Ubiquitous Sensing and Smart City

Assignment Two

made by

Aisha Hagar
Khadija Hesham
Mohammed Elnamory

Supervised by

Prof. Burak Kantarci

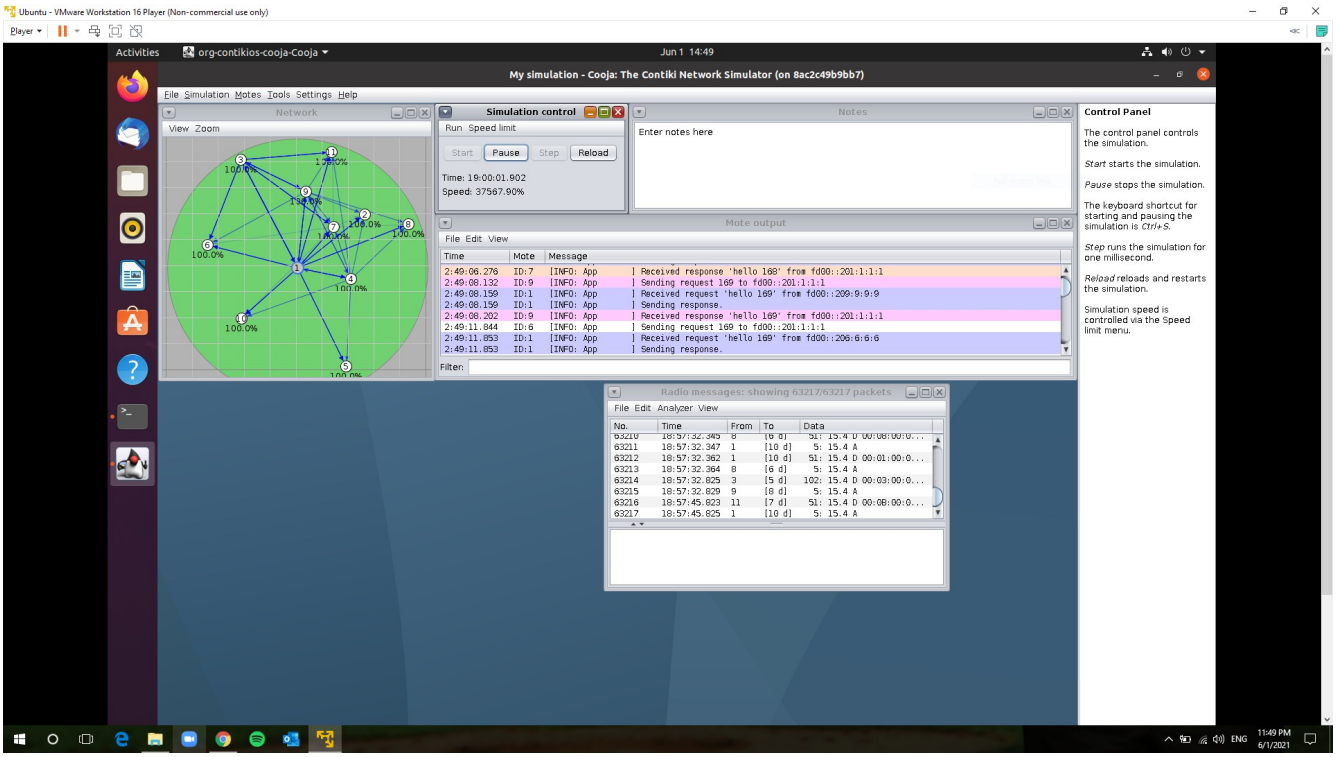
Table of content

1. Network Implementation:

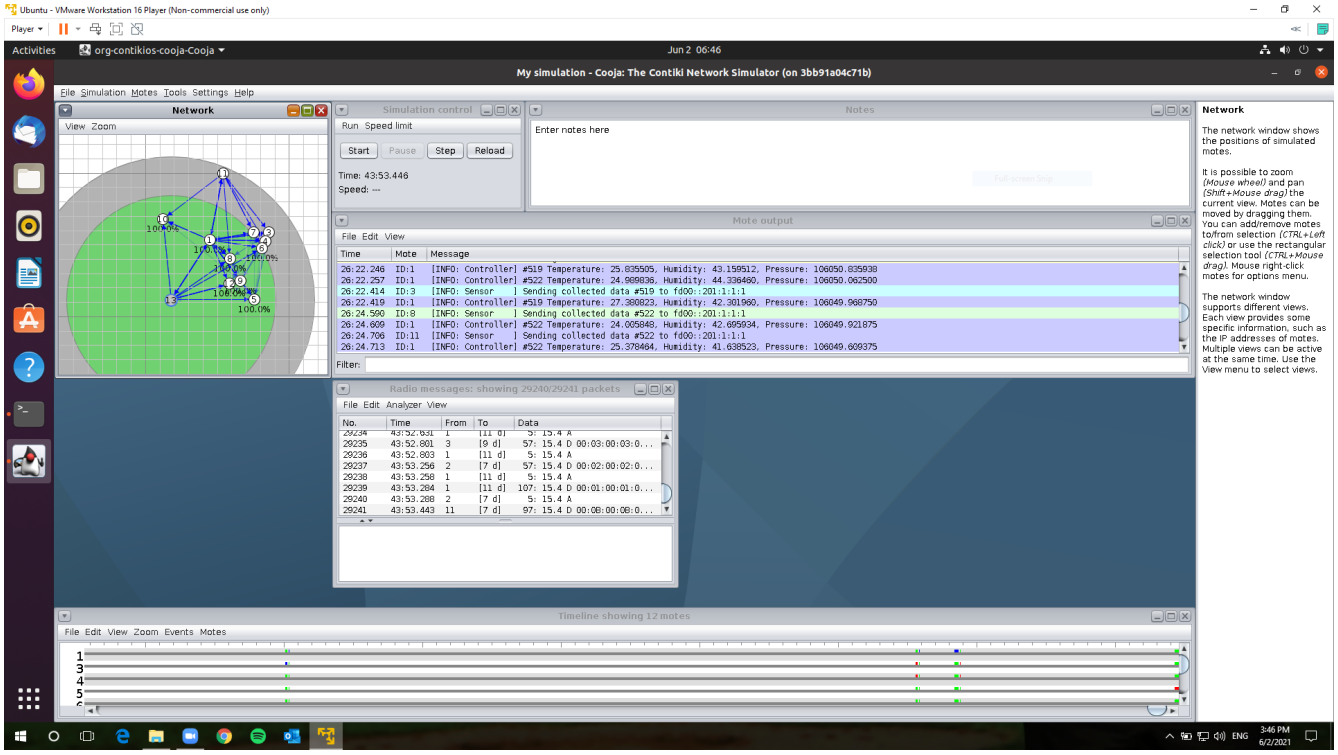
First of all, we used the upd_server file then, we used the upd_attacker1 for some time slicing to track the normal behaviour of the attacker.
After some time slicing, we have paused the simulation and replaced the upd_attacker1 with upd_attacker2 and started the
simulation again to track the malicious behaviour with the same IP address.

Network on the simulator:

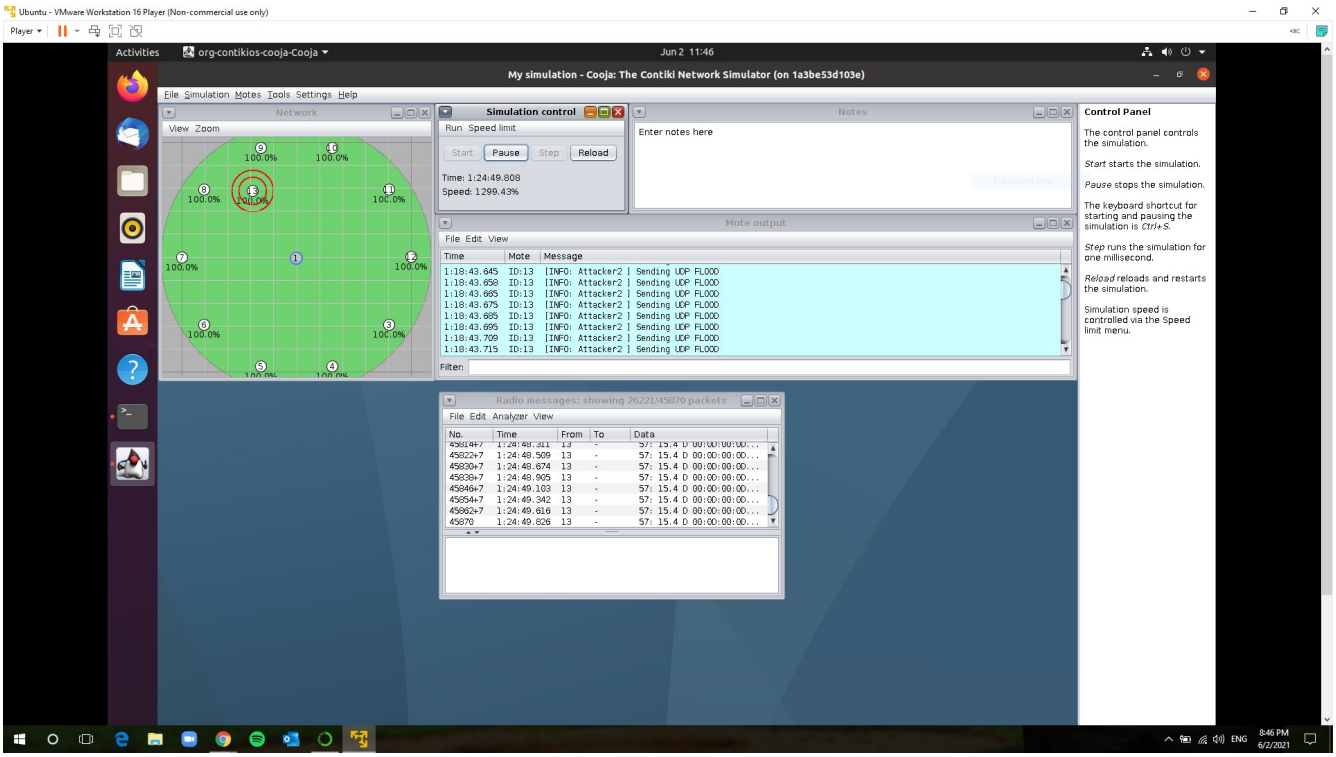
1. Ten attacker servers:



2. Flood attacker1:



3. Flood attacker2:



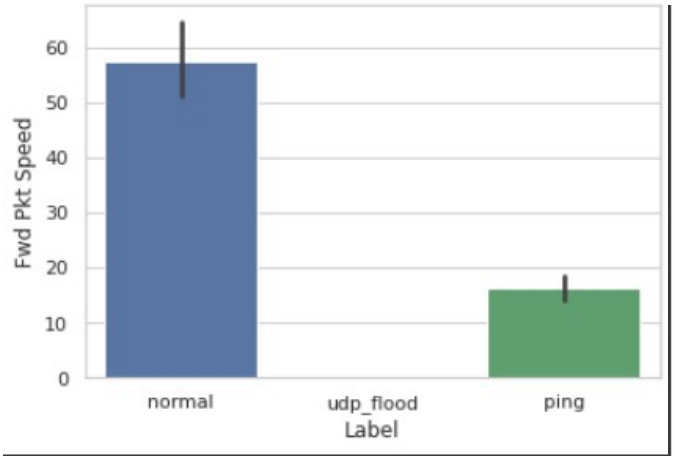
2. Data generation :

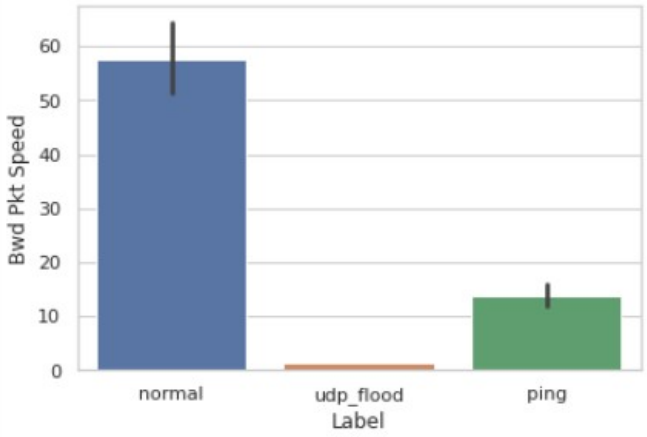
After simulating for three times one for the normal network without any attacker then the second simulation for flood attack and the third one for the Bing attack we used the implemented tool inside the Cooja simulator record tracker to generate a .pcap file for every simulation after the simulation process ended us used the provided python code to transform the pcap files into CSV files.

3. Data Preparation:

We have used Colab cloud service for data preparation and modeling.

We have plotted some insights:





It’s shown here is the normal class has the highest values of both Fwd and Bwd speeds.
As a data cleaning step, we have dropped data came from the hardware, null values and duplicated values.
We also have normalized the data before modeling in order to apply naive bayes classifier.
At last we have splitted the data set for training and testing for scale of 0.33 for testing.

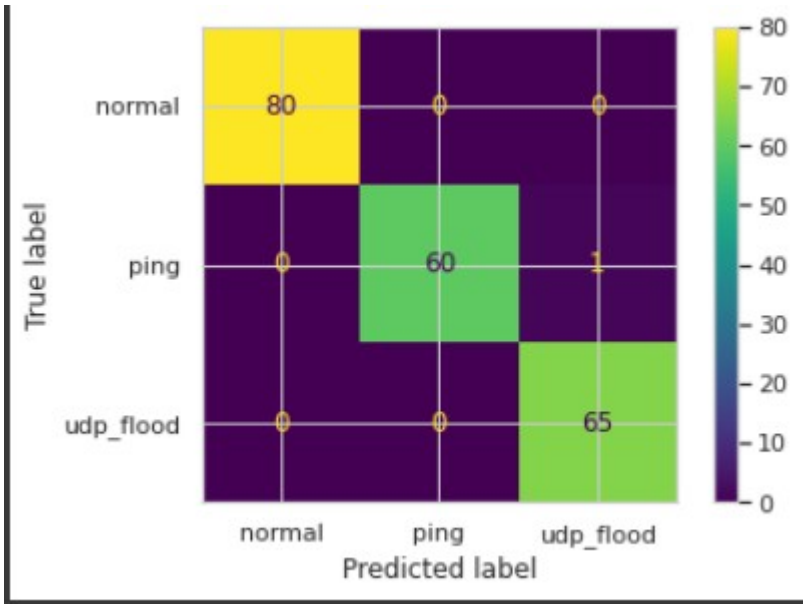
4. Modeling:

4.1 Random Forest:

We are instantiating a random forest object with 10 decision trees with maximum depth of 1 and using a subset of the data set each time.
And here is the macro metrics of the classifier:

```
Accuracy score:  0.9951456310679612
Precision score:  0.9949494949494949
Recall score:  0.9945355191256832
F1 score:  0.9947006498012744
```

Conclusion Matrix:



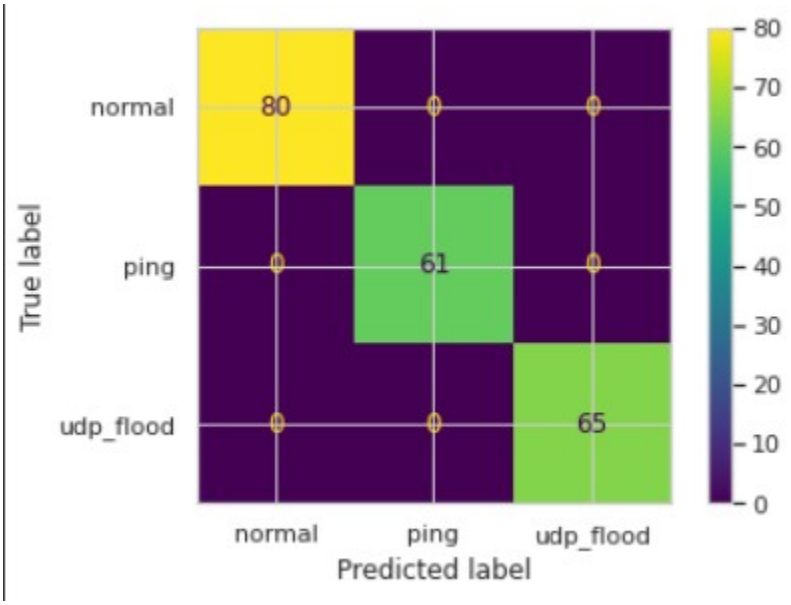
4.2 Naive Bayes:

We applied Gaussian naive bayes to the data, and it performed well to the data.

Metrics:

```
Accuracy score:  1.0
Precision score:  1.0
Recall score:  1.0
F1 score:  1.0
```

Conclusion Matrix:



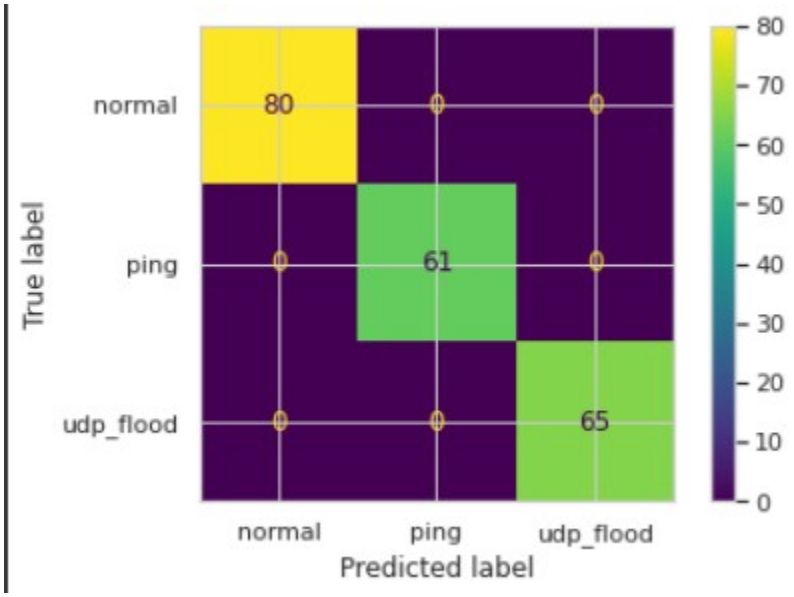
4.2 Adaboost:

We have instantiated an Adaboost object with 10 decision trees with maximum depth of 1 (by default). The overall performance of the algorithm is represent below:

Metrics:

```
Accuracy score:  1.0
Precision score:  1.0
Recall score:  1.0
F1 score:  1.0
```

Conclusion Matrix:



5. Conclusion:

We can conclude that both of Adaboost and Naive Bayes classifiers performed better that Random Forest algorithm.
We may need a larger data set for widening the scope of our learners and prevent overfitting issues.

Notebook used:

https://colab.research.google.com/drive/1a_8SFyKfJiXEIQf4QPyHeCAVY_Gf7nHa?usp=sharing#scrollTo=z-GTYBOFChxj

References:

[1] Lab code and lecture notes
[2] https://github.com/I-Good-Vegetable/NetworkFlowMeterOld
[3] https://scikit-learn.org/stable/modules/naive_bayes.html
[4] https://www.datacamp.com/community/tutorials/adaboost-classifier-python