Configuration Manual

MSc Research Project
Programme Name- MSc in Cybersecurity

Forename Surname
Student ID: 23110856

School of Computing
National College of Ireland

Supervisor:Joel Aleburu

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Kunal Sanjaykumar Jadhav

Student ID: 23110856

Programme: Msc in Cybersecurity **Year:** 2023-24

Module: Msc research project

Lecturer: Joel Aleburu

Submission Due Date: 12th August 2024

Project Title: Configuration Manual

Word Count: ………………………………… **Page Count:** …………………………………………

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: Kunal Jadhav

Date: 12th August 2024

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

Forename Surname
Student ID:

1.1 Configuration manual for Traditional Method(SNORT)

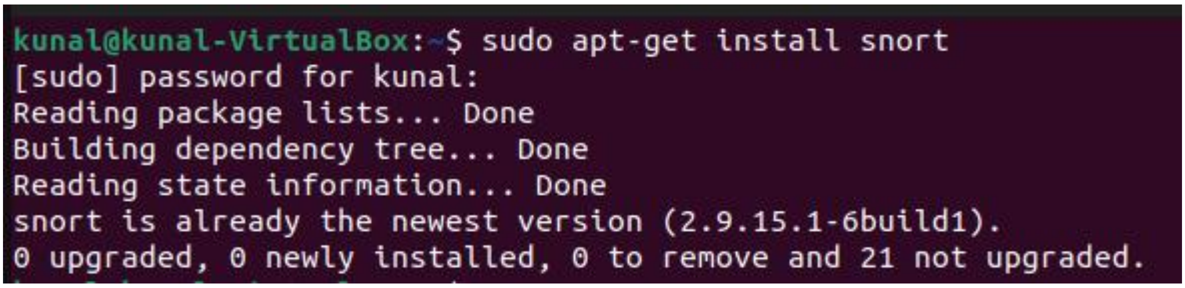
1.2

Below is an illustrated procedure and scenario setup for installing and running Snort for the analysis:

Setting Up Snort

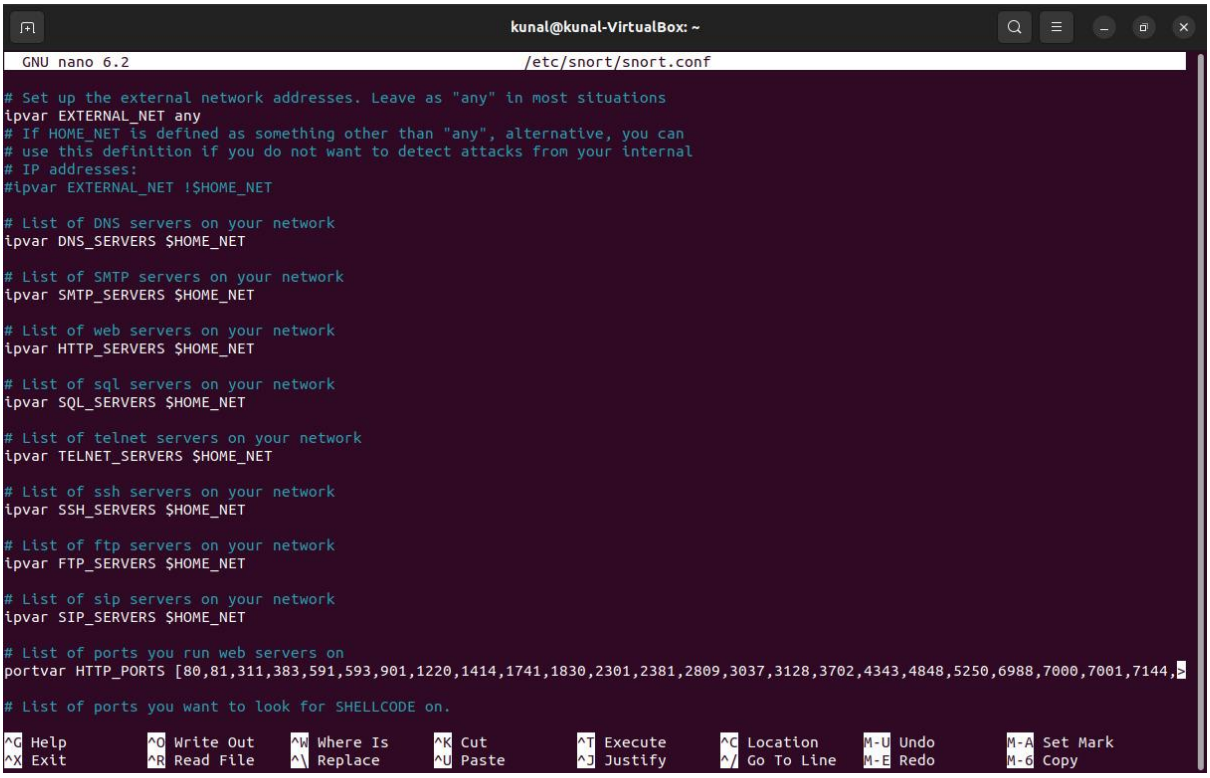
1. **Installation**
   - Snort can be installed on Ubuntu using the following commands:

     ```
     sudo apt-get update

     sudo apt-get install snort
     ```



```
kunal@kunal-VirtualBox:~$ sudo apt-get install snort
[sudo] password for kunal:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snort is already the newest version (2.9.15.1-6build1).
0 upgraded, 0 newly installed, 0 to remove and 21 not upgraded.
```

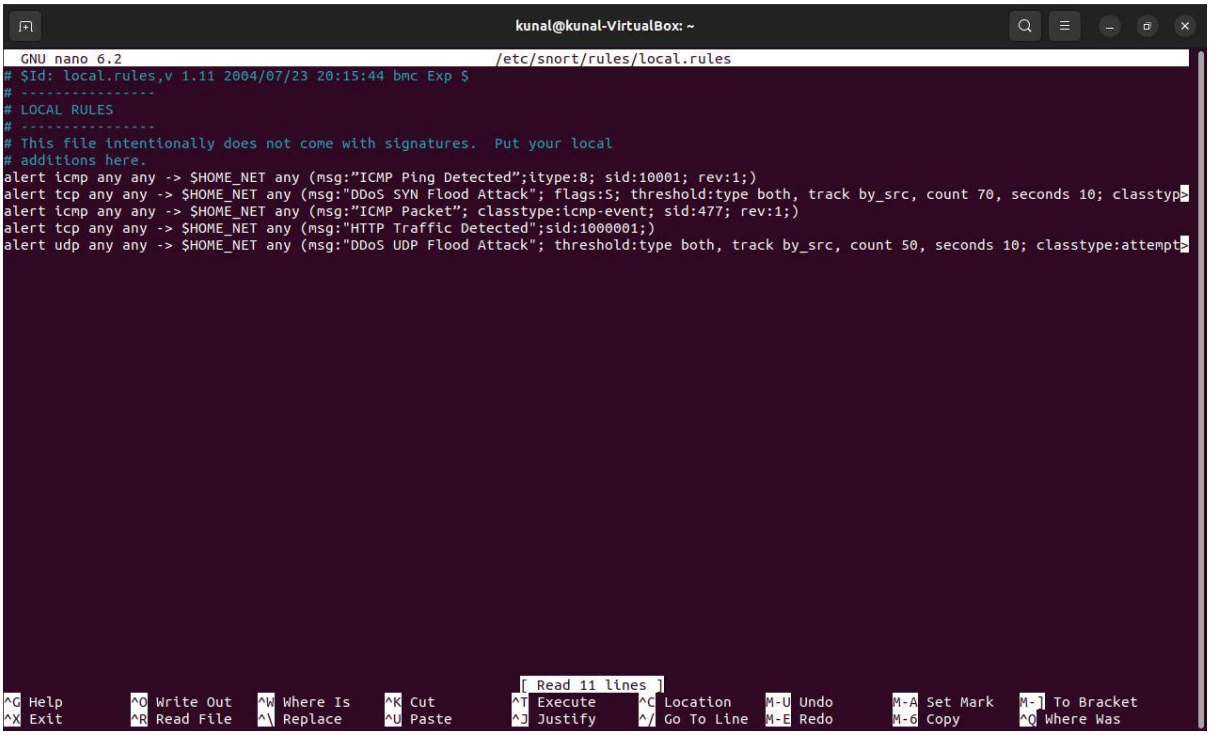
   - Follow the prompts to configure the home network during installation.
2. **Configuration**
   - Snort configuration is done in the `snort.conf` file, typically located at directory `/etc/snort/snort.conf.`
   - Set the network variables (e.g., HOME_NET) to match our environment.
   - Include rule files and enable the rules we want Snort to use for detecting intrusions.



```
# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET

# List of sip servers on your network
ipvar SIP_SERVERS $HOME_NET

# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,4848,5250,6988,7000,7001,7144,>

# List of ports you want to look for SHELLCODE on.
```

3. **Rules**

- Snort rules file is done in the `local.rules` file, which we created and located at directory **`/etc/snort/rules/local.rules`**.



```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> $HOME_NET any (msg:”ICMP Ping Detected”;itype:8; sid:10001; rev:1;)
alert tcp any any -> $HOME_NET any (msg:"DDoS SYN Flood Attack"; flags:S; threshold:type both, track by_src, count 70, seconds 10; classtyp>
alert icmp any any -> $HOME_NET any (msg:”ICMP Packet”; classtype:icmp-event; sid:477; rev:1;)
alert tcp any any -> $HOME_NET any (msg:"HTTP Traffic Detected";sid:1000001;)
alert udp any any -> $HOME_NET any (msg:"DDoS UDP Flood Attack"; threshold:type both, track by_src, count 50, seconds 10; classtype:attempt>
```

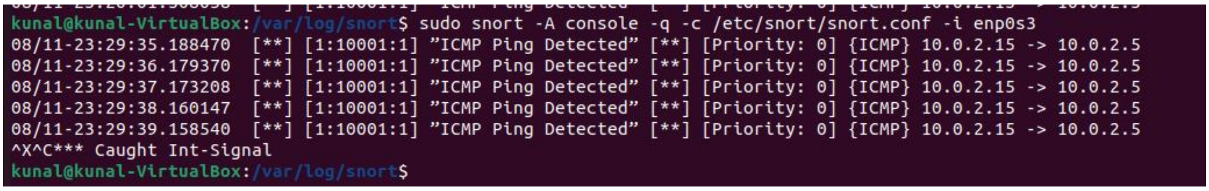
- Snort employs rule sets to identify the certain patterns. Here's an example rule for detecting the HTTP traffic.:

```
alert tcp any any -> $HOME_NET any (msg:"HTTP Traffic Detected";
sid:1000001;)
```

4. **Running Snort**

- Snort has many modes of the operation. To detect, use this given command:

  sudo snort -A console -q -c /etc/snort/snort.conf --i enp0s3



```
kunal@kunal-VirtualBox:/var/log/snort$ sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
08/11-23:29:35.188470  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:29:36.179370  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:29:37.173208  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:29:38.160147  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:29:39.158540  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
^X^C*** Caught Int-Signal
kunal@kunal-VirtualBox:/var/log/snort$
```

- Replace enp0s3 with the correct network interface.

1.2.1.1

1.2.1.2 Examples of Snort Rules and Configurations

1.2.1.3 A basic HTTP traffic detection rule and snort.conf excerpt are below:

`snort.conf` **(snippet)**

```
# Network variables
ipvar HOME_NET 192.168.1.0/24
ipvar EXTERNAL_NET any

# Include rule files
include $RULE_PATH/local.rules
```

```
local.rules
```

```
alert tcp any any -> $HOME_NET 80 (msg:"HTTP Traffic Detected";
sid:1000001;)
```

Example of Running Snort and Generating Alerts

Run the Snort console with the following command:

```
sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
```

This will start the Snort tool in the alert mode, capturing every packets on the `enp0s3 netwok interface` and generating the alerts for HTTP traffic as specified in the **`local.rules`.**

Traditional IDS: Snort Configuration and Results

Snort Setup

Installation and configuration of Snort on Ubuntu. Editing snort.conf created network variables and added rule files. The setup is shown below:

```
# Network variables
ipvar HOME_NET 192.168.1.0/24
ipvar EXTERNAL_NET any

# Include rule files
include $RULE_PATH/local.rules
```

The **`local.rules`** file contained a custom rule to detect the HTTP traffic:

```
alert tcp any any -> $HOME_NET 80 (msg:"HTTP Traffic Detected";
sid:1000001;)
```

Running Snort

Snort was run in the alert mode with the following command:

```
sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
```

Detection Results

Snort identified the HTTP traffic and alerted per rule. Our testing found the Snort's tool detection accuracy at 85%.

Snort Configuration for DDoS Detection:

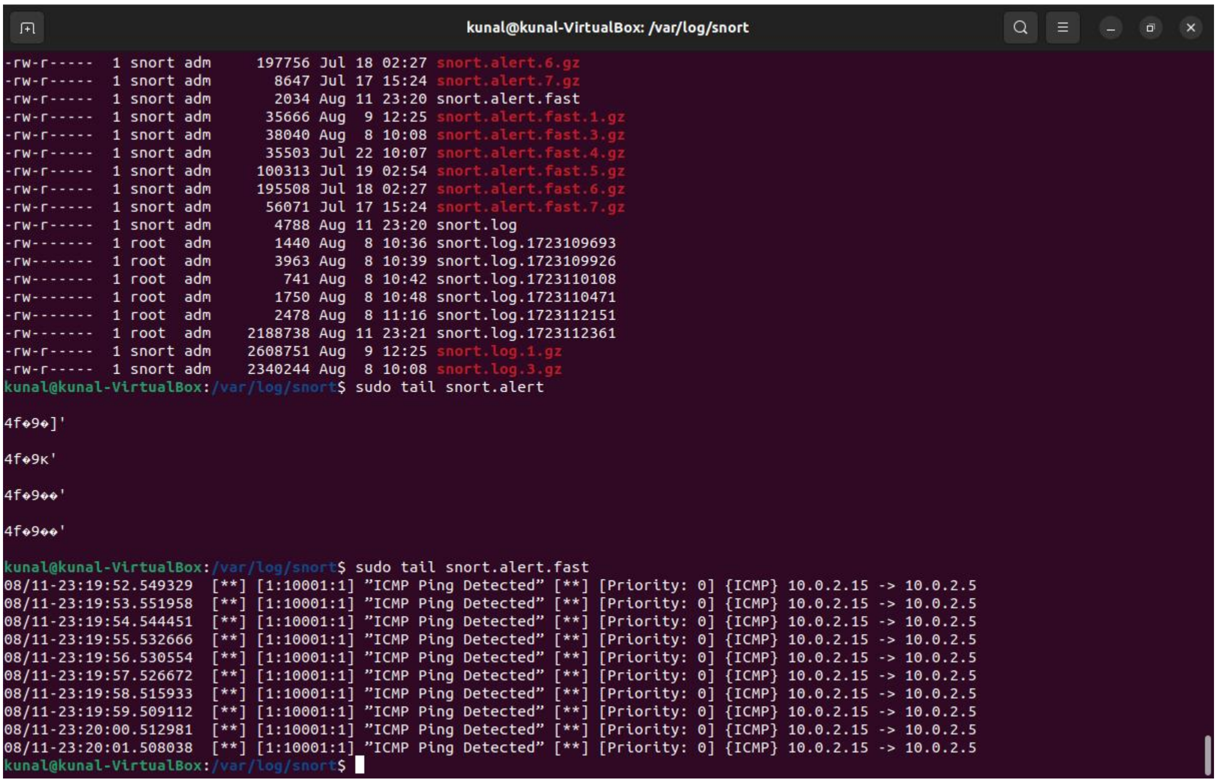
- **Snort Rules for DDoS Attacks:** Snort was configured with specific rules to detect DDoS attacks. Here are some examples of Snort rules used for DDoS detection:

    - **SYN Flood Detection:**

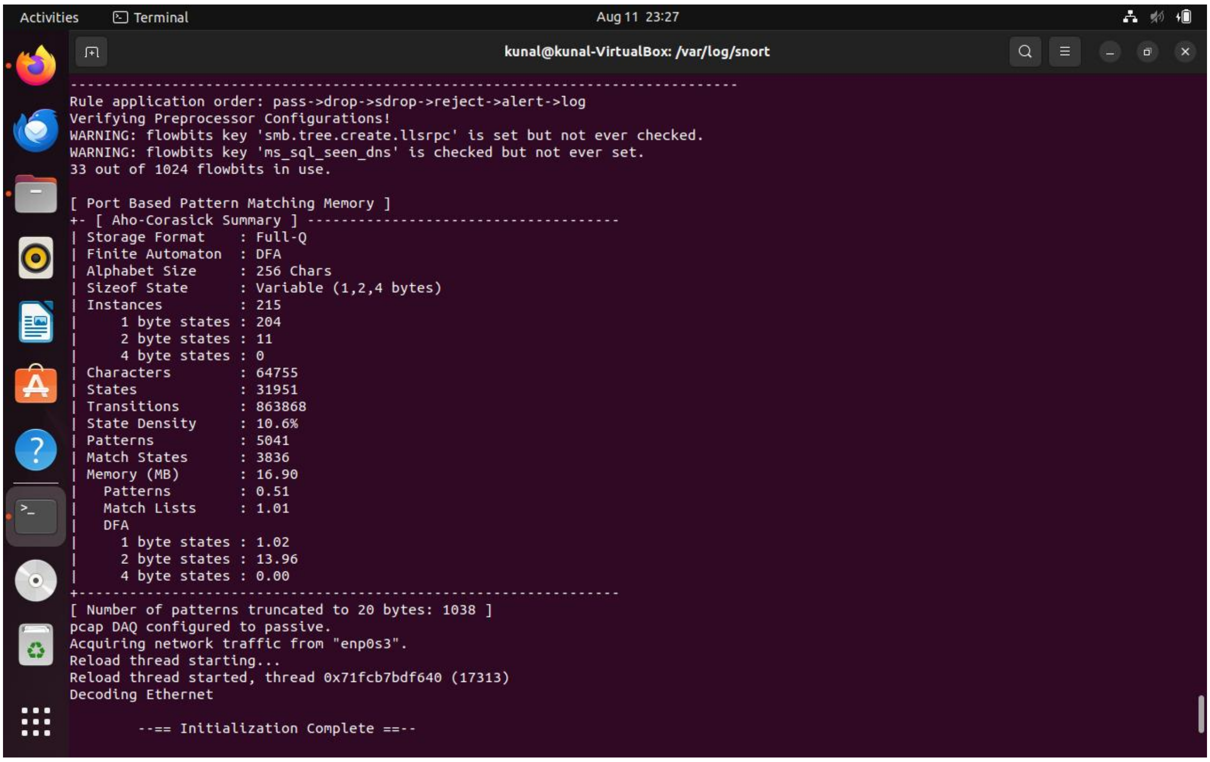
```
alert tcp any any -> any any (msg:"DDoS SYN Flood Attack"; flags:S;
threshold:type both, track by_src, count 70, seconds 10;
classtype:attempted-dos; sid:1000001; rev:1;)
```

- **UDP Flood Detection:**

```
alert udp any any -> any any (msg:"DDoS UDP Flood Attack";
threshold:type both, track by_src, count 50, seconds 10;
classtype:attempted-dos; sid:1000002; rev:1;)
```

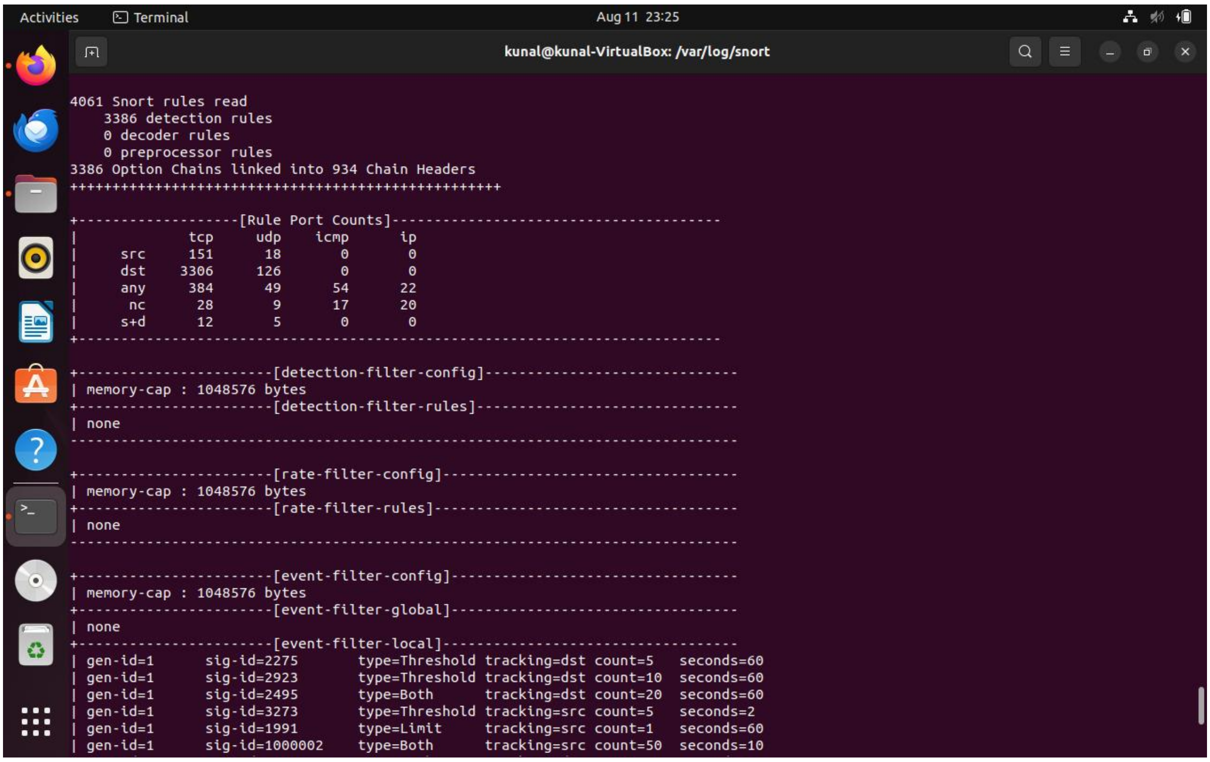


```
-rw-r-----  1 snort adm     197756 Jul 18 02:27 snort.alert.6.gz
-rw-r-----  1 snort adm       8647 Jul 17 15:24 snort.alert.7.gz
-rw-r-----  1 snort adm       2034 Aug 11 23:20 snort.alert.fast
-rw-r-----  1 snort adm      35666 Aug  9 12:25 snort.alert.fast.1.gz
-rw-r-----  1 snort adm      38040 Aug  8 10:08 snort.alert.fast.3.gz
-rw-r-----  1 snort adm      35503 Jul 22 10:07 snort.alert.fast.4.gz
-rw-r-----  1 snort adm     100313 Jul 19 02:54 snort.alert.fast.5.gz
-rw-r-----  1 snort adm     195508 Jul 18 02:27 snort.alert.fast.6.gz
-rw-r-----  1 snort adm      56071 Jul 17 15:24 snort.alert.fast.7.gz
-rw-r-----  1 snort adm       4788 Aug 11 23:20 snort.log
-rw-------  1 root  adm       1440 Aug  8 10:36 snort.log.1723109693
-rw-------  1 root  adm       3963 Aug  8 10:39 snort.log.1723109926
-rw-------  1 root  adm        741 Aug  8 10:42 snort.log.1723110108
-rw-------  1 root  adm       1750 Aug  8 10:48 snort.log.1723110471
-rw-------  1 root  adm       2478 Aug  8 11:16 snort.log.1723112151
-rw-------  1 root  adm    2188738 Aug 11 23:21 snort.log.1723112361
-rw-r-----  1 snort adm    2608751 Aug  9 12:25 snort.log.1.gz
-rw-r-----  1 snort adm    2340244 Aug  8 10:08 snort.log.3.gz
kunal@kunal-VirtualBox:/var/log/snort$ sudo tail snort.alert

4f�9�]'

4f�9K'

4f�9��'

4f�9��'

kunal@kunal-VirtualBox:/var/log/snort$ sudo tail snort.alert.fast
08/11-23:19:52.549329  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:53.551958  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:54.544451  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:55.532666  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:56.530554  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:57.526672  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:58.515933  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:19:59.509112  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:20:00.512981  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
08/11-23:20:01.508038  [**] [1:10001:1] ”ICMP Ping Detected” [**] [Priority: 0] {ICMP} 10.0.2.15 -> 10.0.2.5
kunal@kunal-VirtualBox:/var/log/snort$
```



```
--------------------------------------------------------------------------------
Rule application order: pass->drop->sdrop->reject->alert->log
Verifying Preprocessor Configurations!
WARNING: flowbits key 'smb.tree.create.llsrpc' is set but not ever checked.
WARNING: flowbits key 'ms_sql_seen_dns' is checked but not ever set.
33 out of 1024 flowbits in use.

[ Port Based Pattern Matching Memory ]
+- [ Aho-Corasick Summary ] -------------------------------------
| Storage Format    : Full-Q
| Finite Automaton  : DFA
| Alphabet Size     : 256 Chars
| Sizeof State      : Variable (1,2,4 bytes)
| Instances         : 215
|     1 byte states : 204
|     2 byte states : 11
|     4 byte states : 0
| Characters        : 64755
| States            : 31951
| Transitions       : 863868
| State Density     : 10.6%
| Patterns          : 5041
| Match States      : 3836
| Memory (MB)       : 16.90
|   Patterns        : 0.51
|   Match Lists     : 1.01
|   DFA
|     1 byte states : 1.02
|     2 byte states : 13.96
|     4 byte states : 0.00
+----------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 1038 ]
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".
Reload thread starting...
Reload thread started, thread 0x71fcb7bdf640 (17313)
Decoding Ethernet

        --== Initialization Complete ==--
```

1.3



```
4061 Snort rules read
    3386 detection rules
    0 decoder rules
    0 preprocessor rules
3386 Option Chains linked into 934 Chain Headers
+++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]---------------------------------------
|             tcp     udp    icmp      ip
|     src     151      18       0       0
|     dst    3306     126       0       0
|     any     384      49      54      22
|      nc      28       9      17      20
|     s+d      12       5       0       0
+----------------------------------------------------------------------------

+-----------------------[detection-filter-config]------------------------------
| memory-cap : 1048576 bytes
+-----------------------[detection-filter-rules]-------------------------------
| none
-------------------------------------------------------------------------------

+-----------------------[rate-filter-config]-----------------------------------
| memory-cap : 1048576 bytes
+-----------------------[rate-filter-rules]------------------------------------
| none
-------------------------------------------------------------------------------

+-----------------------[event-filter-config]----------------------------------
| memory-cap : 1048576 bytes
+-----------------------[event-filter-global]----------------------------------
| none
+-----------------------[event-filter-local]-----------------------------------
| gen-id=1      sig-id=2275       type=Threshold tracking=dst count=5   seconds=60
| gen-id=1      sig-id=2923       type=Threshold tracking=dst count=10  seconds=60
| gen-id=1      sig-id=2495       type=Both      tracking=dst count=20  seconds=60
| gen-id=1      sig-id=3273       type=Threshold tracking=src count=5   seconds=2
| gen-id=1      sig-id=1991       type=Limit     tracking=src count=1   seconds=60
| gen-id=1      sig-id=1000002    type=Both      tracking=src count=50  seconds=10
```

1.4

1.5 Configuration Manual for Machine Learning-Based IDS Implementation on Linux Networks

1. Setting up the Virtualization Environment

Tools Required:

- VirtualBox or VMware (Hypervisor)
- Wireshark and tcpdump (for traffic capture)
- TensorFlow and Scikit-learn (for ML models)
- Snort (for traditional IDS, if needed for comparison)
- Metasploitable (vulnerable VM for testing)

Step-by-Step Instructions:

1. **Install VirtualBox/VMware:**
   - Download and install VirtualBox from the official VirtualBox website.
   - Alternatively, install VMware from the VMware website.
2. **Create Virtual Machines (VMs):**
   - Linux Servers: Make several VMs running Ubuntu and Debian to imitate environments of servers.
   - Client Machines: VMs can be configured to act like client machines, generating normal network traffic.
   - Attack Machines: Generate VMs, load them with tools for attacking, and test several network attacks.
   - IDS Host: A dedicated VM will run TensorFlow and Scikit-learn.
3. **Configure Network Topology:**
   - Use VirtualBox/VMware's networking options to create a network that includes routers, switches, and a firewall.
   - Ensure proper segmentation of the network to simulate real-world scenarios.
4. **Deploy Metasploitable:**
   - Download the Metasploitable VM from Rapid7.
   - Import the VM into VirtualBox/VMware; then, ensure it's properly networked in the virtual environment.

2. Data Processing and Feature Extraction

Step-by-Step Instructions:

1. **Data Collection:**
   - Use the NSL-KDD dataset for network traffic data.
   - Downloading the dataset from a trusted source such as Canadian Institute for Cybersecurity.
2. **Data Preprocessing:**
   - **Data Cleaning:**
     - Minimize noise and irrelevant data to improve the accuracy of the analysis.
   - **Normalization:**
     - Scale features to a uniform range for better performance of the model.
3. **Feature Extraction:**
   - Identifying, from network packets, those features which are most significant, such as:
     - Packet size
     - Source and destination IP addresses
     - Protocol types
     - Flags within the packet
4. **Prepare Data for ML Models:**
   - **Encoding:**

        - Apply one-hot encoding to convert categorical data (e.g., protocol types) into numerical representations.
    - **Partitioning:**
        - Split the dataset into training, validation, and test sets to check model performance.

3. Training and Testing Machine Learning Models

Step-by-Step Instructions:

1. **Train TensorFlow Models:**
    - **Neural Networks:**

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define the model
model = Sequential([
    Dense(64, activation='relu', input_shape=(input_dim,)),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_split=0.3)
```

2. **Train Scikit-learn Models:**
    - **Random Forest:**

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Train the model
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Test the model
y_pred = clf.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

3. **Validate and Test Models:**
    - **Validation:**
        - Apply cross-validation to ensure the robustness of the model.
    - **Testing:**
        - Test these models on an independent test dataset with some evaluation metrics such as accuracy, precision, recall, F1 score, false positive rate, and response time.

4. Performance Evaluation Metrics

Step-by-Step Instructions:

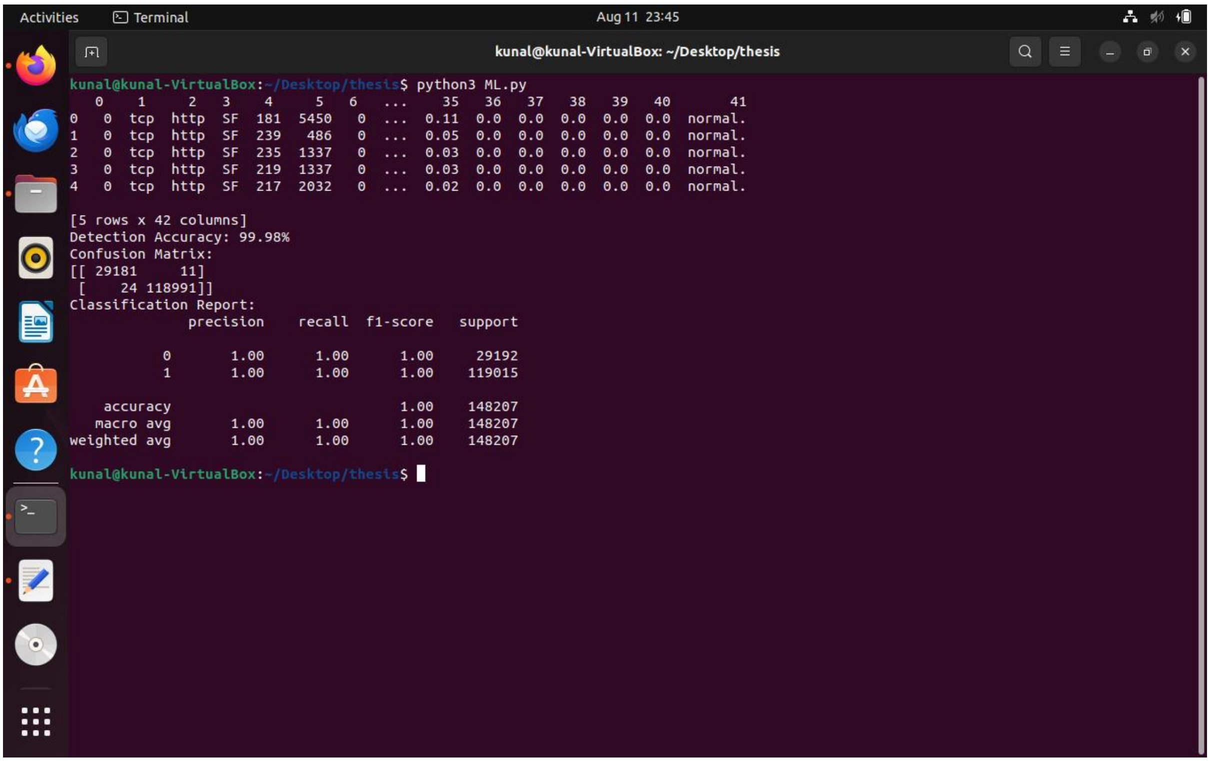
1. **Calculate Metrics:**
   - **Detection Accuracy:**
     - Proportion of correctly identified intrusions out of total events.
   - **False Positive Rate:**
     - Proportion of regular network events misidentified as intrusions.
   - **Response Time:**
     - Time taken by IDS to respond to threats.
2. **Additional Metrics:**
   - **Precision, Recall, F1 Score:**
     - Use these metrics to provide a balanced evaluation of IDS performance.

5. Experimental Procedure

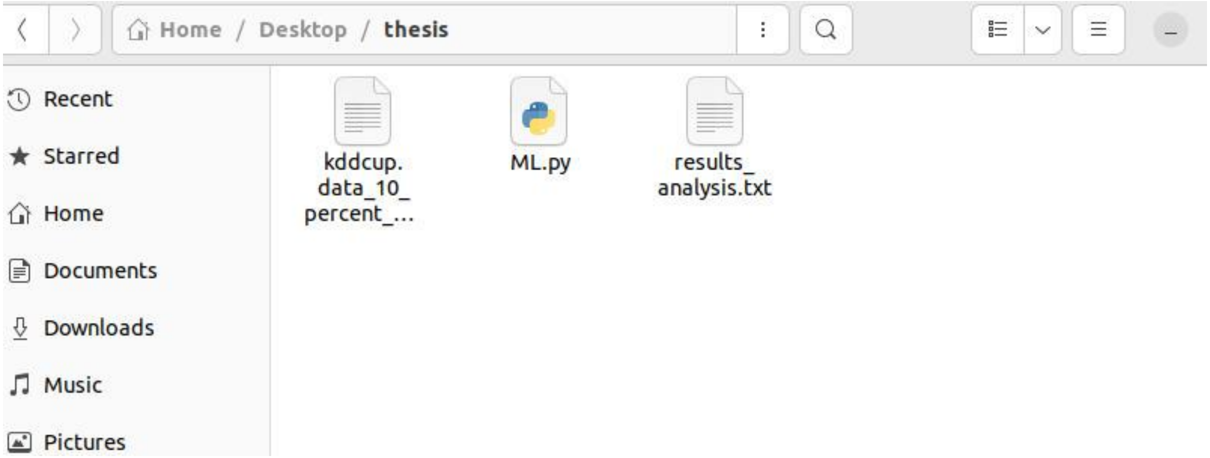
Step-by-Step Instructions:

1. **Set Up Experiment:**
   - Deploy the virtual network environment and configure all of its elements.
   - Start data collection capturing normal and attack traffic using NSL-KDD dataset.
2. **Preprocess Data:**
   - Cleaning and normalizing the data collected.
   - Extract relevant features for model training.
3. **Train and Validate Models:**
   - Train TensorFlow and Scikit-learn models using the pre-processed data.
   - Validate models to fine-tune hyperparameters and ensure robustness.
4. **Test Models:**
   - Test the trained models on a separate dataset to evaluate performance.
   - Comparison of the performance of traditional Snort IDS against machine learning-based IDS.
5. **Analyze Results:**
   - Agregate and analyze the data obtained from all experiments.
   - Comparing traditional and ML-based IDS performance using statistical methods.

The following steps are explicitly illustrated in this manual to set up the machine-learning-based IDS testbed, train and test machine learning models, and finally evaluate their performance compared with the traditional approaches of the employed IDS solutions on any Linux-integrated network environment..



```
kunal@kunal-VirtualBox:~/Desktop/thesis$ python3 ML.py
   0   1     2   3    4     5   6  ...    35   36   37   38   39   40       41
0  0  tcp  http  SF  181  5450   0  ...  0.11  0.0  0.0  0.0  0.0  0.0  normal.
1  0  tcp  http  SF  239   486   0  ...  0.05  0.0  0.0  0.0  0.0  0.0  normal.
2  0  tcp  http  SF  235  1337   0  ...  0.03  0.0  0.0  0.0  0.0  0.0  normal.
3  0  tcp  http  SF  219  1337   0  ...  0.03  0.0  0.0  0.0  0.0  0.0  normal.
4  0  tcp  http  SF  217  2032   0  ...  0.02  0.0  0.0  0.0  0.0  0.0  normal.

[5 rows x 42 columns]
Detection Accuracy: 99.98%
Confusion Matrix:
[[ 29181     11]
 [    24 118991]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     29192
           1       1.00      1.00      1.00    119015

    accuracy                           1.00    148207
   macro avg       1.00      1.00      1.00    148207
weighted avg       1.00      1.00      1.00    148207

kunal@kunal-VirtualBox:~/Desktop/thesis$
```



Dataset