**USER GUIDE :**

DZIME NKOMA HANDY FABRICE JULIEN B1

**For Program 1 in Python:**

import numpy as np

import os

import csv

import matplotlib.pyplot as plt

try:

    with open("DumpFile.txt", encoding="utf8") as fh:

        res = fh.read()

except FileNotFoundError:

    print("Le fichier n'existe pas :", os.path.abspath('DumpFile.txt'))

    exit()

ress = res.split('\n')

tab\_dest = np.array([])

with open("File.csv", "w", newline='') as fic:

    # test est le fichier d'arrivée des extractions

    writer = csv.writer(fic)

    evenement = ["DATE", "SOURCE", "PORT", "DESTINATION", "FLAG", "SEQ", "ACK", "WIN", "OPTIONS", "LENGTH"]

    writer.writerow(evenement)  # écriture de mes titres dans le tableau

    characters = ":"  # définir une variable avec le caractère ":" (qui nous sera utile pour la suite)

    for event in ress:

        if event.startswith('11:42'):

            # déclaration variables et remise à zéro

            seq = ""

            heure1 = ""

            nomip = ""

            port = ""

            flag = ""

            ack = ""

            win = ""

            options = ""

            length = ""

            # Pour la date de l'événement (première colonne)

            texte = event.split(" ")

            heure1 = texte[0]

            # Pour la source (2ème colonne)

            nomip1 = texte[2].split(".")

            nomip = ".".join(nomip1[:min(5, len(nomip1))])  # Utilisation de join pour construire l'IP

            flag2 = nomip not in tab\_dest

            if flag2:

                tab\_dest = np.append(tab\_dest, nomip)

            # Port

            port1 = texte[2].split(".")

            port = port1[-1]

            # Pour la destination (3ème colonne)

            nomip2 = texte[4]

            # Flag

            texte = event.split("[")

            if len(texte) > 1:

                flag1 = texte[1].split("]")

                flag = flag1[0]

            # Seq

            texte = event.split(",")

            if len(texte) > 1 and texte[1].startswith(" seq"):

                seq1 = texte[1].split(" ")

                seq = seq1[2]

            # Ack

            if len(texte) > 2 and texte[2].startswith(" ack"):

                ack1 = texte[2].split(" ")

                ack = ack1[2]

            elif len(texte) > 1 and texte[1].startswith(" ack"):

                ack1 = texte[1].split(" ")

                ack = ack1[2]

            # Win

            if len(texte) > 3:

                if texte[3].startswith(" win"):

                    win1 = texte[3].split(" ")

                    win = win1[2]

                elif texte[2].startswith(" win"):

                    win1 = texte[2].split(" ")

                    win = win1[2]

            # Options

            texte = event.split("[")

            if len(texte) > 2:

                options1 = texte[2].split("]")

                options = options1[0]

            # Length (avec option)

            texte = event.split("]")

            if len(texte) > 2:

                length1 = texte[2].split(" ")

                length = length1[2].replace(characters, "")

            # Length (sans option)

            texte = event.split(",")

            if len(texte) > 3 and texte[3].startswith(" length"):

                length1 = texte[3].split(" ")

                length = length1[2]

            # Écrire dans le fichier CSV

            evenement = [heure1, nomip, port, nomip2, flag, seq, ack, win, options, length]

            writer.writerow(evenement)

print("Tableau final", tab\_dest)

# Tracer le graphique

plt.plot(tab\_dest, [1] \* len(tab\_dest))

plt.show()

**User Guide for Python Code and CSV File (Excel):**

This Python script is designed to process a text file (DumpFile.txt) containing network data, extract specific information, and then store it in a CSV file (file.csv). The script utilizes the NumPy library for array manipulation and the CSV module for CSV file management.

**Usage of the Python Code:**

1. **Input File:**
   * Place your text file (DumpFile.txt) in the same directory as the Python script.
   * Une image contenant texte, capture d’écran, Police, ligne

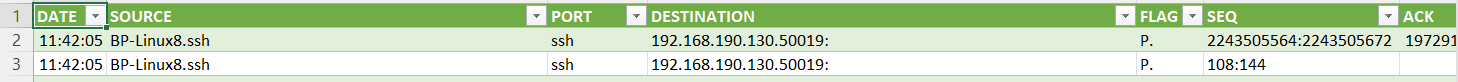
     Description générée automatiquement



1. **Execution:**
   * Run the Python script from a terminal or a Python development environment (such as Spyder).
   * The code will read the DumpFile.txt file, extract specific information, and generate a CSV file (file.csv).
2. **Results:**
   * A CSV file will be created with columns such as "DATE," "SOURCE," "PORT," "DESTINATION," "FLAG," "SEQ," "ACK," "WIN," "OPTIONS," and "LENGTH."
   * 
3. The extracted information from network events starting at "11:42" will be organized in this file.
4. **CSV File Contents (Excel):**
   * The file file.csv can be opened with spreadsheet software like Excel.
   * Each row in the CSV file corresponds to a network event, with the extracted information distributed across the columns.
5. **Error Handling:**
   * If the file "DumpFile.txt" is not found, the script will display a message indicating that the file does not exist.
6. **User Guide for the CSV File (Excel) Generated by the Python Code:**

The CSV file generated by the Python script, named file.csv, can be utilized to visualize and analyze the network information extracted from the text file. Here's how to use this CSV file with spreadsheet software such as Excel:

**Usage of the CSV File (Excel):**

1. **File Location:**
   * The CSV file (file.csv) will be created in the same directory as the Python script. Make sure to locate it.
2. **Opening with Excel:**
   * Double-click on the file file.csv, or open Excel and use the "Open" option to select the file.
3. **Data Format:**
   * The CSV file is structured with columns corresponding to different extracted information, such as "DATE," "SOURCE," "PORT," "DESTINATION," "FLAG," "SEQ," "ACK," "WIN," "OPTIONS," and "LENGTH."
   * 
   * Une image contenant texte, capture d’écran, Police, nombre

     Description générée automatiquement
4. **Exploring the Data:**
   * Explore the data using Excel's features. You can sort, filter, and perform other operations as per your requirements.
5. **Columns and Content:**
   * Each row in the file corresponds to an extracted network event, and each column represents a specific piece of information associated with that event.
6. **Customization:**
   * Modify, add, or delete data in Excel according to your needs. The CSV file serves as a starting point for your analyses.
7. **Saving Changes:**
   * If you make any modifications in Excel, ensure to save the file in CSV format if you want to preserve those changes.

**For the Python program 2:**

import csv

import webbrowser

import matplotlib.pyplot as plt

# Ouvrir le fichier "extrait.txt"

with open("DumpFile.txt", "r") as fichier:

    ipsr = []

    ipde = []

    longueur = []

    flag = []

    seq = []

    heure = []

    flagcounterP = 0

    flagcounterS = 0

    flagcounter = 0

    framecounter = 0

    requestcounter = 0

    replycounter = 0

    seqcounter = 0

    ackcounter = 0

    wincounter = 0

    for ligne in fichier:

        split = ligne.split(" ")

        if "IP" in ligne:

            framecounter += 1

            if "[P.]" in ligne:

                flag.append("[P.]")

                flagcounterP += 1

            elif "[.]" in ligne:

                flag.append("[.]")

                flagcounter += 1

            elif "[S]" in ligne:

                flag.append("[S]")

                flagcounterS += 1

            if "seq" in ligne:

                seqcounter += 1

                seq.append(split[8])

            if "win" in ligne:

                wincounter += 1

            if "ack" in ligne:

                ackcounter += 1

            ipsr.append(split[2])

            ipde.append(split[4])

            heure.append(split[0])

            if "length" in ligne:

                split = ligne.split(" ")

                longueur.append(split[-2] if "HTTP" in ligne else split[-1])

            if "ICMP" in ligne:

                if "request" in ligne:

                    requestcounter += 1

                elif "reply" in ligne:

                    replycounter += 1

# Ajouter une vérification pour éviter la division par zéro

globalreqrepcounter = replycounter + requestcounter

if globalreqrepcounter != 0:

    req = requestcounter / globalreqrepcounter

    rep = replycounter / globalreqrepcounter

else:

    req = rep = 0

globalflagcounter = flagcounter + flagcounterP + flagcounterS

P = flagcounterP / globalflagcounter

S = flagcounterS / globalflagcounter

A = flagcounter / globalflagcounter

flagcounter = [flagcounter]

flagcounterP = [flagcounterP]

flagcounterS = [flagcounterS]

framecounter = [framecounter]

requestcounter = [requestcounter]

replycounter = [replycounter]

seqcounter = [seqcounter]

ackcounter = [ackcounter]

wincounter = [wincounter]

# ... (votre code existant)

# Créer le graphique semi-circulaire pour les drapeaux

name = ['Flag [.]', 'Flag [P]', 'Flag [S]']

data = [A, P, S]

colors = ['#2196F3', '#FFC107', '#4CAF50']  # Blue, Yellow, Green

explode = (0, 0, 0)

fig, ax = plt.subplots(figsize=(8, 6))  # Ajuster la taille de la figure ici

ax.pie(data, explode=explode, labels=name, autopct='%1.1f%%', startangle=90, shadow=True, colors=colors, wedgeprops=dict(width=0.4))

ax.axis('equal')  # Assurer que le graphique soit circulaire

plt.savefig("graphe1.png")

plt.show()

# Créer le graphique semi-circulaire pour les requêtes et réponses

name2 = ['Request', 'Reply']

data2 = [req, rep]

colors2 = ['#4CAF50', '#FFC107']  # Green, Yellow

explode = (0, 0)

fig, ax = plt.subplots(figsize=(8, 6))  # Ajuster la taille de la figure ici

ax.pie(data2, explode=explode, labels=name2, autopct='%1.1f%%', startangle=90, shadow=True, colors=colors2, wedgeprops=dict(width=0.4))

ax.axis('equal')  # Assurer que le graphique soit circulaire

plt.savefig("graphe2.png")

plt.show()

# ... (le reste de votre code)

# Contenu de la page web

# ... (votre code existant)

# Contenu de la page web

htmlcontenu = '''

<html lang="fr">

   <head>

      <meta charset="utf-8">

      <title> Traitement des données </title>

      <style>

      body{

          background-image: url('https://www.codeur.com/blog/wp-content/uploads/2021/08/image-programmation-1.jpg');

          background-repeat: no-repeat;

          background-size: cover;

          background-color: violet;  /\* Ajout de cette ligne pour le fond violet \*/

          color: #e5f2f7;

          background-attachment: fixed;

      }

      </style>

   </head>

   <body>

       <center><h1>Julien Nkoma</h1></center>

       <center><h2>Projet SAE 15</h2></center>

       <center><p>Sur cette page web, nous vous présentons les informations et données pertinentes trouvées dans le fichier à traiter.</p></center>

       <center><h3> Nombre total de trames échangées</h3> %s</center>

       <br>

       <center><h3> Drapeaux (Flags)<h3></center>

       <center>Nombre de flags [P] (PUSH) = %s

       <br>Nombre de flags [S] (SYN) = %s

       <br>Nombre de flag [.] (ACK) = %s

       <br>

       <br>

       <img src="graphe1.png">

       <h3> Nombre de requêtes et réponses </h3>

       Request = %s

       <br>

       Reply = %s

       <br>

       <br>

       <img src="graphe2.png">

       <h3>Statistiques entre seq, win et ack </h3>

       Nombre de seq = %s

       <br>

       Nombre de win = %s

       <br>

       Nombre de ack = %s

   </body>

</html>

''' % (framecounter[0], flagcounterP[0], flagcounterS[0], flagcounter[0], requestcounter[0], replycounter[0], seqcounter[0], wincounter[0], ackcounter[0])

# ... (le reste de votre code)

# Contenu de la page web en Markdown

markdown\_content = f'''

# Julien Nkoma

## Projet SAE 15

Sur cette page web, nous vous présentons les informations et données pertinentes trouvées dans le fichier à traiter.

### Nombre total de trames échangées

{framecounter[0]}

### Drapeaux (Flags)

- Nombre de flags [P] (PUSH) : {flagcounterP[0]}

- Nombre de flags [S] (SYN) : {flagcounterS[0]}

- Nombre de flag [.] (ACK) : {flagcounter[0]}

![Graphique Drapeaux](graphe1.png)

### Nombre de requêtes et réponses

- Request : {requestcounter[0]}

- Reply : {replycounter[0]}

![Graphique Requêtes/Réponses](graphe2.png)

### Statistiques entre seq, win et ack

- Nombre de seq : {seqcounter[0]}

- Nombre de win : {wincounter[0]}

- Nombre de ack : {ackcounter[0]}

'''

# Ouvrir un fichier Markdown pour les données extraites du fichier texte non traité

with open('donnees.md', 'w', newline='') as fichiermd:

    fichiermd.write('# Données extraites du fichier texte non traité\n\n')

    fichiermd.write('| Heure | IP source | IP destination | Flag | Seq | Length |\n')

    fichiermd.write('|-------|-----------|-----------------|------|-----|--------|\n')

    for row in zip(heure, ipsr, ipde, flag, seq, longueur):

        fichiermd.write(f'| {" | ".join(map(str, row))} |\n')

# Ouvrir un fichier Markdown pour les statistiques générales

with open('Stats.md', 'w', newline='') as fichier2md:

    fichier2md.write('# Statistiques générales\n\n')

    fichier2md.write('| Flag[P] (PUSH) | Flag[S] (SYN) | Flag[.] (ACK) | Nombre total de trames | Nombre de request | Nombre de reply | Nombre de sequence | Nombre de acknowledg | Nombre de window |\n')

    fichier2md.write('|-----------------|-----------------|-----------------|------------------------|---------------------|------------------|----------------------|------------------------|------------------|\n')

    for row in zip(flagcounterP, flagcounterS, flagcounter, framecounter, requestcounter, replycounter, seqcounter, ackcounter, wincounter):

        fichier2md.write(f'| {" | ".join(map(str, row))} |\n')

# Ouvrir un fichier Markdown pour la page web

with open("webpage.md", "w") as md:

    md.write(markdown\_content)

    print("Page web en Markdown créée avec succès.")

**User Guide for Python Code and Generated CSV Files:**

This Python script is designed to process a text file (DumpFile.txt), extract specific information, and generate two CSV files (data.csv and Stats.csv). The script utilizes the csv libraries for CSV file manipulation and matplotlib for generating graphs.

**Usage of the Python Code:**

1. **Input File:**
   * Place your text file (DumpFiler.txt) in the same directory as the Python script.
   * Une image contenant texte, Police, nombre, logiciel

     Description générée automatiquement
2. **Execution:**
   * Run the Python script from a terminal or a Python development environment (such as Spyder).
   * The code will read the "DumpFile.txt" file, perform statistical analysis, generate two CSV files (data.csv and Stats.csv), along with two graphs, and the Markdown page (markdown.md) containing information.

**Generated CSV File :**

**data.csv:**



* Contains the extracted information from the text file, such as time, source IP, destination IP, flag, sequence, and length.
* Can be opened with spreadsheet software like Excel.

**Statistics.csv File:**



* Contains general statistics such as the count of [P] (PUSH), [S] (SYN), [.] (ACK) flags, the total number of frames, the count of requests, the count of responses, the count of sequences, the count of acknowledgments (ack), and the count of windows.
* Can be opened with spreadsheet software like Excel.

**Generated Graph 1 (graph1.png):**

Une image contenant cercle, capture d’écran, Graphique, Caractère coloré

Description générée automatiquement

* Circular graph representing the distribution of [P] (PUSH), [S] (SYN), [.] (ACK) flags.
* Saves the graph in PNG format.

**Generated Graph 2 (graph2.png):**

Une image contenant cercle, capture d’écran, Graphique, Caractère coloré

Description générée automatiquement

* Circular graph representing the distribution of requests and responses.
* Saves the graph in PNG format.

**Contenu du Fichier Markdown (markdown.md) :**

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

* Presents relevant information and data found in the processed file.
* Includes the total number of frames, statistics on flags, the count of requests and responses, as well as statistics on sequences, windows, and acknowledgments.

By following these steps, you should be able to use the Python script to extract network information from the text file, generate CSV files, and analyze the data using the generated files and graphs.

I was able to convert this Markdown file to an HTML file using the Dillinger converter.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, cercle, graphisme

Description générée automatiquement