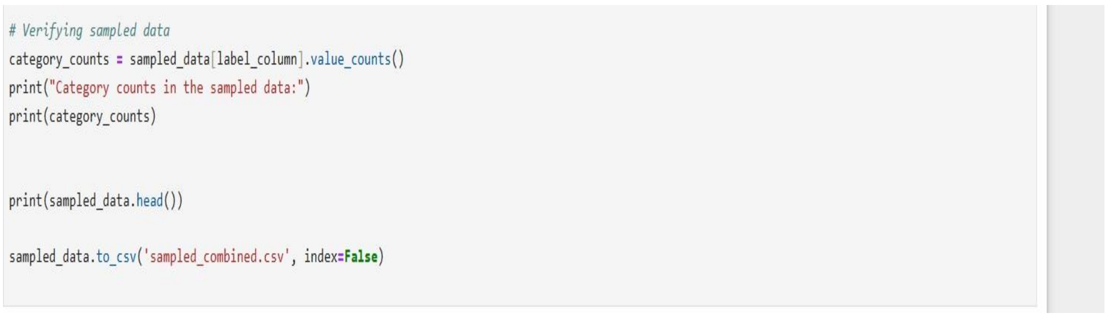

jupyter cnn-lstm Last Checkpoint: 19 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[1]: import os
     import pandas as pd


     folder_path = 'health'
     csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')]
     dataframes = []

     for file in csv_files:
         file_path = os.path.join(folder_path, file)
         df = pd.read_csv(file_path, low_memory=False)
         dataframes.append(df)


     combined_df = pd.concat(dataframes, ignore_index=True)
     output_file_path = 'combined.csv'
     combined_df.to_csv(output_file_path, index=False)
     print(f"All files combined and saved to {output_file_path}")

     All files combined and saved to combined.csv

[2]: data=pd.read_csv('combined.csv')
     data.head()
```

DATA SAMPLING

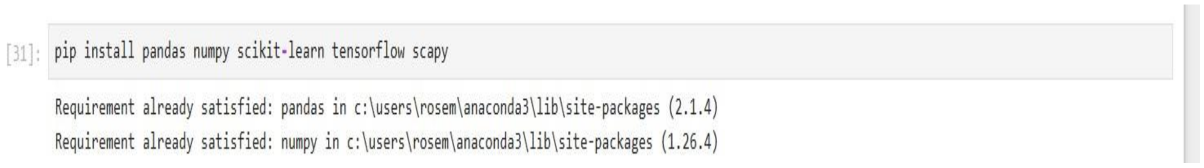


```
[25]: import pandas as pd
      import numpy as np
      from sklearn.model_selection import train_test_split

      dtype_spec = {
          26: 'str',
          28: 'str',
          35: 'str',
          40: 'str'
      }

      df = pd.read_csv('combined.csv', dtype=dtype_spec)


      label_column = 'class'
      environmentMonitoring_label = 'environmentMonitoring'
      patientMonitoring_label = 'patientMonitoring'
      Attack_label = 'Attack'

      # sampling
      n_sample = 3000

      #number of samples per class
      num_classes = 3
      samples_per_class = int(np.ceil(n_sample / num_classes))


      required_labels = [environmentMonitoring_label, patientMonitoring_label, Attack_label]
      if not set(required_labels).issubset(df[label_column].unique()):
          raise ValueError("Dataset must contain all specified categories: environmentMonitoring, patientMonitoring, and Attack.")

      # Sampling the data
      environmentMonitoring_sampled = df[df[label_column] == environmentMonitoring_label].sample(n=samples_per_class, random_state=42)
      patientMonitoring_sampled = df[df[label_column] == patientMonitoring_label].sample(n=samples_per_class, random_state=42)
      attack_sampled = df[df[label_column] == Attack_label].sample(n=samples_per_class, random_state=42)

      # Combine the samples
      sampled_data = pd.concat([environmentMonitoring_sampled, patientMonitoring_sampled, attack_sampled], axis=0)


      sampled_data = sampled_data.sample(frac=1, random_state=42).reset_index(drop=True)
```

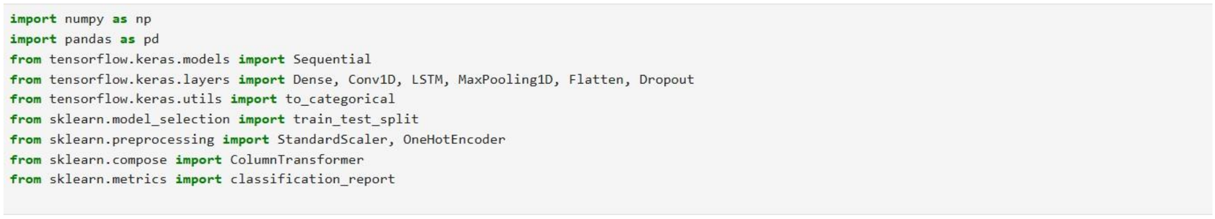


```
# Verifying sampled data
category_counts = sampled_data[label_column].value_counts()
print("Category counts in the sampled data:")
print(category_counts)


print(sampled_data.head())

sampled_data.to_csv('sampled_combined.csv', index=False)
```

Installation of Necessary Libraries

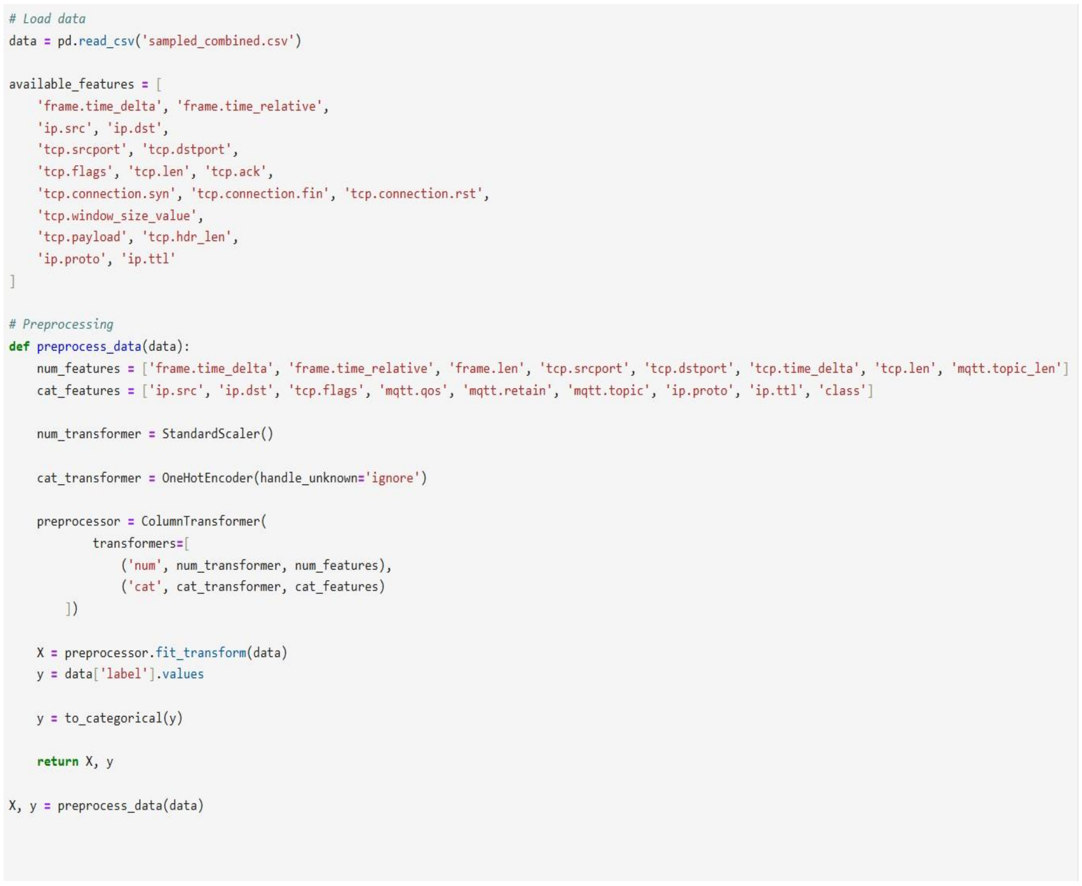


```
[31]: pip install pandas numpy scikit-learn tensorflow scapy

      Requirement already satisfied: pandas in c:\users\rosem\anaconda3\lib\site-packages (2.1.4)
      Requirement already satisfied: numpy in c:\users\rosem\anaconda3\lib\site-packages (1.26.4)
```



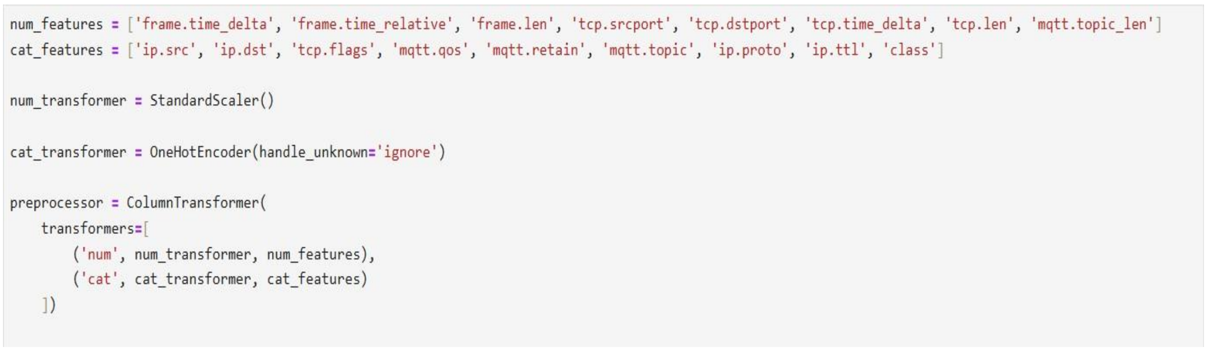
```
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, LSTM, MaxPooling1D, Flatten, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import classification_report
```

Data Preprocessing

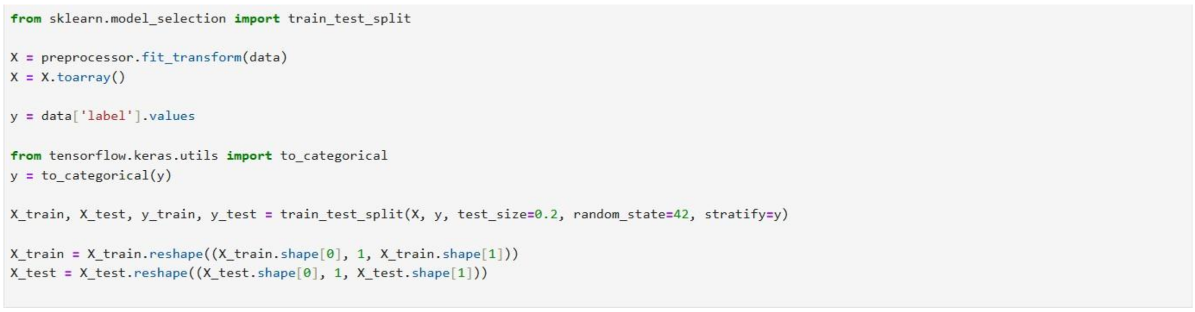


```
# Load data
data = pd.read_csv('sampled_combined.csv')

available_features = [
    'frame.time_delta', 'frame.time_relative',
    'ip.src', 'ip.dst',
    'tcp.srcport', 'tcp.dstport',
    'tcp.flags', 'tcp.len', 'tcp.ack',
    'tcp.connection.syn', 'tcp.connection.fin', 'tcp.connection.rst',
    'tcp.window_size_value',
    'tcp.payload', 'tcp.hdr_len',
    'ip.proto', 'ip.ttl'
]

# Preprocessing
def preprocess_data(data):
    num_features = ['frame.time_delta', 'frame.time_relative', 'frame.len', 'tcp.srcport', 'tcp.dstport', 'tcp.time_delta', 'tcp.len', 'mqtt.topic_len']
    cat_features = ['ip.src', 'ip.dst', 'tcp.flags', 'mqtt.qos', 'mqtt.retain', 'mqtt.topic', 'ip.proto', 'ip.ttl', 'class']

    num_transformer = StandardScaler()

    cat_transformer = OneHotEncoder(handle_unknown='ignore')

    preprocessor = ColumnTransformer(
            transformers=[
                ('num', num_transformer, num_features),
                ('cat', cat_transformer, cat_features)
        ])

    X = preprocessor.fit_transform(data)
    y = data['label'].values

    y = to_categorical(y)

    return X, y

X, y = preprocess_data(data)
```

Data Reshaping and Splitting

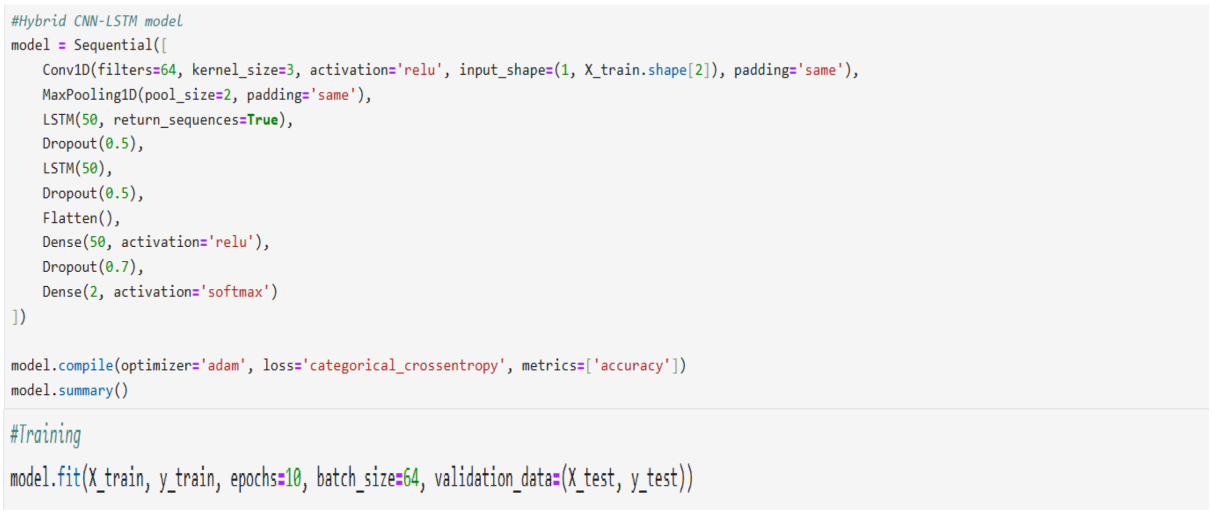


```
num_features = ['frame.time_delta', 'frame.time_relative', 'frame.len', 'tcp.srcport', 'tcp.dstport', 'tcp.time_delta', 'tcp.len', 'mqtt.topic_len']
cat_features = ['ip.src', 'ip.dst', 'tcp.flags', 'mqtt.qos', 'mqtt.retain', 'mqtt.topic', 'ip.proto', 'ip.ttl', 'class']

num_transformer = StandardScaler()

cat_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_features),
        ('cat', cat_transformer, cat_features)
    ])
```



```
from sklearn.model_selection import train_test_split

X = preprocessor.fit_transform(data)
X = X.toarray()

y = data['label'].values

from tensorflow.keras.utils import to_categorical
y = to_categorical(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
```

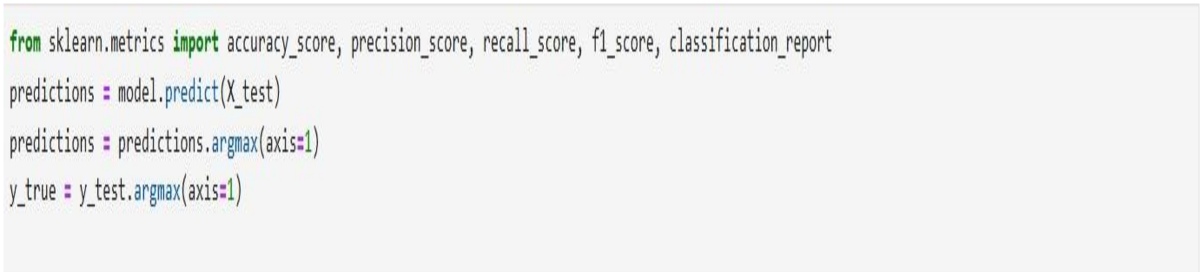
CNN-LSTM model



```
#Hybrid CNN-LSTM model
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(1, X_train.shape[2]), padding='same'),
    MaxPooling1D(pool_size=2, padding='same'),
    LSTM(50, return_sequences=True),
    Dropout(0.5),
    LSTM(50),
    Dropout(0.5),
    Flatten(),
    Dense(50, activation='relu'),
    Dropout(0.7),
    Dense(2, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
#Training
model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test, y_test))
```

Model Prediction



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
predictions = model.predict(X_test)
predictions = predictions.argmax(axis=1)
y_true = y_test.argmax(axis=1)
```

Evaluation

```
#Evaluation
accuracy = accuracy_score(y_true, predictions)
precision = precision_score(y_true, predictions, average='macro')
recall = recall_score(y_true, predictions, average='macro')
f1 = f1_score(y_true, predictions, average='macro')

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

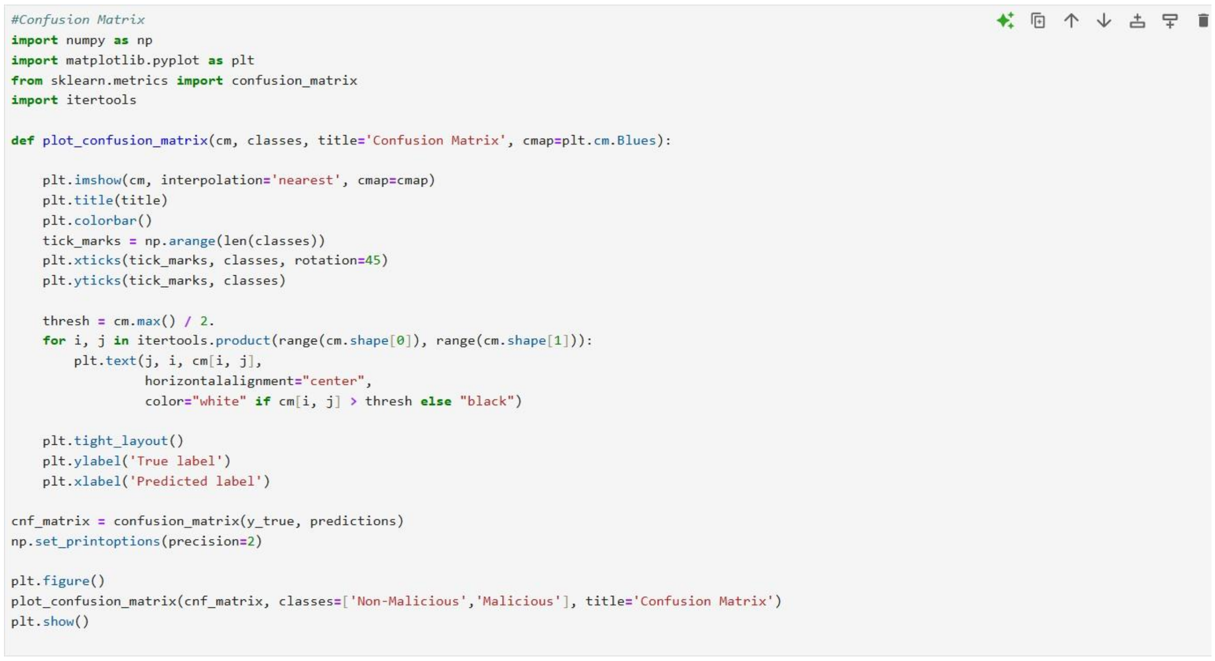
Classification Report

```
#Prediction and Classification Report
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

predictions = model.predict(X_test)
predictions = predictions.argmax(axis=1)
y_true = y_test.argmax(axis=1) if y_test.ndim > 1 else y_test
print("\nClassification Report:")
print(classification_report(y_true, predictions))
```

Malicious Score

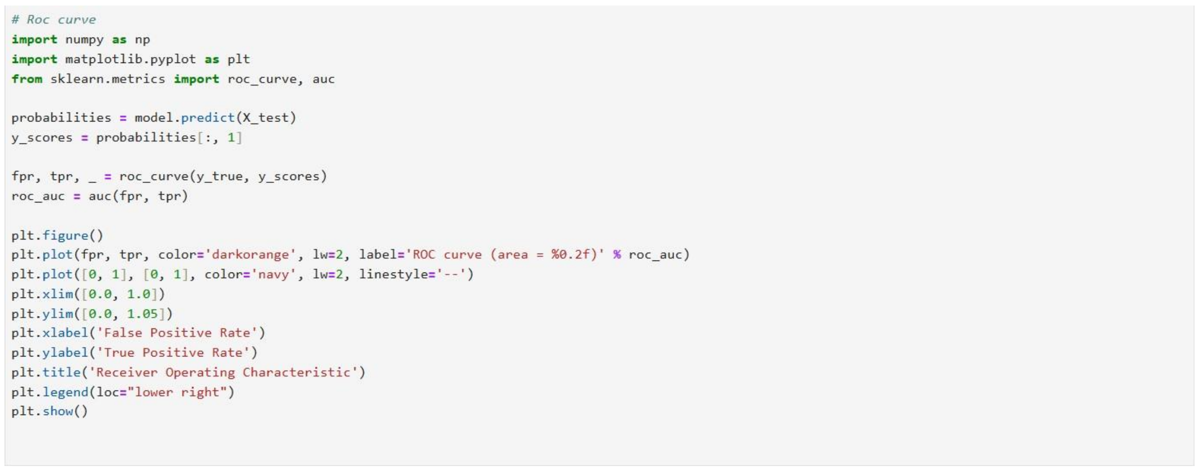
```
#malicious score
probabilities = model.predict(X_test)
predictions = probabilities.argmax(axis=1)

threshold = 0.5
predicted_classes = (probabilities[:, 1] > threshold).astype(int)
if predicted_classes.sum() > 0:
    malicious_score = np.mean(probabilities[:, 1][predicted_classes == 1])
else:
    malicious_score = 0

print(f"Malicious Score: {malicious_score:.4f}")
```

Confusion Matrix



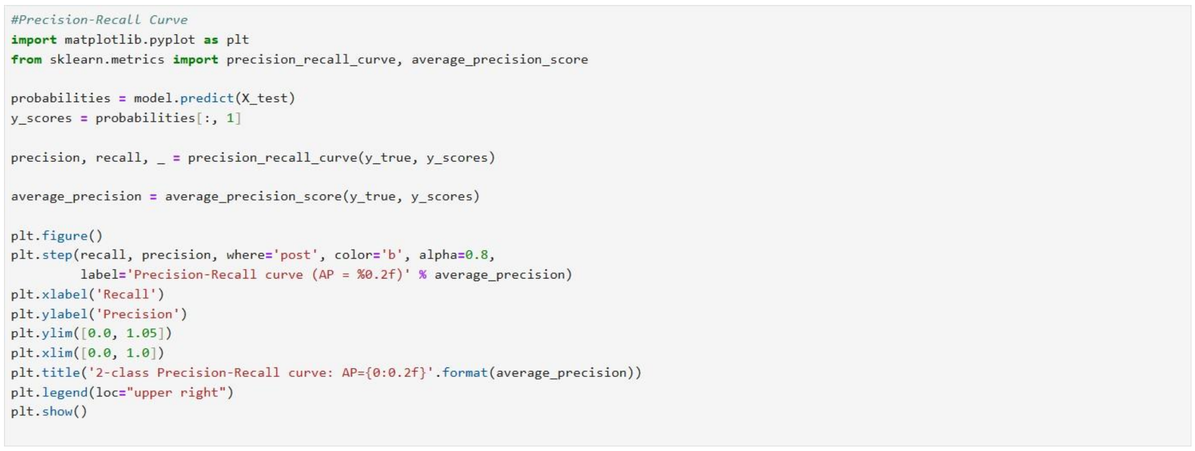
```python
#Confusion Matrix
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools

def plot_confusion_matrix(cm, classes, title='Confusion Matrix', cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cnf_matrix = confusion_matrix(y_true, predictions)
np.set_printoptions(precision=2)

plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['Non-Malicious','Malicious'], title='Confusion Matrix')
plt.show()
```

ROC curve



```python
# Roc curve
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

probabilities = model.predict(X_test)
y_scores = probabilities[:, 1]

fpr, tpr, _ = roc_curve(y_true, y_scores)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

Precision -Recall Curve



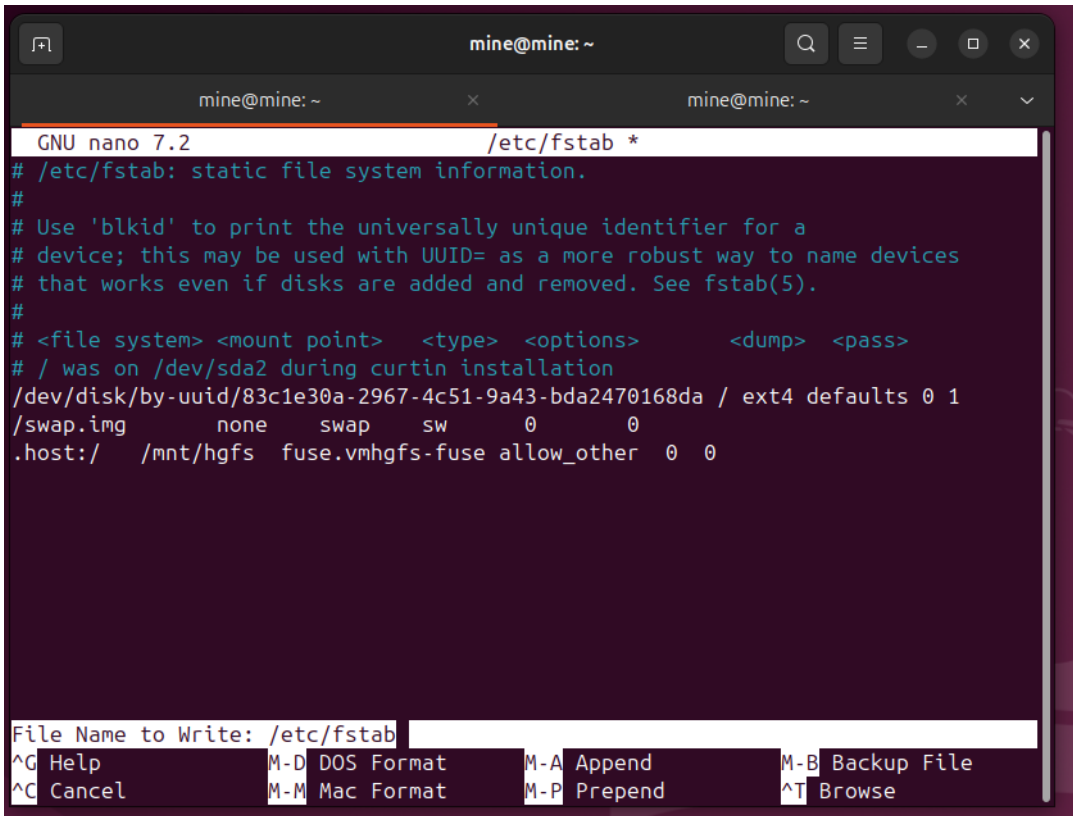
```python
#Precision-Recall Curve
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve, average_precision_score

probabilities = model.predict(X_test)
y_scores = probabilities[:, 1]

precision, recall, _ = precision_recall_curve(y_true, y_scores)

average_precision = average_precision_score(y_true, y_scores)

plt.figure()
plt.step(recall, precision, where='post', color='b', alpha=0.8,
         label='Precision-Recall curve (AP = %0.2f)' % average_precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(average_precision))
plt.legend(loc="upper right")
plt.show()
```

```python
# Save the combined model
combined_model.save('combined_model.h5')
print("Combined model saved as 'combined_model.h5'.")
```

7 Procedures for model deployment and generation of Snort Rule

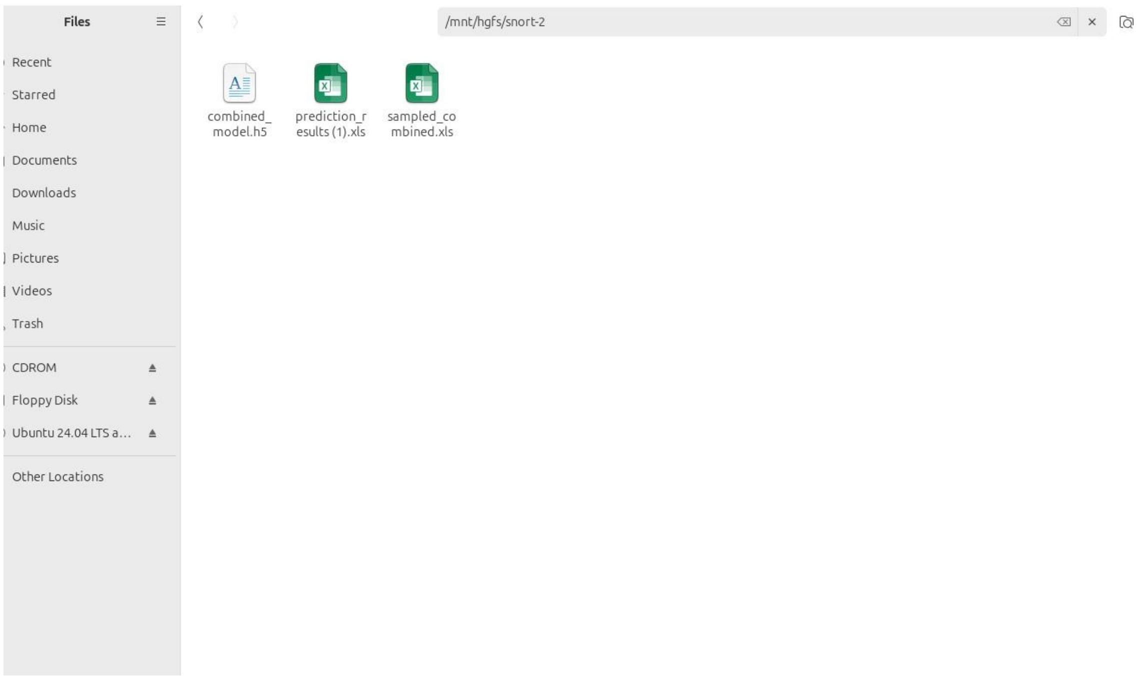
1. Transfer the saved model to the VM ware by installing open-vm-tools packages which include the folder sharing, drag and drop and clipboard sharing functionalities. Use the command ‘sudo apt install open-vm-tools open-vm-tools-desktop’.
2. Create a directory for the shared folder and mount this folder which allow VM to access directory on the host machine using ‘sudo mount -t fuse. vmhgfs-fuse .host:/shared folder.
3. Use /etc/fstab file to automatically mount filesystems at boot time.
4. Transfer the saved model by the command ‘cp /path/to/combined_model.h5 /path/shared folder/on/host.
5. Setup an environment by installing python and its libraries in Ubuntu.
6. Crete a python script for loading the model, preprocess incoming data, make predictions and generating snort rules.
7. The generated SNORT rule will be added to local.rules inside /etc/snort/rules
8. Open the snort.conf file using ’sudo nano /etc/snort/snort.conf’ and check whether the local.rules are included.
9. SNORT will start listening to network packages using ‘sudo snort -A console -c /etc/snort/snort.conf.

fstab file edition

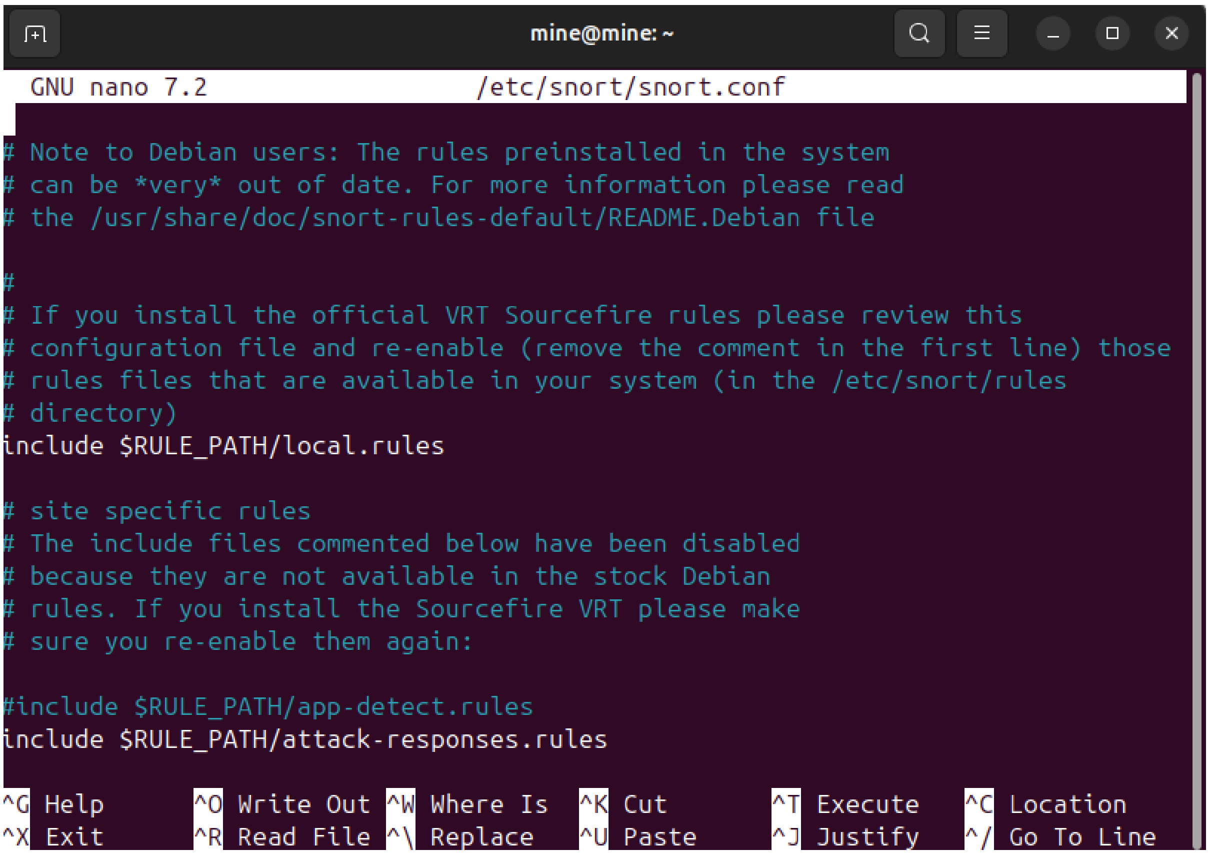


```
  GNU nano 7.2                        /etc/fstab *
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/83c1e30a-2967-4c51-9a43-bda2470168da / ext4 defaults 0 1
/swap.img       none    swap    sw      0       0
.host:/   /mnt/hgfs  fuse.vmhgfs-fuse allow_other  0  0









File Name to Write: /etc/fstab
^G Help           M-D DOS Format      M-A Append          M-B Backup File
^C Cancel         M-M Mac Format      M-P Prepend         ^T Browse
```

Transferring the saved model

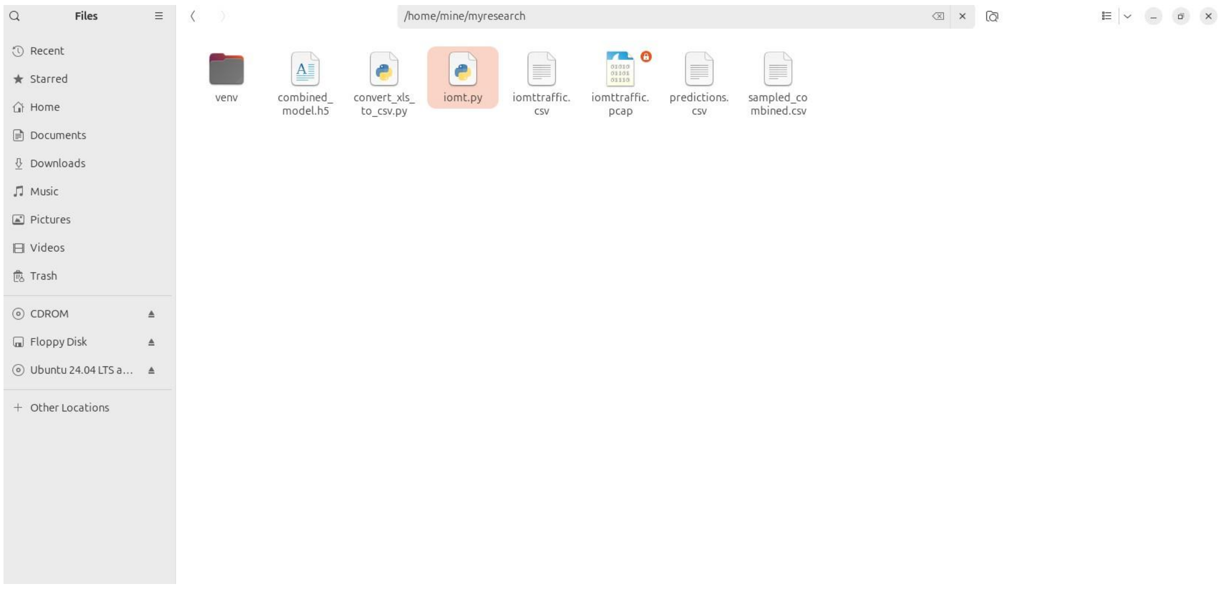


Editing snort.conf file

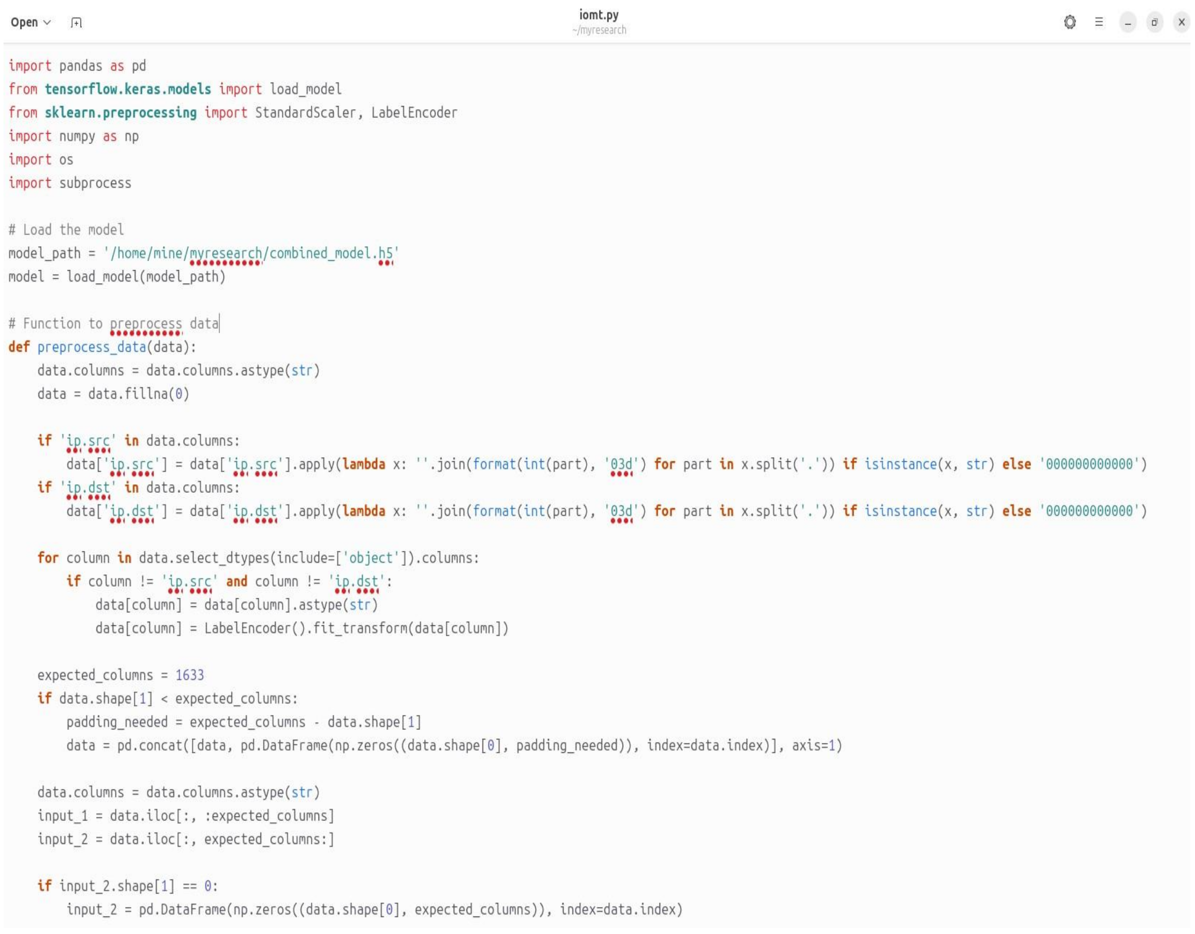


```
  GNU nano 7.2                    /etc/snort/snort.conf

# Note to Debian users: The rules preinstalled in the system
# can be *very* out of date. For more information please read
# the /usr/share/doc/snort-rules-default/README.Debian file

#
# If you install the official VRT Sourcefire rules please review this
# configuration file and re-enable (remove the comment in the first line) those
# rules files that are available in your system (in the /etc/snort/rules
# directory)
include $RULE_PATH/local.rules

# site specific rules
# The include files commented below have been disabled
# because they are not available in the stock Debian
# rules. If you install the Sourcefire VRT please make
# sure you re-enable them again:

#include $RULE_PATH/app-detect.rules
include $RULE_PATH/attack-responses.rules

^G Help        ^O Write Out ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit        ^R Read File ^\ Replace   ^U Paste      ^J Justify    ^/ Go To Line
```

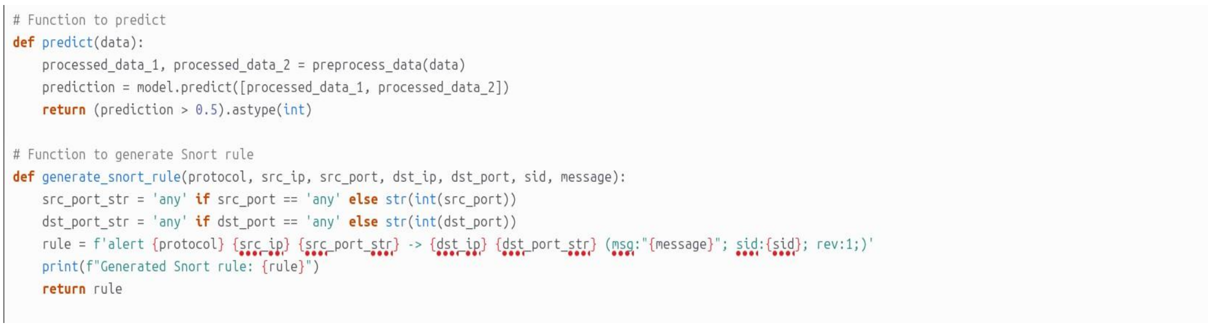
Creating iomt.py files



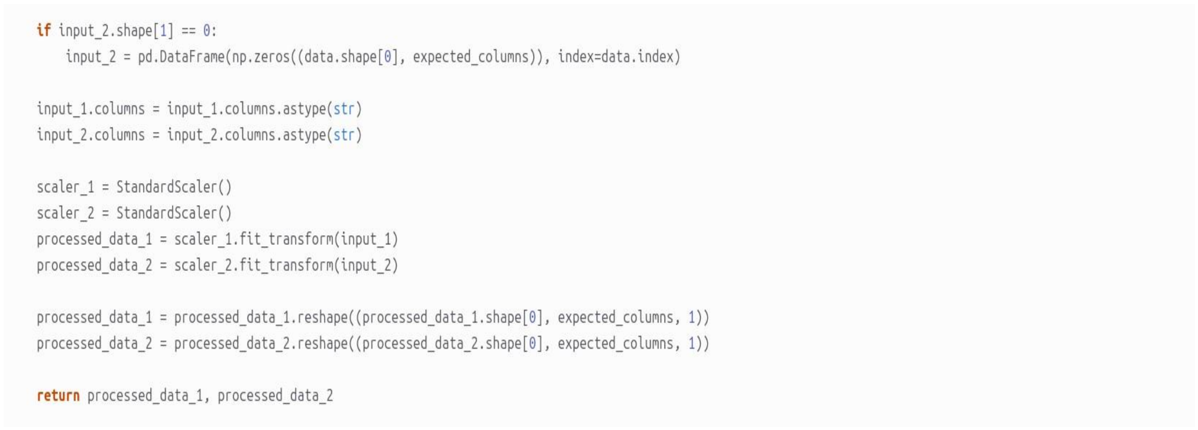
Python Script for prediction and SNORT rule generation



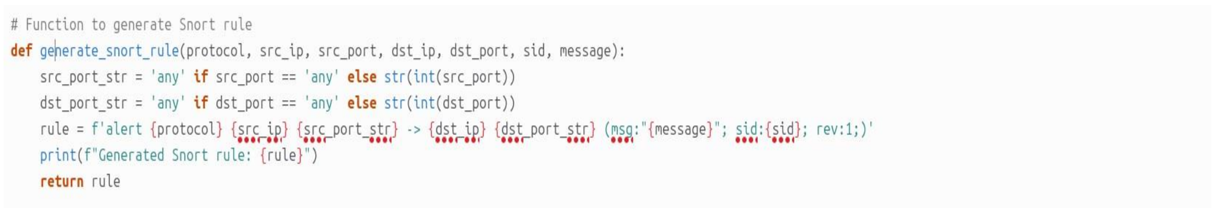
```python
import pandas as pd
from tensorflow.keras.models import load_model
from sklearn.preprocessing import StandardScaler, LabelEncoder
import numpy as np
import os
import subprocess

# Load the model
model_path = '/home/mine/myresearch/combined_model.h5'
model = load_model(model_path)

# Function to preprocess data
def preprocess_data(data):
    data.columns = data.columns.astype(str)
    data = data.fillna(0)

    if 'ip.src' in data.columns:
        data['ip.src'] = data['ip.src'].apply(lambda x: ''.join(format(int(part), '03d') for part in x.split('.')) if isinstance(x, str) else '000000000000')
    if 'ip.dst' in data.columns:
        data['ip.dst'] = data['ip.dst'].apply(lambda x: ''.join(format(int(part), '03d') for part in x.split('.')) if isinstance(x, str) else '000000000000')

    for column in data.select_dtypes(include=['object']).columns:
        if column != 'ip.src' and column != 'ip.dst':
            data[column] = data[column].astype(str)
            data[column] = LabelEncoder().fit_transform(data[column])

    expected_columns = 1633
    if data.shape[1] < expected_columns:
        padding_needed = expected_columns - data.shape[1]
        data = pd.concat([data, pd.DataFrame(np.zeros((data.shape[0], padding_needed)), index=data.index)], axis=1)

    data.columns = data.columns.astype(str)
    input_1 = data.iloc[:, :expected_columns]
    input_2 = data.iloc[:, expected_columns:]

    if input_2.shape[1] == 0:
        input_2 = pd.DataFrame(np.zeros((data.shape[0], expected_columns)), index=data.index)
```



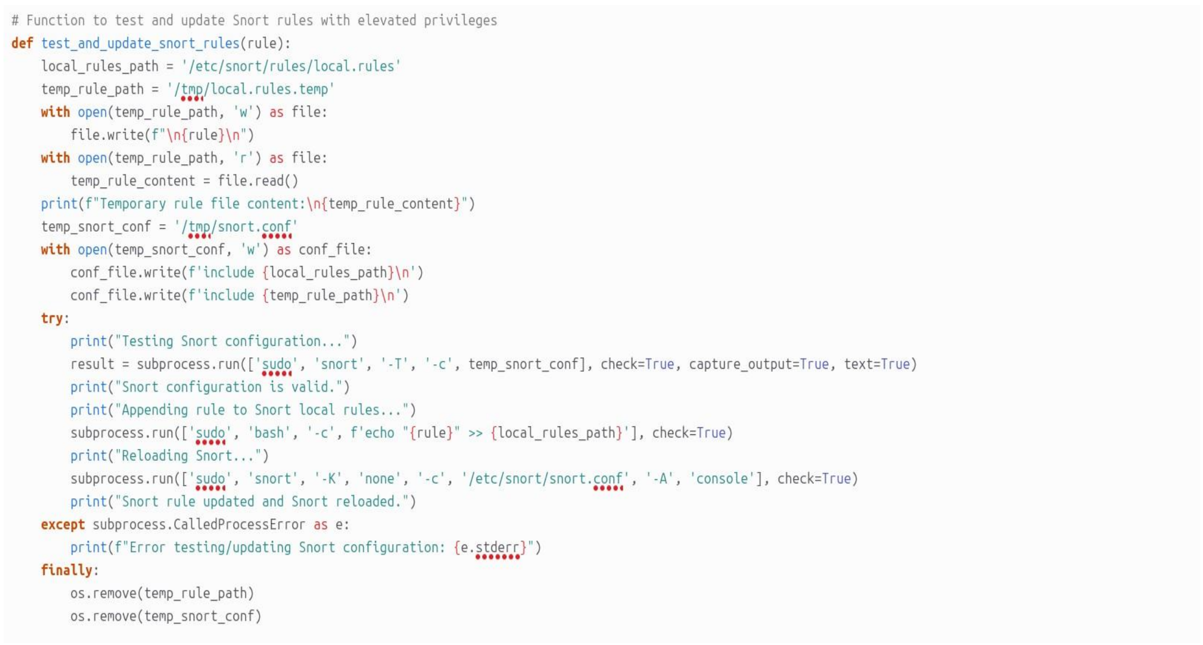
```python
# Function to predict
def predict(data):
    processed_data_1, processed_data_2 = preprocess_data(data)
    prediction = model.predict([processed_data_1, processed_data_2])
    return (prediction > 0.5).astype(int)

# Function to generate Snort rule
def generate_snort_rule(protocol, src_ip, src_port, dst_ip, dst_port, sid, message):
    src_port_str = 'any' if src_port == 'any' else str(int(src_port))
    dst_port_str = 'any' if dst_port == 'any' else str(int(dst_port))
    rule = f'alert {protocol} {src_ip} {src_port_str} -> {dst_ip} {dst_port_str} (msg:"{message}"; sid:{sid}; rev:1;)'
    print(f"Generated Snort rule: {rule}")
    return rule
```



```python
    if input_2.shape[1] == 0:
        input_2 = pd.DataFrame(np.zeros((data.shape[0], expected_columns)), index=data.index)

    input_1.columns = input_1.columns.astype(str)
    input_2.columns = input_2.columns.astype(str)

    scaler_1 = StandardScaler()
    scaler_2 = StandardScaler()
    processed_data_1 = scaler_1.fit_transform(input_1)
    processed_data_2 = scaler_2.fit_transform(input_2)

    processed_data_1 = processed_data_1.reshape((processed_data_1.shape[0], expected_columns, 1))
    processed_data_2 = processed_data_2.reshape((processed_data_2.shape[0], expected_columns, 1))

    return processed_data_1, processed_data_2
```



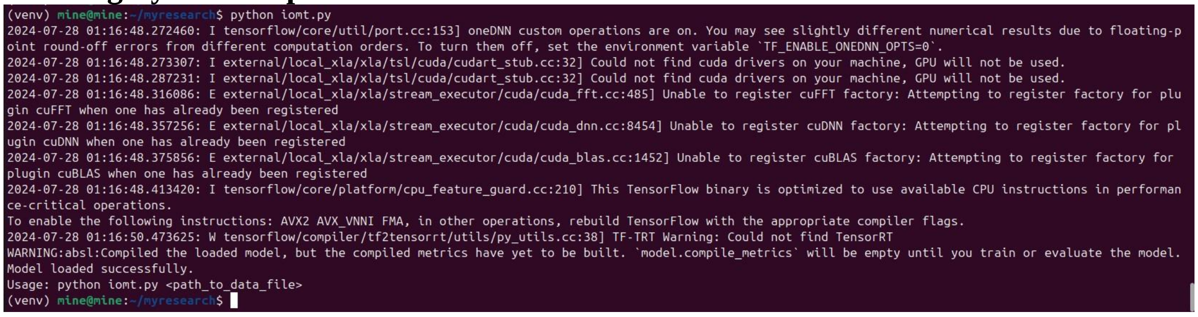
```python
# Function to generate Snort rule
def generate_snort_rule(protocol, src_ip, src_port, dst_ip, dst_port, sid, message):
    src_port_str = 'any' if src_port == 'any' else str(int(src_port))
    dst_port_str = 'any' if dst_port == 'any' else str(int(dst_port))
    rule = f'alert {protocol} {src_ip} {src_port_str} -> {dst_ip} {dst_port_str} (msg:"{message}"; sid:{sid}; rev:1;)'
    print(f"Generated Snort rule: {rule}")
    return rule
```



```python
# Function to test and update Snort rules with elevated privileges
def test_and_update_snort_rules(rule):
    local_rules_path = '/etc/snort/rules/local.rules'
    temp_rule_path = '/tmp/local.rules.temp'
    with open(temp_rule_path, 'w') as file:
        file.write(f"\n{rule}\n")
    with open(temp_rule_path, 'r') as file:
        temp_rule_content = file.read()
    print(f"Temporary rule file content:\n{temp_rule_content}")
    temp_snort_conf = '/tmp/snort.conf'
    with open(temp_snort_conf, 'w') as conf_file:
        conf_file.write(f'include {local_rules_path}\n')
        conf_file.write(f'include {temp_rule_path}\n')
    try:
        print("Testing Snort configuration...")
        result = subprocess.run(['sudo', 'snort', '-T', '-c', temp_snort_conf], check=True, capture_output=True, text=True)
        print("Snort configuration is valid.")
        print("Appending rule to Snort local rules...")
        subprocess.run(['sudo', 'bash', '-c', f'echo "{rule}" >> {local_rules_path}'], check=True)
        print("Reloading Snort...")
        subprocess.run(['sudo', 'snort', '-K', 'none', '-c', '/etc/snort/snort.conf', '-A', 'console'], check=True)
        print("Snort rule updated and Snort reloaded.")
    except subprocess.CalledProcessError as e:
        print(f"Error testing/updating Snort configuration: {e.stderr}")
    finally:
        os.remove(temp_rule_path)
        os.remove(temp_snort_conf)
```

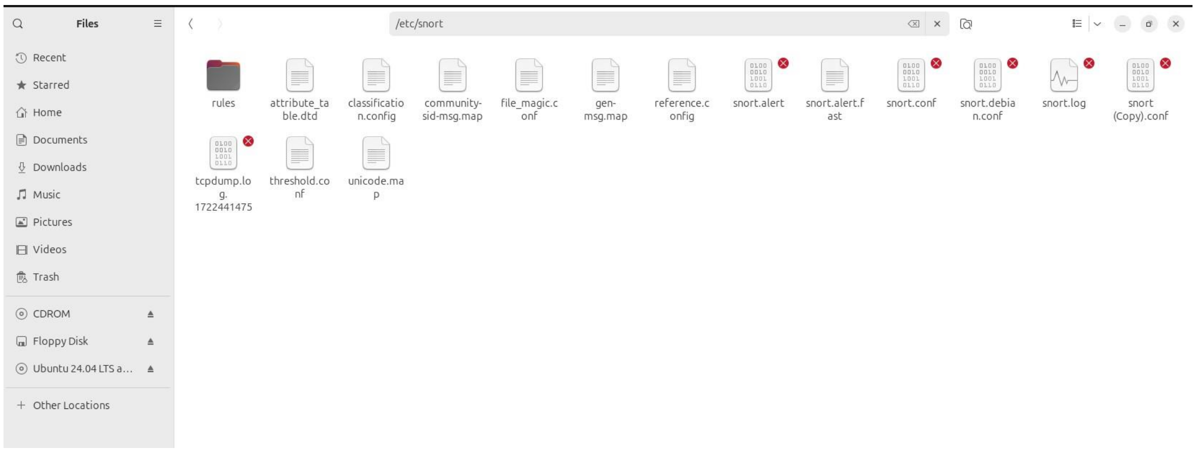
```python
# Main execution
if __name__ == "__main__":
    input_csv = "/home/mine/myresearch/iomttraffic.csv"
    input_data = pd.read_csv(input_csv, header=None)
    input_data.columns = ['index', 'timestamp', 'ip.src', 'ip.dst', 'tcp.srcport', 'tcp.dstport', 'size', 'protocol', 'time1', 'time2']
    input_data['tcp.srcport'] = input_data['tcp.srcport'].fillna(0).astype(int)
    input_data['tcp.dstport'] = input_data['tcp.dstport'].fillna(0).astype(int)
    print("Columns in CSV:", input_data.columns)
    print("First row types before rule generation:", input_data.dtypes)

    # Generate predictions and save to CSV
    predictions = predict(input_data)
    input_data['predictions'] = predictions

    # Add label column based on the predictions
    input_data['label'] = input_data['predictions'].apply(lambda x: "Detected MQTT publish packet to sensors/temperature" if x == 1 else "No threat detected")

    predictions_csv = "/home/mine/myresearch/predictions.csv"
    input_data.to_csv(predictions_csv, index=False)
    print(f"Predictions saved to {predictions_csv}")

    mqtt_traffic = input_data[(input_data['tcp.srcport'] == 1883) | (input_data['tcp.dstport'] == 1883)]
    if not mqtt_traffic.empty:
        print("MQTT traffic detected. Generating Snort rule.")
        protocol = 'tcp'
        src_ip = mqtt_traffic['ip.src'].iloc[0]
        src_port = mqtt_traffic['tcp.srcport'].iloc[0]
        dst_ip = mqtt_traffic['ip.dst'].iloc[0]
        dst_port = mqtt_traffic['tcp.dstport'].iloc[0]
        rule1 = generate_snort_rule(protocol, 'any', 'any', '192.168.136.129', '1883', 1000001, "Detected MQTT publish packet to sensors/temperature")
        rule2 = generate_snort_rule(protocol, '192.168.136.130', '20', '192.168.136.129', '1883', 10000073453453, "Detected traffic in IoMT device")
        test_and_update_snort_rules(rule1)
        test_and_update_snort_rules(rule2)
    else:
        print("No MQTT traffic detected.")
```

Running Python script

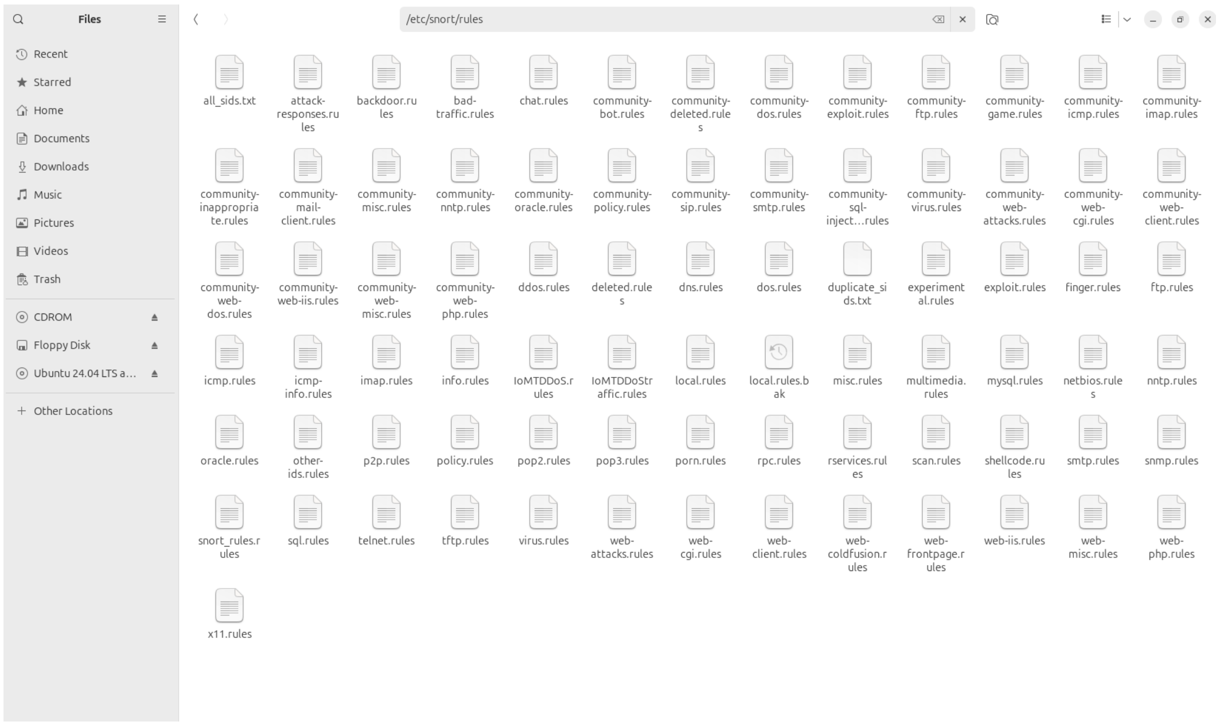


```
(venv) mine@mine:~/myresearch$ python iomt.py
2024-07-28 01:16:48.272460: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-p
oint round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-07-28 01:16:48.273307: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-07-28 01:16:48.287231: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-07-28 01:16:48.316086: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to register cuFFT factory: Attempting to register factory for plu
gin cuFFT when one has already been registered
2024-07-28 01:16:48.357256: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to register cuDNN factory: Attempting to register factory for pl
ugin cuDNN when one has already been registered
2024-07-28 01:16:48.375856: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to register cuBLAS factory: Attempting to register factory for
plugin cuBLAS when one has already been registered
2024-07-28 01:16:48.413420: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performan
ce-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-07-28 01:16:50.473625: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Model loaded successfully.
Usage: python iomt.py <path_to_data_file>
(venv) mine@mine:~/myresearch$
```

Snort File



Snort Rules



```
/etc/snort/rules

all_sids.txt  attack-responses.rules  backdoor.rules  bad-traffic.rules  chat.rules  community-bot.rules  community-deleted.rules  community-dos.rules  community-exploit.rules  community-ftp.rules  community-game.rules  community-icmp.rules  community-imap.rules
community-inappropriate.rules  community-mail-client.rules  community-misc.rules  community-nntp.rules  community-oracle.rules  community-policy.rules  community-sip.rules  community-smtp.rules  community-sql-inject…rules  community-virus.rules  community-web-attacks.rules  community-web-cgi.rules  community-web-client.rules
community-web-dos.rules  community-web-iis.rules  community-web-misc.rules  community-web-php.rules  ddos.rules  deleted.rules  dns.rules  dos.rules  duplicate_sids.txt  experimental.rules  exploit.rules  finger.rules  ftp.rules
icmp.rules  icmp-info.rules  imap.rules  info.rules  IoMTDDoS.rules  IoMTDDoStraffic.rules  local.rules  local.rules.bak  misc.rules  multimedia.rules  mysql.rules  netbios.rules  nntp.rules
oracle.rules  other-ids.rules  p2p.rules  policy.rules  pop2.rules  pop3.rules  porn.rules  rpc.rules  rservices.rules  scan.rules  shellcode.rules  smtp.rules  snmp.rules
snort_rules.rules  sql.rules  telnet.rules  tftp.rules  virus.rules  web-attacks.rules  web-cgi.rules  web-client.rules  web-coldfusion.rules  web-frontpage.rules  web-iis.rules  web-misc.rules  web-php.rules
x11.rules
```

8 References

[1]Anaconda, 2022.Anaconda [online]
Available at: Download Anaconda Distribution | Anaconda
[2]Ubuntu download [online]:
Available at: Download Ubuntu Desktop | Ubuntu
[3]Kali Linux[online]:
Available at: Get Kali | Kali Linux
[3]SNORT Installation [online]
Available at: Snort - Network Intrusion Detection & Prevention System
[4]Python Download [Online]
Available at: Download Python | Python.org
[5]VMware, 2024 [Online]
Available at: Download VMware Workstation Player | VMware