National College of Ireland

# Configuration Manual

MSc Research Project
Cyber Security

Harsh Dharmendra Patel
Student ID: X21141932

School of Computing
National College of Ireland

Supervisor: Prof. Jawad Salahuddin

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Harsh Dharmendra Patel |
| **Student ID:** | X21141932 |
| **Programme:** | MSC in Cybersecurity **Year:** 2022-2023 |
| **Module:** | Research Project |
| **Lecturer:** | Mr. Jawad Salahuddin |
| **Submission Due Date:** | 1st February 2023 |
| **Project Title:** | A Hybrid IDS using Machine Learning and Semantic Rules for Power System to Detect Cyber-Attacks. |

**Word Count:1138** **Page Count: 11**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Harsh Dharmendra Patel

**Date:** 31st January 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Harsh Dharmendra Patel
Student ID: x21141932

## 1 Introduction

The configuration manual is a report which helps us to understand the steps used for this project. It includes a guide for the development, implementation, installation, and for the deployment of the project " A Hybrid IDS using Machine Learning and Semantic Rules for Modern Power System to detect cyber-attacks" presented in this report. The main motive of this manual is to help and support at every stage of the process to achieve the final output and results, which are in this report. The manual consists of all the information about hardware, software, and procedures that are used to implement this project.

## 2 System Specification

The specification of the system is as follows,

- AMD Ryzen 7 4800H with Radeon Graphics @ 2.90 GHz
- GPU; Nvidia Geforce RTX 3050
- RAM; 24 GB
- SSD; 1TB + 500 Gb
- System Type: 64-bit Operating Systems
- Operating System: Windows 11

## 3 Software Specifications

In this section, we will discuss the software specification used to implement this model. The Anaconda prompt is used for this project and python is used as a programming language. There are some other libraries and packages installed to get proper and systematic results.

- Anaconda Prompt
- Python 3.9.12
- Sublime text 3.0
- Pandas
- Pickle
- Tkinter
- NumPy
- Anaconda Navigator
- Matplotlib
- Joblib
- Kali linux
- Ubuntu 18.04
- Virtual Box

## 4 Steps for Configuration of Machine Learning

1. To download and install Anaconda3 (Anconda, 2022)
2. extract the ‘ml_env.rar’ folder and paste it to the anaconda3’s envs folder
3. Extract the ‘HIDS_Fullcode 1’ folder
4. Run the Anaconda Navigator
5. Open in the anaconda3 prompt and in prompt used the command cd/d to change the directory to HIDS Fullcode 1.
6. (cd) Navigate to ‘HIDS_Fullcode 1’ folder.
7. Now run command: conda activate ml_env
8. Run command: python output_nsl_kdd.py
9. Run command : python output edge_iiot.py

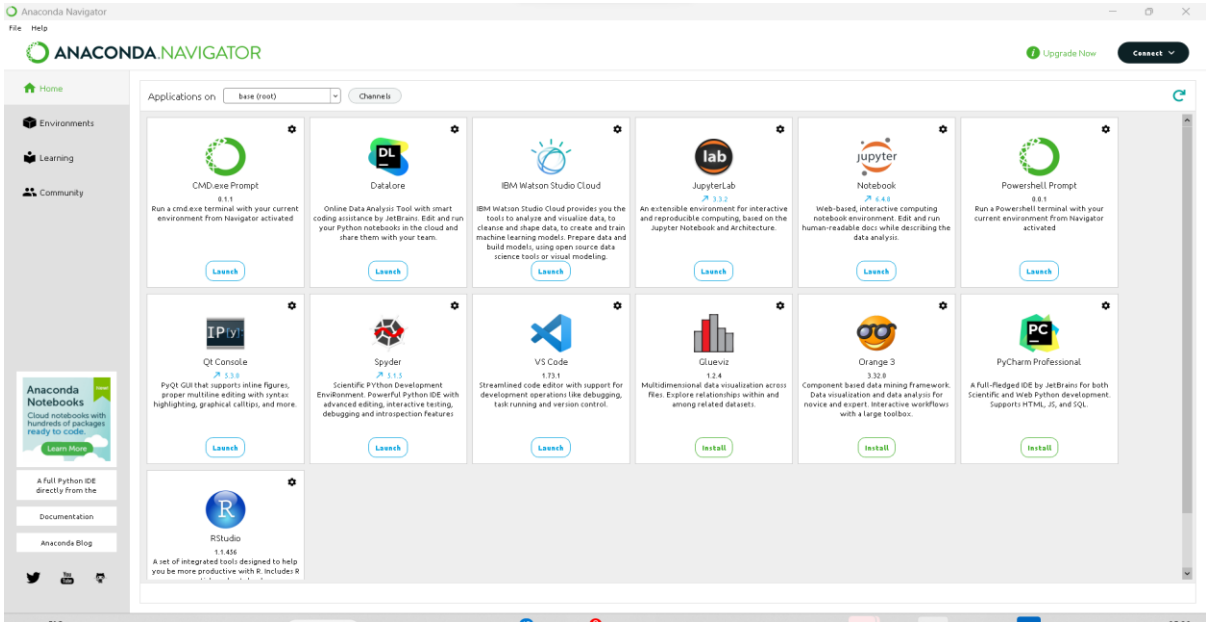


Figure 1 : Anaconda Navigator

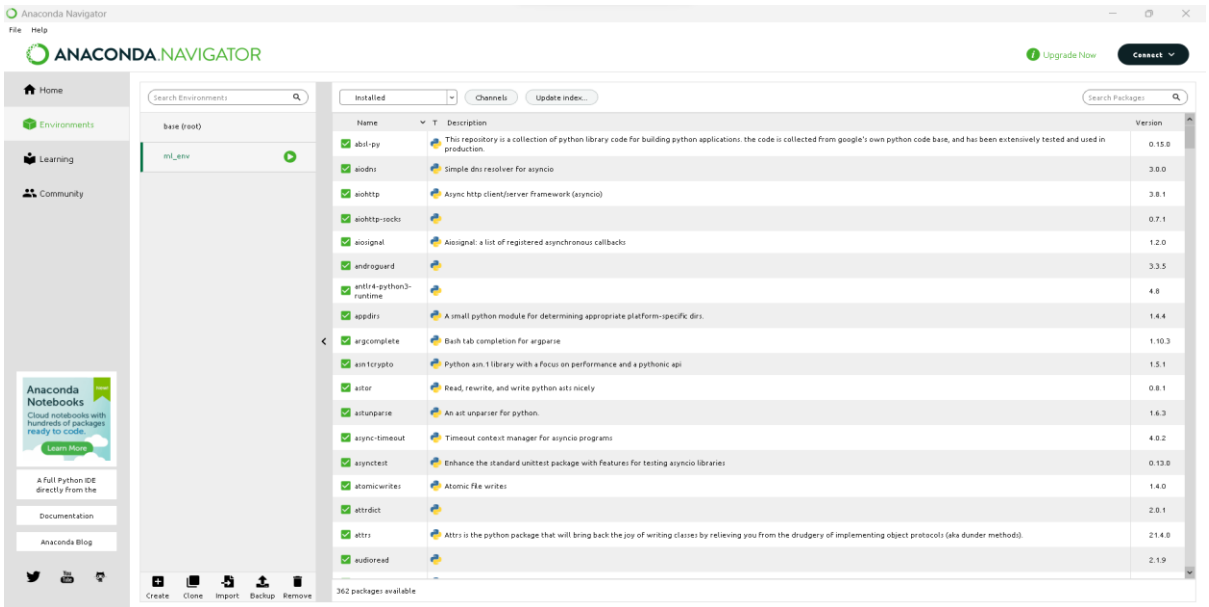


Figure 2: ml_env

# 5 Steps for Configuration of Virtual Machines and Custom rules for Snort

1. Download and Install VirtualBox (Virtual Box, 2022).
2. Set up an environment.
3. Download Kali Linux and install select the network as a host-only adapter.
4. Download and install Ubuntu 18.04 and select the network as a host-only adapter (Ubuntu, n.d.).
5. Here Kali Linux is attacking the machine and Ubuntu is the victim machine.
6. Then in the Ubuntu terminal type sudo apt-get install snort -y
7. Install the snort packages and select Ok.
8. Then in another terminal check the interface name and Ip address of the victim machine using “ifconfig”
9. Then Open snort configuration file and then type sudo nano /etc/snort/snort.conf
10. 'ipvar HOME_NET any' change to 'ipvar HOME_NET ip address'. In this write the address of the victim machine.
11. To view snort rules, use ls or cd/etc/snort/rules/
12. If we change any rules, we need to test the configuration file using the command.
13. sudo snort -T -c /etc/snort/snort.conf
14. The snort will start listening to the network packets using the command.
15. sudo snort -A console -c /etc/snort/snort.conf
16. Then copy the IP address of the victim machine (Ubuntu) and the Open terminal in kali linux.
17. nmap of this Ip provided in kali linux terminal to "Scan Ubuntu System”.
18. Then type the nmap address in kali and we can see that snort will be detecting information leak in Ubuntu system.
19. Then we can send ping to Ubuntu system, and we can see that snort is detecting ICMP ping.
20. To perform DDOS attack use command
21. sudo apt install hping3 -y
22. sudo hping3 -S -p 80 --flood --rand-source 192.168.56.101 in this we can see that multiple botnets are used to create network traffic at the victim IP address.
23. Now we can see that Snort is detecting DDOS attack.
24. For FTP Brute Force attack
25. ftp and the IP address of the victim machine.
26. For SSH Brute Force attack.
27. ssh and the IP address of the victim machine.
28. For example- ssh 192.168.56.101

# 6 Procedure for Machine Learning

## 6.1 Pre-Processing the data

- Loading the Edge-IIoT dataset and removing the unwanted columns, duplicate rows, null values and removing other attack categories.
- Loading the NSL-KDD dataset setting up the column plus data and creating a new csv file.

```
###load dataset
data=pd.read_csv("Data/EDGE_IIOT/dataset.csv")
print(data.head())
print(len(data.columns))


print(data['Attack_type'].value_counts())


#Remove (unwanted columns,NULL values, duplicated rows , Nan values)
unwanted_columns = ["frame.time", "ip.src_host", "ip.dst_host", "arp.src.proto_ipv4","arp.dst.proto_ipv4",
            "http.file_data","http.request.full_uri","icmp.transmit_timestamp",
            "http.request.uri.query", "tcp.options","tcp.payload","tcp.srcport","tcp.dstport", "udp.port", "mqtt.msg","Attack_label"]

data.drop(unwanted_columns, axis=1, inplace=True)
data.dropna(axis=0, how='any', inplace=True)
data.drop_duplicates(subset=None, keep="first", inplace=True)

print(len(data.columns))

#removing other attack categories
data.drop(data.index[data['Attack_type'] == 'Ransomware'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'Uploading'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'Backdoor'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'Vulnerability_scanner'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'Port_Scanning'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'XSS'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'Password'], inplace=True)
data.drop(data.index[data['Attack_type'] == 'Fingerprinting'], inplace=True)


```

Figure 3: Pre- Processing Edge-IIoT dataset

```
###load dataset
data=pd.read_csv("Data/NSL_KDD/dataset.csv")
print(data.head())
print(data.columns)

#Set column names
data.set_axis(['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot','num_failed_logins',
    'logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creations','num_shells','num_access_files',
    'num_outbound_cmds','is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate','rerror_rate',
    'srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count','dst_host_same_srv_rate',
    'dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate',
    'dst_host_rerror_rate','dst_host_srv_rerror_rate','attack','level'], axis=1,inplace=True)

#print(data)

#Saving the dataset (data + column names )
#data.to_csv('Data/NSL_KDD/pre_dataset1.csv',index=False)


print(data['attack'].value_counts())

#Here we are perform  mapping normal to 0, all attacks to 1
get_attack_category = data.attack.map(lambda x: 0 if x == 'normal' else 1)
data['Category'] = get_attack_category

#print(data.head())

data.drop(["attack", "level"], axis = 1, inplace=True)
print(data.head())
print(data.columns)


```

Figure 4: Pre- Processing NSL-KDD  dataset

## 6.2 Feature selection

```
# feature selection(function call)
fs = select_features(x, y)


column_names=[]
# iterating the columns
for col in x.columns:
    column_names.append(col)

feature_list=[]
# what are scores for the features
for i in range(len(fs.scores_)):
  print('Feature %d: %f' % (i, fs.scores_[i]))
  feature_list.append(fs.scores_[i])

dictionary = dict(zip(column_names, feature_list))
clean_dict = {k: dictionary[k] for k in dictionary if not pd.isna(dictionary[k])}
sorted_d = dict( sorted(clean_dict.items(), key=operator.itemgetter(1),reverse=True))
print(sorted_d)


#Creating new dataframe based on feature scores (14 features)
my_data=data[['dns.qry.name.len','dns.qry.qu','udp.time_delta','tcp.flags.ack','tcp.flags',
'tcp.len','tcp.ack_raw','tcp.checksum','icmp.seq_le','udp.stream','tcp.connection.fin','icmp.checksum','tcp.seq','http.response','Attack_type'
print(my_data.head())
print(my_data.columns)
```

Figure 5: Best 14 features are selected from Edge-IIoT dataset

```
73  # feature selection(function call)
74  fs = select_features(x, y)
75
76
77  column_names=[]
78  # iterating the columns
79  for col in x.columns:
80      column_names.append(col)
81
82  feature_list=[]
83  # what are scores for the features
84  for i in range(len(fs.scores_)):
85    print('Feature %d: %f' % (i, fs.scores_[i]))
86    feature_list.append(fs.scores_[i])
87
88  dictionary = dict(zip(column_names, feature_list))
89  sorted_d = dict( sorted(dictionary.items(), key=operator.itemgetter(1),reverse=True))
90  print(sorted_d)
91
92
93  #Creating new dataframe based on feature scores (15 features)
94  my_data=pre_data[['same_srv_rate','dst_host_srv_count','dst_host_same_srv_rate','logged_in','dst_host_srv_serror_rate',
95  'dst_host_serror_rate','serror_rate','srv_serror_rate','flag','count','dst_host_count','service','protocol_type',
96  'src_bytes','dst_bytes','Category']]
97  print(my_data.head())
98  print(my_data.columns)
99
100
101 #my_data.to_csv("Project_Dataset/NSL_KDD/final_dataset_preprocessed.csv",index=False)
102 # print(my_data.head())
```

Figure 6: Best 15 features are selected from NSL-KDD dataset

```
 99
100
101   def select_features(x, y):
102
103       # configure to select a subset of features
104       fs = SelectKBest(score_func=f_classif, k="all")
105
106       # learn relationship from training data
107       fs.fit(x, y)
108
109       return fs
```

Figure 7: Feature selection function using the Annova classifier

## 6.3 Training and Testing of models

```
# TRAIN - TEST SPLITTING
x_train, x_test, y_train, y_test = train_test_split(x_final, y_final, test_size=0.2)
print("\nTraining set")
print(x_train.shape)
print(y_train.shape)
print("\nTesting set")
print(x_test.shape)
print(y_test.shape)


#Data balancing using SMOTE
counter = Counter(y_train)
print("Before Balancing :", counter)

smt = SMOTE(k_neighbors=1)

x_train_sm, y_train_sm = smt.fit_resample(x_train, y_train)

counter = Counter(y_train_sm)
print("After Balancing : ", counter)


#Perform standardization
scaler = StandardScaler()
x_train_sm = scaler.fit_transform(x_train_sm)
x_test = scaler.transform(x_test)
print(x_train_sm.shape, y_train_sm.shape, x_test.shape, y_test.shape)
pickle.dump(scaler,open('Trained_Model/EDGE_IIOT/scaler_edgeiiot.pkl','wb'))
```

Figure 8: Training and testing of Edge-IIoT dataset and using smote for standardization.

```
# TRAIN - TEST SPLITTING
x_train, x_test, y_train, y_test = train_test_split(x_final, y_final, test_size=0.2)
print("\nTraining set")
print(x_train.shape)
print(y_train.shape)
print("\nTesting set")
print(x_test.shape)
print(y_test.shape)


#Data balancing using SMOTE
counter = Counter(y_train)
print("Before Balancing :", counter)

smt = SMOTE(k_neighbors=1)

x_train_sm, y_train_sm = smt.fit_resample(x_train, y_train)

counter = Counter(y_train_sm)
print("After Balancing : ", counter)


#Perform standardization
scaler = StandardScaler()
x_train_sm = scaler.fit_transform(x_train_sm)
x_test = scaler.transform(x_test)
print(x_train_sm.shape, y_train_sm.shape, x_test.shape, y_test.shape)
pickle.dump(scaler,open('Trained_Model/NSL_KDD/scaler_nsl.pkl','wb'))
```

Figure 9: Training and testing of NSL-KDD dataset and using smote for standardization.

## 6.4 Results

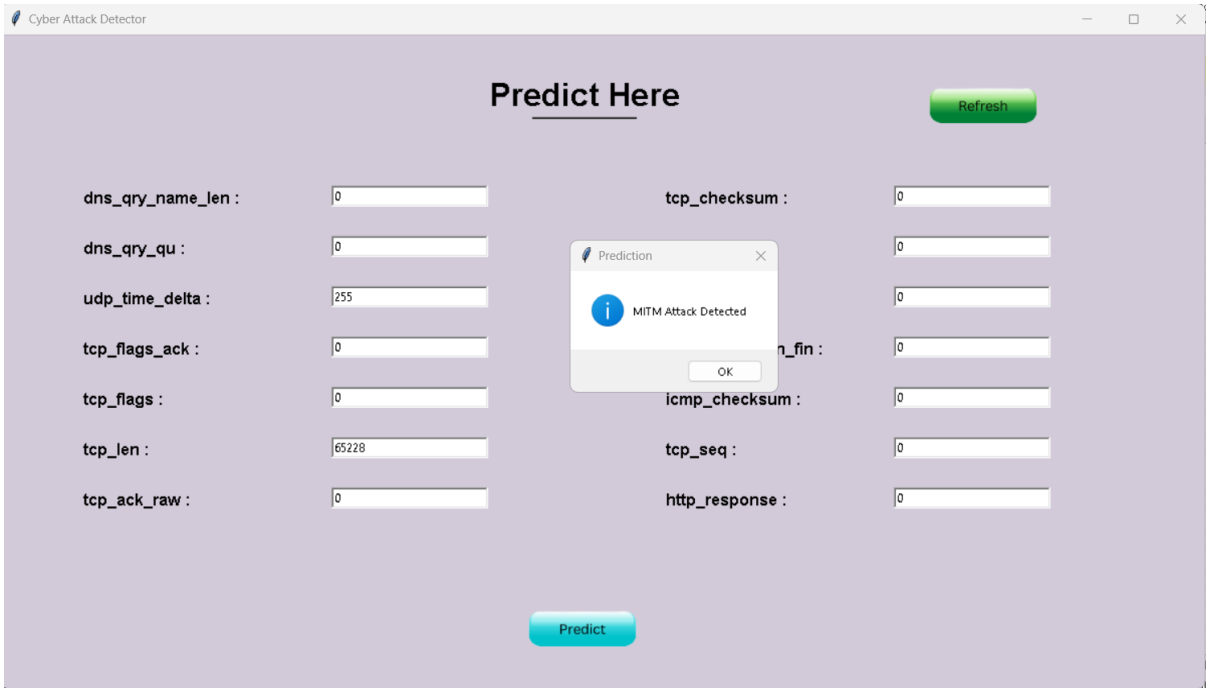


Figure 10: MiTM attack is detected in Edge-IIoT dataset

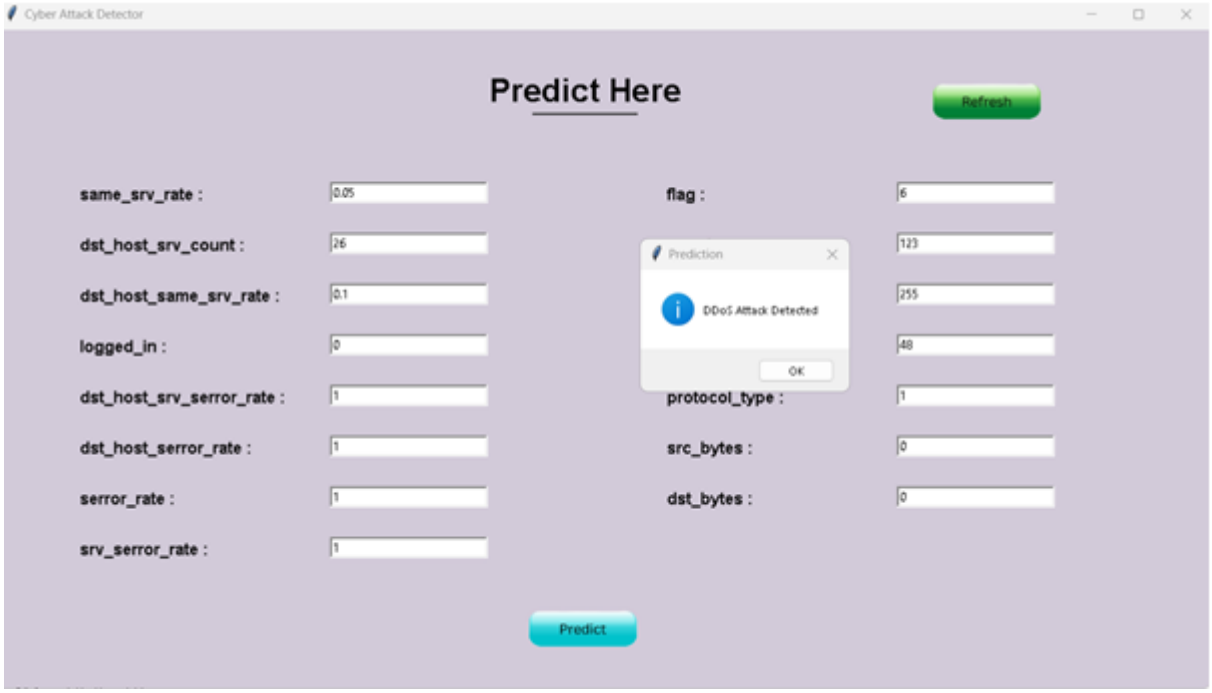


Figure 11: DDOS attack is detected for NSl-KDD dataset

# 7 Procedure for Virtual Box

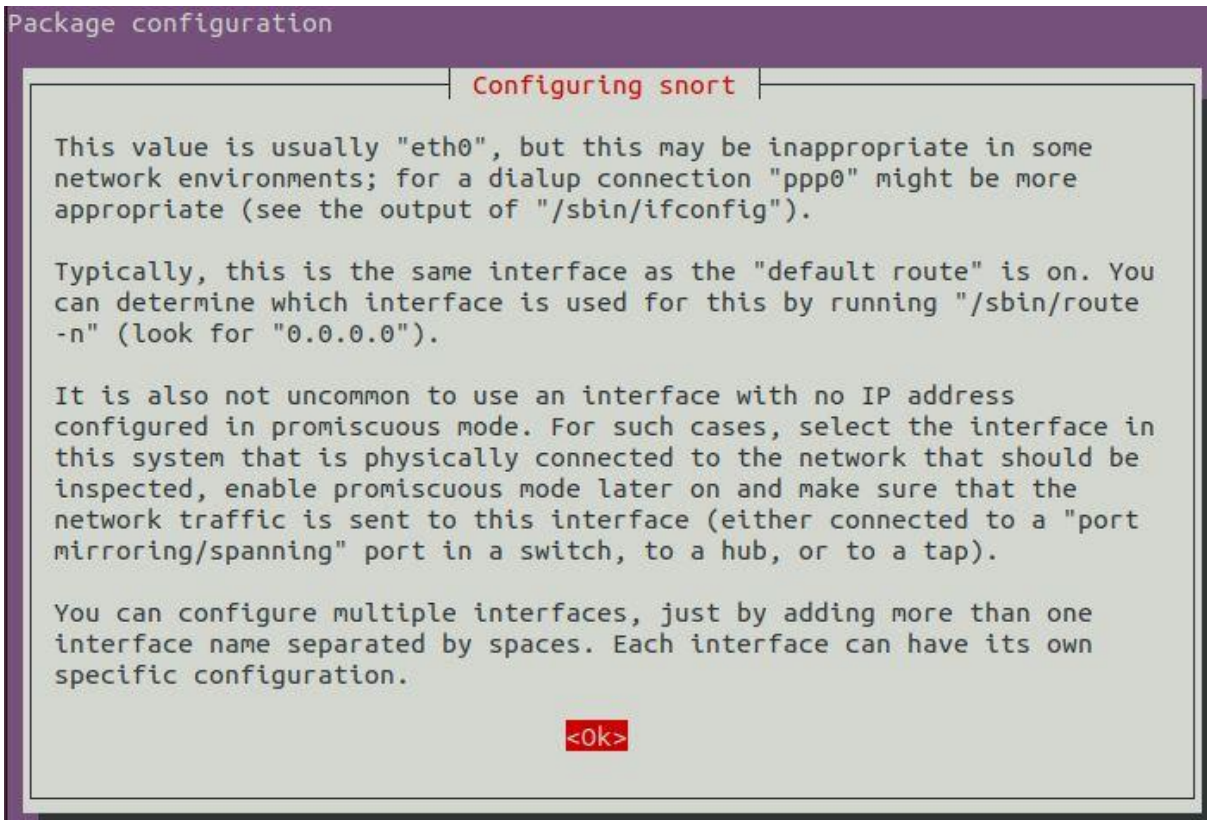


Figure 12: Snort configuration

```
flemingdenni@flemingdenni:/etc/snort$ cd rules
flemingdenni@flemingdenni:/etc/snort/rules$ ls
attack-responses.rules          community-web-dos.rules    policy.rules
backdoor.rules                  community-web-iis.rules    pop2.rules
bad-traffic.rules               community-web-misc.rules   pop3.rules
chat.rules                      community-web-php.rules    porn.rules
community-bot.rules             ddos.rules                 rpc.rules
community-deleted.rules         deleted.rules              rservices.rules
community-dos.rules             dns.rules                  scan.rules
community-exploit.rules         dos.rules                  shellcode.rules
community-ftp.rules             experimental.rules         smtp.rules
community-game.rules            exploit.rules              snmp.rules
community-icmp.rules            finger.rules               sql.rules
community-imap.rules            ftp.rules                  telnet.rules
community-inappropriate.rules   icmp-info.rules            tftp.rules
community-mail-client.rules     icmp.rules                 virus.rules
community-misc.rules            imap.rules                 web-attacks.rules
community-nntp.rules            info.rules                 web-cgi.rules
community-oracle.rules          local.rules                web-client.rules
community-policy.rules          misc.rules                 web-coldfusion.rules
community-sip.rules             multimedia.rules           web-frontpage.rules
community-smtp.rules            mysql.rules                web-iis.rules
community-sql-injection.rules   netbios.rules              web-misc.rules
community-virus.rules           nntp.rules                 web-php.rules
community-web-attacks.rules     oracle.rules               x11.rules
community-web-cgi.rules         other-ids.rules
community-web-client.rules      p2p.rules
```

Figure 13: Snort rules

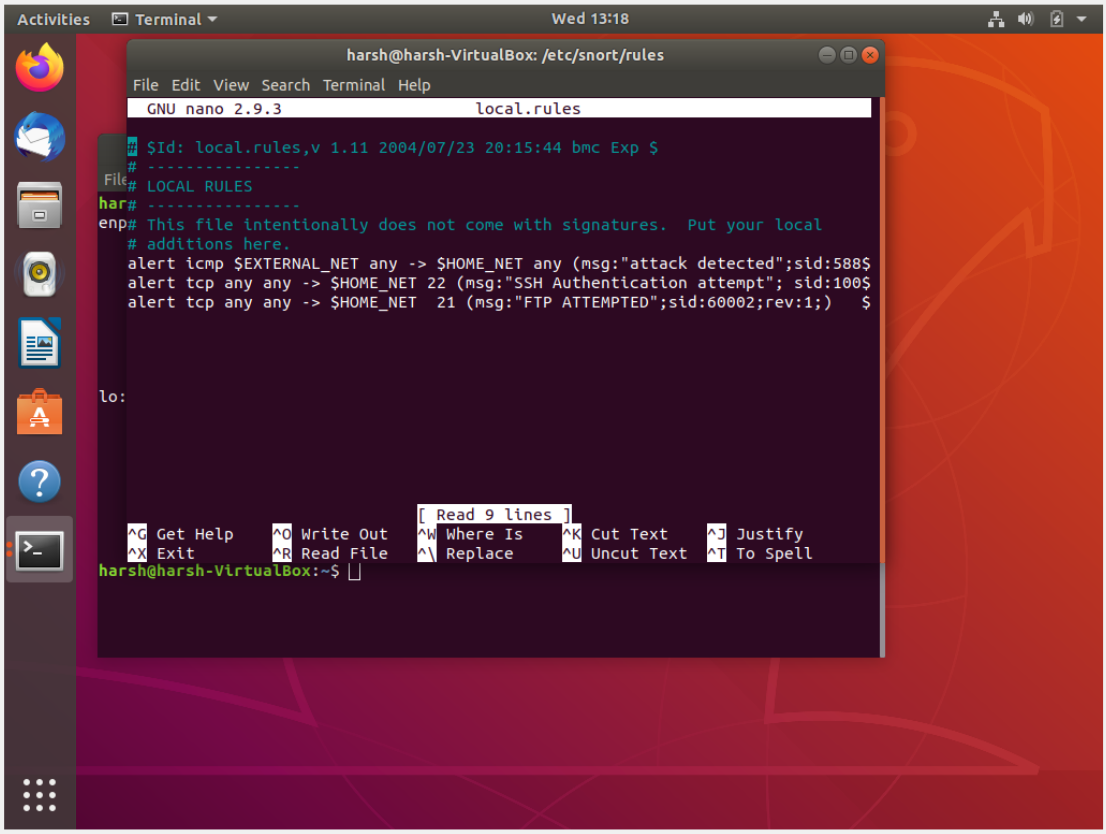


Figure 14: Custom rules for Snort

# 8 References

Anconda, 2022. *Anaconda.* [Online]
Available at: https://www.anaconda.com/products/distribution
Ubuntu, n.d. *Ubuntu download.* [Online]
Available at: https://releases.ubuntu.com/18.04/
[Accessed 13 December 2022].
Virtual Box, 2022. [Online]
Available at: https://www.virtualbox.org/wiki/Downloads