МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
Университет ИТМО

**ФАКУЛЬТЕТ БЕЗОПАСНОСТИ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Лабораторная работа №2
по теме «Обработка и тарификация трафика NetFlow»
по предмету «Управление мобильными устройствами»

Работу выполнил
Студент группы №3347
очного отделения:

03.04.20 Якимов Ярослав

Проверил

_____
Федоров И.Р.

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2020

Цель работы: познакомиться с протоколом NetFlow и тарификацией данных, исходя из трафика пользователя.

Задачи:

- Парсинг файла с дампом трафика и формирование читабельного файла
- Тарификация выбранного пользователя по своему варианту
- Построение графика зависимости объема трафика от времени

Средства реализации: для реализации был выбран язык программирования Python, как отвечающий всем современным требованиям в разработке. Кроме того, он предоставляет возможность написать быстрый код, как по времени разработки, так и по времени исполнения. Тем более он имеет встроенные модули для работы с файлами json и библиотеки для формирования удобных и читабельных графиков – plotly и numpy.

Исходный код (так же по ссылке):

```
import json

import time

import numpy as np

import plotly.graph_objects as go

def drawDiagram(in_traffic, out_traffic, person_ip, show=True):

    in_traffic['dates'].sort()

    in_traffic['volume'].sort()

    out_traffic['dates'].sort()

    out_traffic['volume'].sort()

    fig = go.Figure()

    fig.add_trace(go.Scatter(

        x=in_traffic['dates'], y=in_traffic['volume'],
name="Входящий трафик"))

    fig.add_trace(go.Scatter(

        x=out_traffic['dates'], y=out_traffic['volume'],
name="Исходящий трафик"))

    fig.add_trace(go.Scatter(

        x=out_traffic['dates'], y=[x + y for x, y in
zip(in_traffic['volume'], out_traffic['volume'])], name="Общий
трафик"))

    fig.update_layout(title=f'Входящий и исходящий трафик по IP
{person_ip}',

                      xaxis_title='Дата',

                      yaxis_title='Объем трафика (в байтах)')

    fig.show() if show else ''

def main(person_ip):

    in_traffic = {'sum': 0, 'dates': [], 'volume': []}

    out_traffic = {'sum': 0, 'dates': [], 'volume': []}

    with open('dump.json', "r") as f:

        data = json.load(f)


    for item in data:

        if 'src4_addr' in item:
```

```
            if item['src4_addr'] == person_ip:
                out_traffic['sum'] += int(item['in_bytes'])
                out_traffic['dates'].append(item['t_first'])
                out_traffic['volume'].append(out_traffic['sum'])
            elif item['dst4_addr'] == person_ip:
                in_traffic['sum'] += int(item['in_bytes'])
                in_traffic['dates'].append(item['t_first'])
                in_traffic['volume'].append(in_traffic['sum'])
    drawDiagram(in_traffic, out_traffic, person_ip)
    sum_traffic_kbytes = (in_traffic['sum'] +
out_traffic['sum']) / 1024
    sum_traffic_bytes = (in_traffic['sum'] + out_traffic['sum'])
    if sum_traffic_kbytes > 200:
        billing_kbytes = (sum_traffic_kbytes - 200) * 1 + 200 *
0.5
    else:
        billing_kbytes = sum_traffic_kbytes*0.5
    if sum_traffic_bytes > 200:
        billing_bytes = (sum_traffic_bytes - 200) * 1 + 200 *
0.5
    else:
        billing_bytes = sum_traffic_bytes*0.5
    return billing_kbytes, billing_bytes
person_ip = "192.0.73.2"
start_time = time.time()
billing_kbytes, billing_bytes = main(person_ip)
with open("result.txt", "w", encoding='utf-8') as f:
    f.write(
        f'Total billing if we use Kbytes: {billing_kbytes}
rub.\nTotal billing if we use Bytes: {billing_bytes} rub.\nTook
{time.time() - start_time} seconds',)
```

Выводы: Я познакомился с форматом NetFlow и научился тарифицировать абонентов, на основании файлов с записями об их трафике.