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:

Якимов Ярослав

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Федоров И.Р.



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](https://github.com/coltmister/MobileDevYakimov)):

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.