3iL\ALLIANCE
</sciences et techniques informatiques>

Prog Avancée & Cybersécurité

Algo & Python

TP n°5

Analyse avancée de trafic

1 – Calculez :

a. la quantité d'information transmise dans un réseau par communication TCP ;

b. le débit de ces communications.

2 – Soit le code suivant :

```
def calculate_entropy(data):
    """Calculate the entropy of the given data."""
    if not data:
        return 0
    counter = {}
    for char in data:
        if char in counter:
            counter[char] += 1
        else:
            counter[char] = 1
    length = len(data)
    return -sum((count / length) * math.log(count / length, 2) for count in coun
ter.values())
```

Utilisez la mesure de l'entropie pour détecter la présence de paquet chiffré dans une capture réseau. Pouvez vous vous en servir pour savoir si un utilisateur fait du « *tunneling* » en DNS ou en ICMP ?

3 – Soit le code suivant :

```
#!/usrbin/env python3

import pandas as pd
import numpy as np
from pandas.io.formats.style import Styler

from http.server import BaseHTTPRequestHandler, HTTPServer


class RequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.end_headers()
        self.wfile.write(travail_pandas().encode('utf8'))

def run_server():
    server_address = ('', 8000)
    httpd = HTTPServer(server_address, RequestHandler)
    print("Server running at http://localhost:8000")
    httpd.serve_forever()

def travail_pandas():
    random_data = np.random.randint(10, 25, size=(5, 3))
    df = pd.DataFrame(random_data, columns=['COLUMN1', 'COLUMN2', 'COLUMN3'])
    return df.to_html()

run_server()
```

Créer une exportation Web pour un exercide de la fiche de TP 4.

4 – Soit le code suivant :

```
# Import required libraries
from scapy.all import *
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Packet handler to capture source and destination IPs
def packet_handler(packet):
    if packet.haslayer('IP'):
        return {
            'src_ip': packet['IP'].src,
            'dst_ip': packet['IP'].dst
        }

packets = rdpcap("nitroba.pcap")

# Extract source and destination IPs from packets
packet_data = [packet_handler(p) for p in packets if packet_handler(p)]

# Create a DataFrame with the IP communication data
df = pd.DataFrame(packet_data)

# Create a graph using NetworkX
G = nx.from_pandas_edgelist(df, source='src_ip', target='dst_ip')

# Calculate degree centrality (most connected nodes)
degree_centrality = nx.degree_centrality(G)

# Find the node with the highest degree centrality
most_important_node = max(degree_centrality, key=degree_centrality.get)
print(f"Most important node (highest degree centrality): {most_important_node}")

# Plot the network
plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G)  # Position nodes using a force-directed layout
nx.draw(G, pos, with_labels=True, node_size=700, node_color="lightblue", font_size=10)
nx.draw_networkx_nodes(G, pos, nodelist=[most_important_node], node_color='red',
node_size=800)  # Highlight most important node
plt.title("Network Graph - Most Important Node Highlighted")
plt.show()
```

Adaptez le à une utilisation de `tshark` à la place de Scapy.