# Internship Report

from 1st to 28 June 2019

**by**

**Saneï Nessrine**

# NETWORK TRAFFIC ANOMALY DETECTION

المجمع الكيميائي التونسي

GROUPE CHIMIQUE TUNISIEN

Internship supervisor :

Bedoui Mouafak

Pedagogical supervisor :

Drira Fadoua

# Gratitude

I would like to thank all the people and especially the internet who contributed to the success of my internship, who accompanied and helped me during the writing of this report.

I would like to thank my internship supervisor,Mr.Bedoui Mouafak, for his welcome, his services and his words that motivated me to work hard. Thanks also to his confidence I was able to accomplish  my missions all alone and developed my knowledge. His words will remain engraved in my memory "the engineer must be independent and autonomous ".

# Contents

# Table of figures

# I . Presentation of the reception structure and Description of tasks

## 1. Introduction

This chapter presents the activity of the Tunisian Chemical Group. It describes my assignment department, IT division, regional management of Gabes, and its various tasks.
The second part gives an overview of the tasks I did during the internship period.

## 2. General presentation of the GCT

The Tunisian Chemical Group (GCT) is a Tunisian public company whose purpose is to transform phosphate extracted in Tunisia into chemicals such as phosphoric acid or fertilizers.
GCT manufactures various types of granular, coated and uncoated fertilizers such as ammonium phosphates (MAP and DAP), ammonium nitrates (agricultural and porous NA), calcium phosphates (DCP) and triple superphosphate (TSP).

This company has 4 industrial phosphate processing sites located in Sfax and Mdhilla (for TSP production), in Gabes (for the production of Phosphoric Acid, DAP, DCP and AN) and Skhira (for the production of Phosphoric Acid).

In the various GCT plants, phosphoric acid is produced according to the SIAPE process developed in Tunisia.

Currently, the Ghannouch Industrial Zone (Gabes) is home to the GCT's largest production platform.

The products are exported through Sfax, Skhira and Gabes ports. These ports are also used for the import of raw materials such as sulfur and ammonia.

## 3. Organization of DCOSI and the IT Department

Since 1995, the IT Department has been created within the Chemical Group in Tunis. It is responsible for developing the GCT's main information systems (IS) policies and for steering and overseeing their implementation. This department is attached to the Central Management Organization and Information System and subsequently to the General Secretariat.

In addition to the IT department, the DCOSI includes an Archive and Documentation Department, an Organization and Upgrade Department, and an Information System Security Division (Figure 1).

The IT department contains 03 divisions (in Tunis, Gabes and Sfax) as well as 03 services (Industrial IT, network and IT service for the Mdhilla site).

# 4. Computer Division of Gabes

The computer division of Gabes is modernizing and deploying the information system in the GCT and especially the different sites in the region of Gabes : improving its productivity, managing interconnections, optimizing and securing the system.

This division is composed of three services : the Office service, the Application Administration service and the IT Center service.

## 4.1. Members of the team

The staff of the DI of Gabes includes 5 managers and 12 technicians among whom 05 technicians are distributed each in a site (01 for each factory and 02 for the socio-cultural center) in order to ensure the direct intervention and facilitate the user assistance.

## 4.2. Tasks of the DI

Among the tasks carried out by the DI of Gabes, we can mention:

- Ensure the operation, in the best conditions, of the hardware and software park
- Ensure the purchase of computer equipment and software.
- Supervise the infrastructure of computer networks and guarantee their operation and their security.
- Ensure the hierarchical supervision of all IT teams.
- Manage subcontracting: call for tenders, choice of service providers, management

# II. Network Traffic Anomaly Detection

## 1. Concept :

Cybersecurity is a growing research area with direct commercial impact to organizations and companies in every industry. With all other technological advancements in the Internet of Things (IoT), mobile devices, cloud computing, 5G network, and artificial intelligence, the need for Cybersecurity is more critical than ever before. These technologies drive the need for tighter Cybersecurity implementations, while at the same time act as enablers to provide more advanced security solutions.

In this project we discuss a framework that can predict Cybersecurity risk by identifying normal network behavior and detect network traffic anomalies. Our work focuses on the analysis of the historical network traffic data to identify network usage trends and security vulnerabilities. In doing so ,the data analytic , machine-learning tools and algorithms, such as Kmeans and Knn algorithms ,were utilized to assess Cybersecurity risk and identify baseline network behavior in order to detect abnormal patterns in network traffic.

## 2. Objective:

The goal of this project is to :

- generate a Data_set that fit the topic and deal with,
- analyze, detect and predict existing and emerging threats, network behaviors and alerting mechanisms while applying Knn and Kmeans clustering algorithms to a data_set.

# 3. Machine learning ML :

Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data ,ML allows computers to find hidden insights without being explicitly programmed where to look. There are 3 main types of ML algorithms :

- **Supervised learning** models are used when we have labeled data and are trying to predict a label based off of known features. Examples of Supervised Learning: Regression,Decision Tree, Randon Forest , KNN etc.

- **Unsupervised learning** models are used when we only have the input variables (X) and no corresponding output variables. They use unlabeled training data to model the underlying structure of the data. Examples of Unsupervised Learning: Apriori algorithm, K-means.

- **Reinforcement learning** is a type of machine learning algorithm that allows an agent to decide the best next action based on its current state by learning behaviors that will maximize a reward. Reinforcement algorithms usually learn optimal actions through trial and error.

Within this project and according to the framer ,we will use both of Knn and Kmeans clustering algorithms in order to detect the gct network traffic anomaly ,that are easier to implement with python . [1]

## 3.1. KNN (k- Nearest Neighbors) :

The K-Nearest Neighbors algorithm uses the entire data set as the training set, rather than splitting the data set into a training set and test set. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function. These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. If K = 1, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modeling. In our case we will find out that the best K is 34 .[2]

## 3.2. K-Means :

K-means is an iterative algorithm that groups similar data into clusters.It calculates the centroids of k clusters and assigns a data point to that cluster having least distance between its centroid and the data point. We start by choosing a value of k. Here, let us say k = 2 , we define 2 clusters ‘normal” and ‘malicious’. After, we randomly assign each observation to a cluster, and finds the centroid of each cluster.Then, the algorithm iterates through two steps: Reassign data points to the cluster whose centroid is closest. Calculate new centroid of each cluster.These two steps are repeated till the within cluster variation cannot be reduced any further. The within cluster variation is calculated as the sum of the euclidean distance between the data points and their respective cluster centroids. [2]

## 4. Tools:

**Wireshark** : **Wireshark** is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level. **Wireshark** has a rich feature set which includes the following: Deep inspection of hundreds of protocols, with more being added all the time,Live capture and offline analysis , Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility,Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000.. , Output can be exported to XML, PostScript®, CSV, or plain text ….[3]

**Tranalyzer** : Tranalyzer2 is a lightweight flow generator and packet analyzer designed for simplicity, performance and scalability. The program is written in C and built upon the libpcap library. It provides functionality to pre- and post-process IPv4/IPv6 data into flows and enables a trained user to see anomalies and network defects even in very large datasets. It supports analysis with special bit coded fields and generates statistics from key parameters of IPv4/IPv6 Tcpdump traces either being live-captured from an Ethernet interface or one or several pcap files. The quantity of binary and text based output of Tranalyzer2 depends on enabled modules, herein denoted as plugins. [4]

**Jupyter Notebook** : The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.[5]

**Python** : Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. [6]

## 5. Methodology :

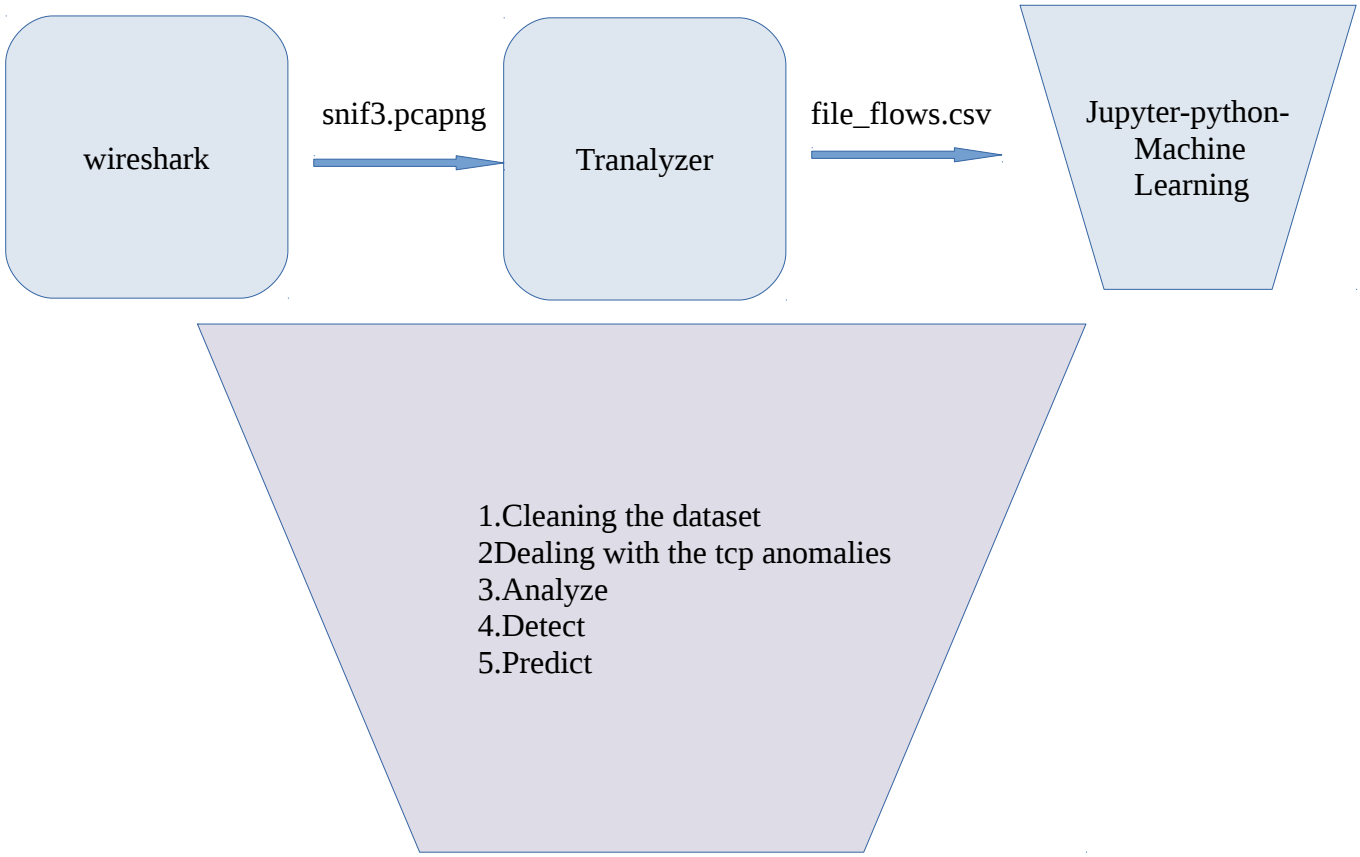


Figure 1- Sequences of work and methodology

## 5.1. Protocol-Based-Detection :

Protocol anomalies look at different trends in specific network protocols and send alerts when there are deviations. By analyzing network traffic for port-protocol anomalies, such as outgoing TCP connections to TCP 443 that is not SSL, you can effectively detect intrusions.

The anomaly detection is based on the specific threshold to avoid normal variation in the traffic. It is important that these baselines are re-evaluated and remodeled real-time to reflect regular changes in trend and seasonality.

Error Value = Forecasted value - Observed Value

Anomaly = I Error Value / Forecast Value I > 4

The threshold value can be adjusted to any reasonable ratio based on the volatility of the network trend. Here, we assume 4 as a ratio. It means if the absolute value of Error Value/Forecast Value is more than 4, it is considered an anomaly. In our case all this calculation is done with tranalyzer2 and generated in the ‘tcpAnomaly’ field .[7]

The figure bellow demonstrates the followed gait.

```
In [6]: # EXTRACTING MALICIOUS PACKETS BASED ON TCPANOMALY
        # TODO BASED ON MAC & ARP & UDP ANOMALIES !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        df_len = len(df_initial.index)
        anomaly = np.zeros(df_len)
        for i in range(df_len):
            tcp_anom = df_initial.iloc[i]['tcpAnomaly']
            cr = (int(tcp_anom,16))
            if cr > 4:
                anomaly[i] = 1
```

Figure 2- Extracting malicious packets based on tcpanomaly.

## 5.2. Sequential order of work :

### a. Importing needed libraries :

**Importing a library means loading it into the memory and then it's there for you to work with.**

**NumPy** is a package in **Python** used for Scientific Computing. **NumPy** package is used to perform different operations.

**Pandas** is a **Python** package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It allows opening a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc.

**Seaborn** is a **Python** data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

**Scapy** is a **Python** program that enables the user to send, sniff and dissect and forge network packets. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more.[8]

```
In [4]: from scapy.all import *
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        from datetime import datetime
```

Figure 3- Importing libraries

### b. Reading the csv file

```
In [5]: # READING CSV
        df_initial = pd.read_csv('files/file_flows.csv')
```

Figure 4- Reading csv file

### c. Extracting malicious packets based on TCP anomalies

### d. Adding anomaly column and modifying the dataframe

```
In [7]: # ADDING ANOMALY COLUMN CLASSIFIER
        df_initial.insert(2,'anomaly',anomaly)

In [8]: df_initial[df_initial['anomaly']>0]
```

Out[8]:

| | %dir | flowInd | anomaly | flowStat | timeFirst | timeLast | duration | numHdrDesc | numHdrs | hdrDes |
|---|---|---|---|---|---|---|---|---|---|---|
| 176 | A | 124 | 1.0 | 0x0000000000004000 | 1.560542e+09 | 1.560542e+09 | 23.897090 | 1 | 3 | sll:ipv4 |
| 177 | B | 124 | 1.0 | 0x0000000000004001 | 1.560542e+09 | 1.560542e+09 | 23.896848 | 1 | 3 | sll:ipv4 |

Figure 5- Dataframe modification

## e. Split the data into a training set and a testing set. Then train out model on the training set and then use the test set to evaluate the model.

We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable, in this case the “anomaly” column.

```
In [11]: from sklearn.model_selection import train_test_split

In [12]: # PREPARING TRAIN & TEST DATA
         X = df.drop('anomaly',axis=1)
         y = df['anomaly']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

Figure 6- train-test-split

## f. Use KNN to create a model that directly predicts a class for a new data point based off of the features.

```
In [13]: from sklearn.neighbors import KNeighborsClassifier

In [14]: # INSTANTIATING THE KNN MODEL
         knn = KNeighborsClassifier()
```

Figure 7- Knn implementation

## g. Fit the training data.

## h. Predict and evaluate the knn model.

```
In [16]: pred = knn.predict(X_test)

In [17]: from sklearn.metrics import classification_report, confusion_matrix

In [18]: # STATISTICS OF OUR KNN MODEL
         print(confusion_matrix(y_test,pred))
         print(classification_report(y_test,pred))
```

```
[[535   8]
 [ 21   9]]
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.97       543
         1.0       0.53      0.30      0.38        30

    accuracy                           0.95       573
   macro avg       0.75      0.64      0.68       573
weighted avg       0.94      0.95      0.94       573
```

Figure 8- Knn prediction_classification report_confusion matrix

## i. Create multiple model to get the best K parameter value .

```
In [19]: # CREATING MULTIPLE MODELS TO GET THE BEST K PARAMETER (DISTABCE) VALUE
         error_rate = []
         for i in range (1,60):

             knn = KNeighborsClassifier(n_neighbors=i)
             knn.fit(X_train,y_train)
             pred_i = knn.predict(X_test)
             error_rate.append(np.mean(pred_i != y_test))
```

Figure 9- Knn models

## g. Plot k vs the error rate and choose the  best K parameter with the lower error rate .

```
In [20]: #PLOTTING K VS ERROR RATE
plt.figure(figsize=(10,6))
plt.plot(range(1,60),error_rate,
         color='blue', linestyle='dashed',marker='o',
        markerfacecolor='red', markersize =10)
plt.title("Error Rate vs K Value")
plt.xlabel("K")
plt.ylabel("Error Rate")
```

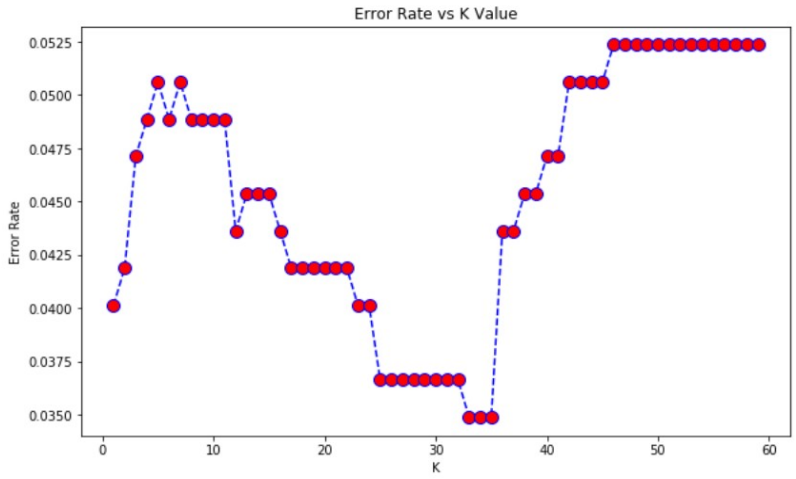


Figure 10- K parameter vs error rate

```
In [21]: # BEST CHOICE WITH LOW ERROR IS FOR K IN 32:36

In [22]: knn = KNeighborsClassifier(n_neighbors=34)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

[[542   1]
 [ 19  11]]
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.98       543
         1.0       0.92      0.37      0.52        30
```

Figure 11- Choosing best K parameter

## k. Applying the KMeans clustering algorithm.

```
In [23]: from sklearn.cluster import KMeans
```

Figure 12- KMeans implementation

## l. Define two clusters “normal” and “malicious”.

```
In [24]: kmeans = KMeans(n_clusters=2)

In [25]: kmeans.fit(df.drop('anomaly',axis=1))

Out[25]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)

In [27]: #the cluster center vectors
kmeans.cluster_centers_

Out[27]: array([[ 5.65674004e+02,  1.93179331e+01,  1.00000000e+00,
                  2.95335430e+00,  6.73322851e+00,  1.00000000e+00,
                  1.95285954e+04,  4.47127883e+01,  1.69737945e+01,
                  4.59052248e+04,  2.10586792e+03,  1.28600629e+01,
                  1.06348008e+02,  3.06912858e+01,  1.97770915e+01,
```

Figure 13- Clusters

## m. Visualize data.

```
In [29]: def converter(cluster):
             if cluster==1:
                 return 'malicious'
             else:
                 return 'normal'

In [30]: df['Kind'] = df['anomaly'].apply(converter)

In [31]: df.head()
```

Out[31]:

| :pRTTAckJitAve | icmpTCcnt | icmpEchoSuccRatio | icmpPFindex | connSip | connDip | connSipDip | connSipDprt | connF | Kind |
|---|---|---|---|---|---|---|---|---|---|
| l.0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 2.0 | normal |
| .0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1.0 | normal |
| l.0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 2.0 | normal |
| .0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1.0 | normal |
| l.0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 2.0 | normal |

Figure 14- Defining the clusters

```
In [46]: sns.set_style('darkgrid')
         graph = sns.FacetGrid(df,hue="Kind",palette='coolwarm',size=6,aspect=2)
         graph = graph.map(plt.hist,'anomaly',bins=20,alpha=0.7)
```

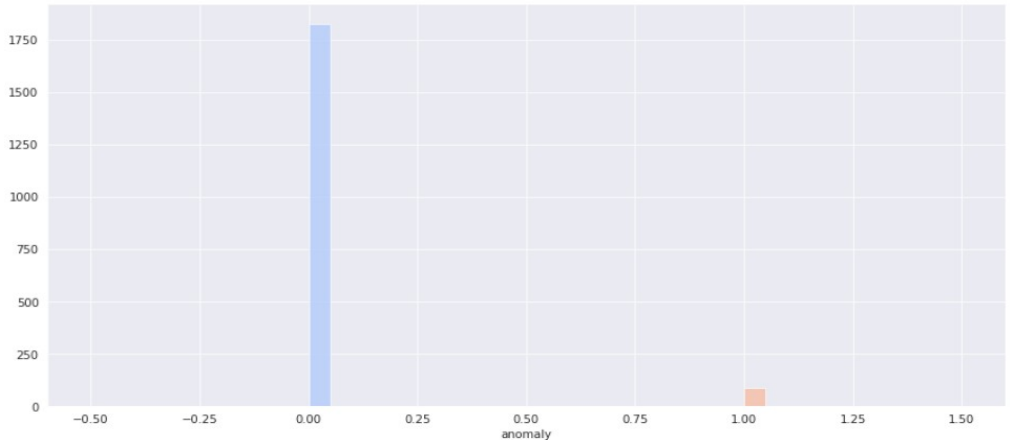


Figure 15- GCT network traffic ‘s anomalies

## 6. Conclusion :

As a result, we found out that 96% of the sniffed GCT network traffic are considered as normal network behaviors while 4% were considered as malicious anomalies.

# III.Conclusion

The results obtained so far show that, with simple heuristics, the most immediate positive result is a better knowledge of the functioning of the network and that trivial anomalies are easily identifiable.

The experiments were conducted with multiple approaches to get more insights into the network patterns and traffic trends to detect anomalies.

These results are very satisfactory for a project whose only claim was to understand a technology and its potential.

Afterwards, we will surely continue on this path to develop the missing bricks and explore not so trivial learning algorithms.

# References

[1] [Free Tutorials.Eu] Udemy - Python for Data Science and Machine Learning Bootcamp

[2] https://www.dataquest.io/blog/top-10-machine-learning-algorithms-for-beginners

[3] www.wireshark.org

[4] Introduction p14 from “Tranalyzer2 Version 0.8.4, Beta, Tarantula “ written by the Tranalyzer Development Team

[5] www.jupyter.org

[6] https://www.guru99.com/python-tutorials.html

[7] "Cyber Security Risk Analysis Framework - Network Traffic Anomaly Detection " by Lwin P. Moe

[8] www.python.org