Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ «Интеллектуальные технологии»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«21» 06 2023 г.

Декан _____

_____ И.А. Трещёв
(подпись)

«21» 06 2023 г.

УТВЕРЖДАЮ

Проректор по научной работе

_____ А.В. Космынин
(подпись)

«21» 06 2023 г.

«Использование моделей машинного обучения для обнаружения сканирования сети»
Комплект *проектной* документации

Руководитель СКБ _____ 21.06.2023 *Г.В. Москалец*
(подпись, дата)

Руководитель проекта _____ 21.06.2023 *Г.В. Москалец*
(подпись, дата)

Комсомольск-на-Амуре 2023

Карточка проекта

Название	Использование нейронных сетей для прогнозирования угроз информационной безопасности
Тип проекта	Тип проекта: техническое творчество (инициативный)
Исполнители	Студент _____ Г.В. Москалец – 8ИБ-1 Студент _____ Е. И. Монастырная – 0ИБ-1
Срок реализации	01.11.2022-30.05.2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ
на разработку

Название проекта: «Использование нейронных сетей для прогнозирования угроз информационной безопасности»

Назначение: Разработана нейронная сеть для прогнозирования угроз информационной безопасности. и разработано программное обеспечение для прогнозирования информационной безопасности предприятия.
Область использования: Может применеятся в области кибербезопасность для анализа и прогнозирования угроз в компьютерных сетях предприятий, обнаружения аномалий и потенциальных атаки.В финансовой сфере может быть использовано для обнаружения мошеннических операций, предсказания атак и угроз финансовым организациям.
Функциональное описание устройства: Программа разработана для автоматизации обнаружения сканирования сети, а также для упрощения работы ИБ отдела предприятия.
Техническое описание устройства: Программа состоит из 3 блоков программного кода, основные части отвечают за сбор трафика, его обработку и вывод о собранном трафике.
Требования: Intel core 2 DUO и выше, Ubuntu 20, не менее 2ГБ свободной памяти
План работ:

Наименование работ	Срок
Изучение существующих методов обнаружения трафика	*10.2022*
Опредление основных функций программы	*10.2022*
Проктирование общей структуры программы	*11.2022*
Разработка модуля сбора и обработки трафика	*11.2022*
Разработка модуля прогнозирования временного ряда	*01.2023*
Разработка модуля бинарной классификации	*02.2023*
Тестирование и отладка	*03.2023*

Комментарии:

Перечень графического материала:

1. Листинги;
2. Изображения;

Руководитель проекта 21.06.2023 *Г.В. Москалец*

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ

«Использование нейронных сетей для прогнозирования угроз информационной безопасности»

Руководитель проекта _____ 21.06.2023 *Г.В. Москалец*

(подпись, дата)

Комсомольск-на-Амуре 2023

Содержание

Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основными техническими характеристиками, устройством, правилами установки и эксплуатации устройства «Использование нейронных сетей для прогнозирования угроз информационной безопасности» (далее «изделие»).

Паспорт входит в комплект поставки изделия. Прежде, чем пользоваться изделием, внимательно изучите правила обращения и порядок работы с ним. В связи с постоянной работой по усовершенствованию изделия, повышающей его надежность и улучшающей условия эксплуатации, в конструкцию могут быть внесены изменения, не отраженные в данном издании.

1.1 Наименование изделия

Полное наименование изделия – «Использование нейронных сетей для прогнозирования угроз информационной безопасности».

1.2 Наименования документов, на основании которых ведется проектирование изделия

Проектирвание «Использование нейронных сетей для прогнозирования угроз информационной безопасности» осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке изделия

Заказчиком проекта «Использование нейронных сетей для прогнозирования угроз информационной безопасности» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 17.

Исполнителями проекта «Использование нейронных сетей для прогнозирования угроз информационной безопасности» являются участники

студенческого конструкторского бюро «Интеллектуальные технологии», студенты групп 8ИБ-1 Москалец Георгий Вадимович

Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 2.001-2013. Единая система конструкторской документации. Общие положения.

ГОСТ 2.102-2013. Единая система конструкторской документации. Виды и комплектность конструкторских документов.

ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам.

ГОСТ 2.610-2006. Единая система конструкторской документации. Правила выполнения эксплуатационных документов.

ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

ГОСТ 2.051-2006. Единая система конструкторской документации. Электронные документы. Общие положения.

ГОСТ 2.052-2006. Единая система конструкторской документации. Электронная модель изделия. Общие положения.

ГОСТ 2.601-2013. Единая система конструкторской документации. Эксплуатационные документы.

2 Назначение и принцип действия

2.1 Назначение изделия

Использование нейронных сетей для прогнозирования угроз информационной безопасности

В состав изделия входят:

- Паспорт,
- Программная реализация.

2.2 Области использования изделия

Может применеятся в области кибербезопасность для анализа и прогнозирования угроз в компьютерных сетях предприятий, обнаружения аномалий и потенциальных атаки.В финансовой сфере может быть использовано для обнаружения мошеннических операций, предсказания атак и угроз финансовым организациям.

2.3 Принцип действия изделия

Программа запускается на виртуальной машине Ubuntu в терминале. Она собирает трафик в течении 15 секунд, после чего на основе собранного датасета и обучения выдвигает оценку о сканировании сети. Если решает, что сканирование есть, то отключает маршрутизацию тем самым прекращает дальнейшее сканирование сети злоумышленником.

3 Состав изделия и комплектность

В комплект поставки входит:

- Паспорт,
- Программная реализация.

4 Устройство и описание работы изделия

4.1 Описание работы изделия

Запуск программы начинается при вводе в терминале определённой команды. Далее происходит сбор трафика в течении 15 секунд, чтобы потом прогнозировать на основе этого трафика временной ряд.

После этого обработанные данные по трафику отправляются в модуль бинарной классификации, который определяет класс трафика как сканированный или не сканированный.

Если программа определяет в трафике следы сканирования, то она отключает маршрутизацию между клиентами, тем самым останавливая сканирование и дальнейшую атаку.

5 Условия эксплуатации

Изделие выпускается в климатическом исполнении УХЛ 4.2 по ГОСТ 15150-69 и предназначен для использования в стационарных условиях в закрытых помещениях при соответствующих климатических условиях:

- интервал температур от +10 до +35 °С;
- относительная влажность воздуха до 80 % при температуре +25 °С;
- высота над уровнем моря не более 2000 м;
- атмосферное давление от 86,6 до 106 кПа (от 650 до 800 мм рт. ст.).

В помещении, где используется изделие не должно возникать условий для конденсации влаги (выпадения росы). *Изделие является электронным прибором, требующим бережного обращения.*

Для обеспечения безотказной работы, сохранения точности и его сбережения необходимо соблюдать следующие правила:

- изучить паспорт, прежде чем приступить к работе с изделием;
- *при необходимости указать дополнительные пункты*
- не допускать самостоятельную разборку изделия.

5.1 Правила и особенности размещения изделия

Изделие должно быть расположено на расстоянии не менее 1 м от нагревательных приборов.

ВНИМАНИЕ! При эксплуатации изделия запрещается проводить самостоятельно какие-то либо работы по извлечению и установке внутренних компонентов изделия.

5.2 Меры безопасности

Необходимо соблюдать требования техники безопасности и следующие меры предосторожности:

- *не оставлять изделие включенным без наблюдения;*

- *отладка программы должна производиться только квалифицированными специалистами;*

5.3 Правила хранения и транспортирования

Транспортирование изделия в упакованном виде может производиться железнодорожным, автомобильным (в закрытых транспортных средствах), воздушным, речным и морским видами транспорта в соответствии с правилами перевозок грузов, действующих на транспорт данного вида. Условия транспортирования изделия по части воздействия климатических факторов должны соответствовать группе 5 по ГОСТ 15150.

После транспортирования изделие должно быть выдержано не менее 2 часов в транспортной таре при температуре 20±5 °С и относительной влажности воздуха не более 80 %.

Распакованное изделие должно храниться в отапливаемом и вентилируемом чистом помещении при температуре от +5 до +40 °С и относительной влажности воздуха не более 60 %. При температуре ниже 25 °С допускается увеличение относительной влажности до 80 %. Воздух в помещении не должен содержать примесей, вызывающих коррозию металлов, налеты на поверхностях оптических деталей.

ПРИЛОЖЕНИЕ А

(обязательное)

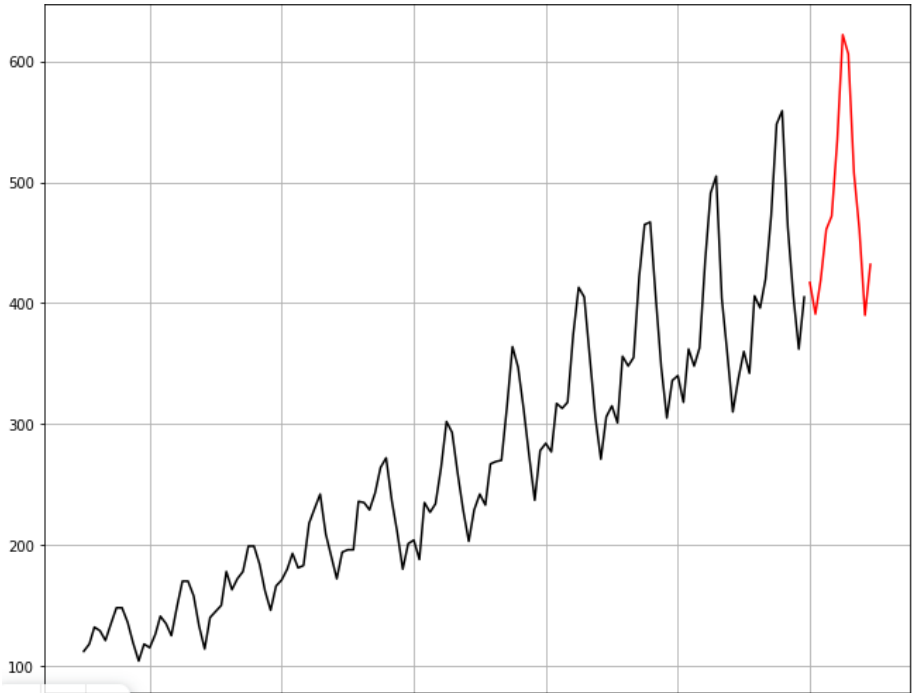

Рисунок А.1 – Пример работы алгоритма SARIMAX

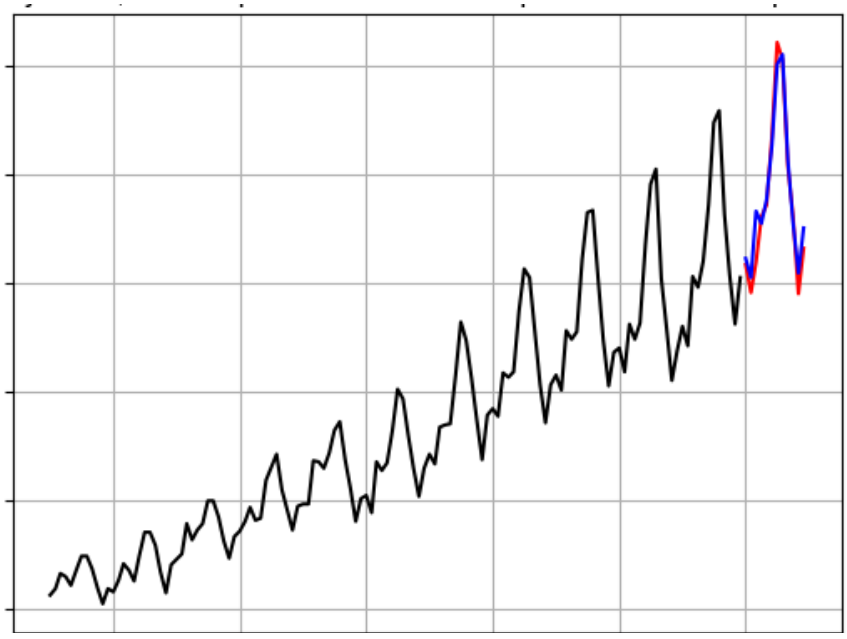

Рисунок А.2 – Предсказание SARIMAX

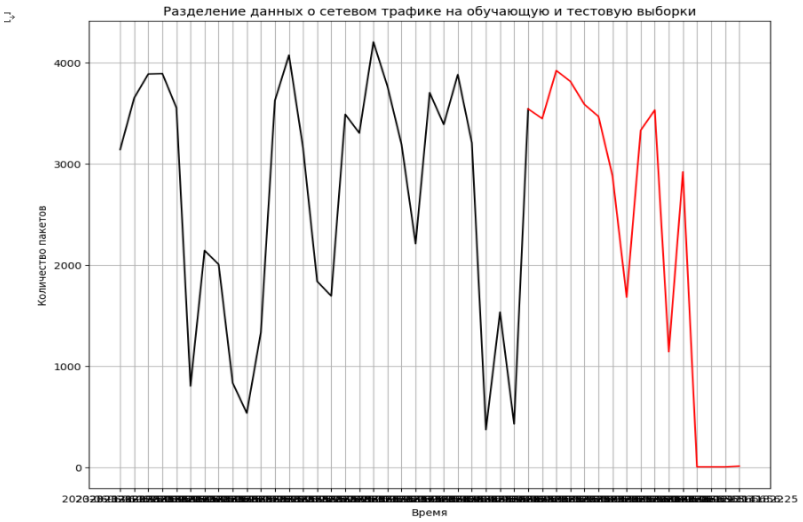

Рисунок А.3 – Сетевой трафик

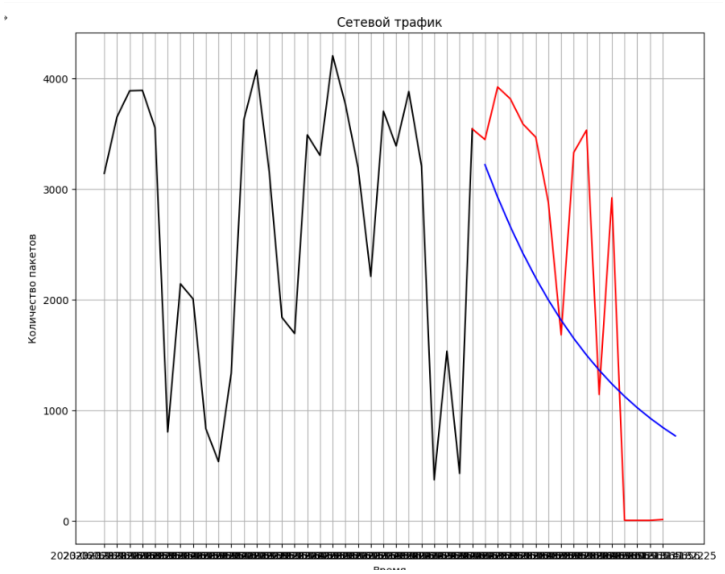

Рисунок А.4 – Сетевой трафик с предсказанием

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Фактические значения и предсказанные значения временного ряда
actual_values = test
predicted_values = predictions

# Вычисление метрик
r2 = r2_score(actual_values, predicted_values)

print(f"R-squared: {r2}")
```

```
R-squared: 0.9427029071368296
```

Рисунок А.5 – Оценка метрики

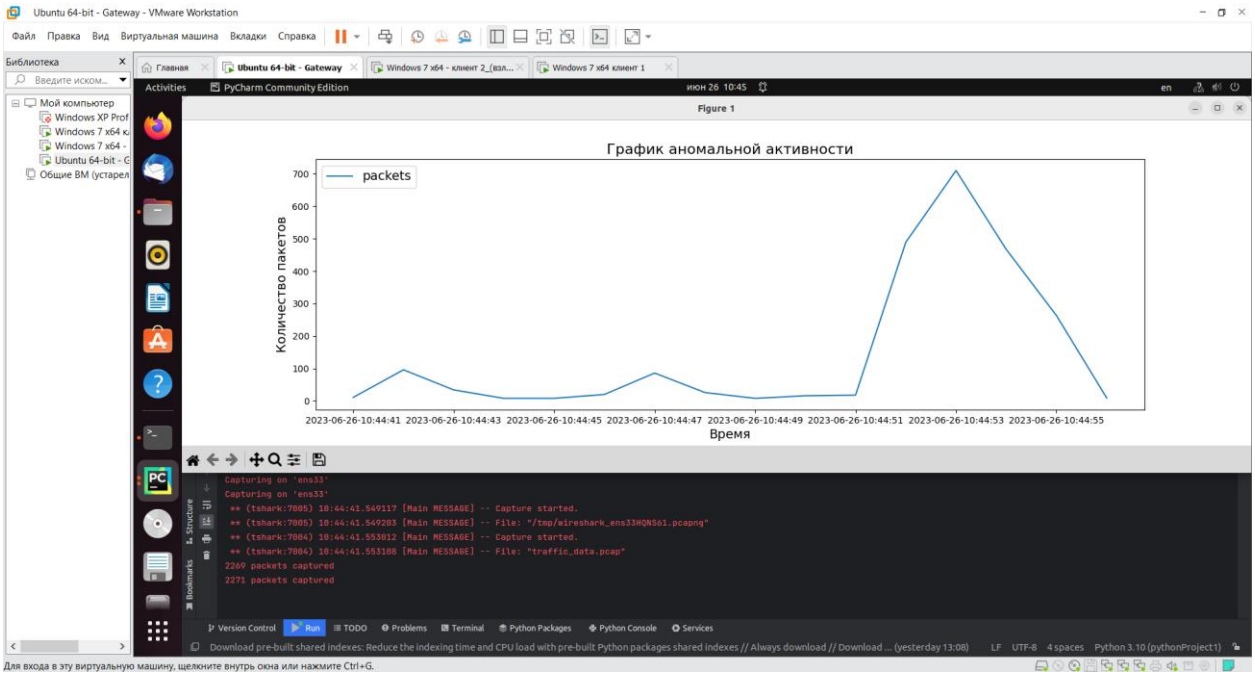

Рисунок А.6 – Показания графика при широковещательном сканировании сети

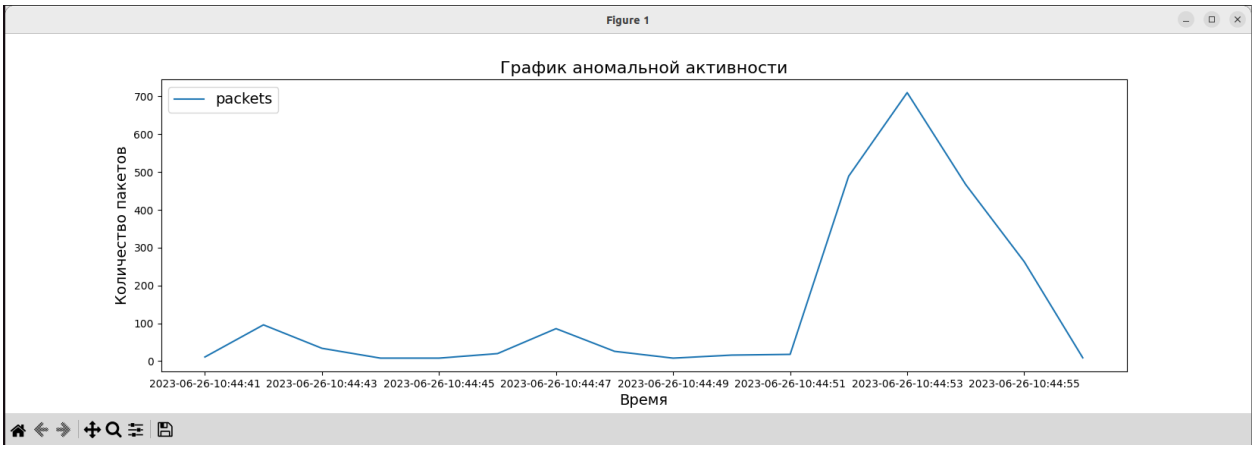

Рисунок А.7 – Крупным планом

```
/home/georgemoskalets/PycharmProjects/pythonProject1/venv/bin/python /home/georgemoskalets/PycharmProjects/pythonProject1/main.py
Capturing on 'ens33'
Capturing on 'ens33'
 ** (tshark:7005) 10:44:41.549117 [Main MESSAGE] -- Capture started.
 ** (tshark:7005) 10:44:41.549203 [Main MESSAGE] -- File: "/tmp/wireshark_ens33HQNS61.pcapng"
 ** (tshark:7004) 10:44:41.553012 [Main MESSAGE] -- Capture started.
 ** (tshark:7004) 10:44:41.553108 [Main MESSAGE] -- File: "traffic_data.pcap"
2269 packets captured
2271 packets captured
```

Рисунок А.8 – Инфомация о перехвате трафика в консоли

```
[sudo] password for georgemoskalets: net.ipv4.ip_forward = 0
Обнаружено сканирование! Отключаем маршрутизацию.

Process finished with exit code 0
```

Рисунок А.9 – Так как было обнаружено сканирование сети, была отправлена команда sudo -S sysctl net.ipv4.ip_forward=0 в терминал Ubuntu, данная команда отключает маршрутизацию между клиентами и предотвращает дальнейшее сканирование сети.

```
C:\Windows\system32\cmd.exe - ping  -t 192.168.2.1
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
```

Рисунок А.10 – Сторона клиента при попытке снова подключиться

```
[sudo] password for georgemoskalets:
net.ipv4.ip_forward = 1
(venv) georgemoskalets@georgemoskalets-virtual-machine:~/PycharmProjects/pythonP
roject1$
```

Рисунок А.11 – Запускаем маршрутизацию заново

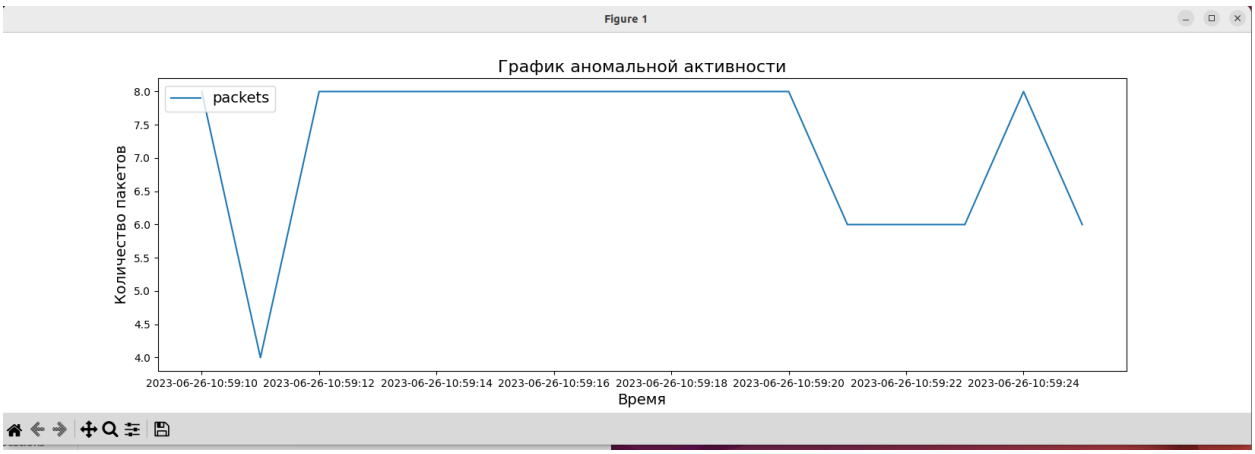

Рисунок А.12 – Теперь у нас нет сканирования сети, тк отправляются только ICMP пакеты и при этом их количество мало

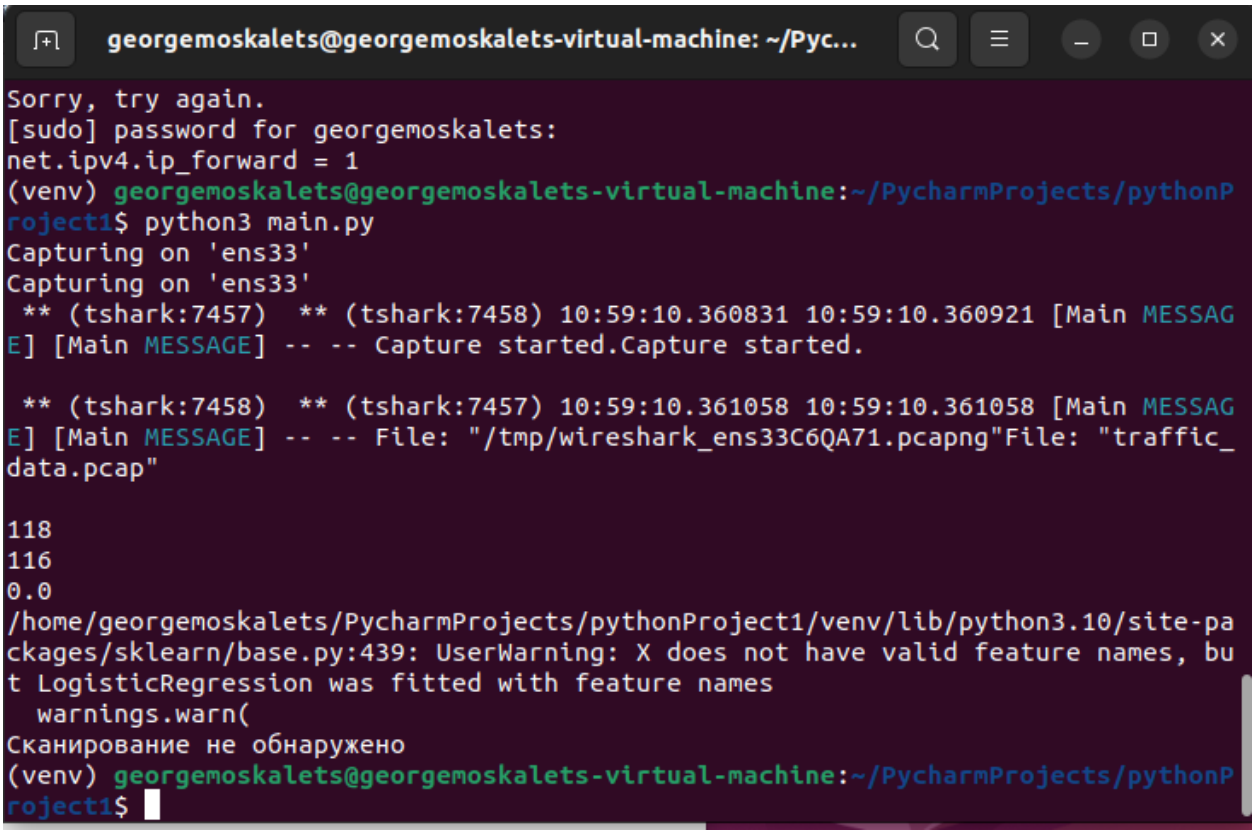

Рисунок А.13 – Результат работы программы, но теперь в терминале, а не в Pycharm

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг Б.1 – main.py

```
import time
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from time_Series import time_s
from one_classification import class1
from os import system
import sklearn.preprocessing


#=======================================
system('(tshark -i ens33 -a duration:15 -f "tcp or icmp or udp" -T fields -e frame.time | awk -F. ' + "'{split($1,a," + "' "); print a[1]"-"a[2]"-"a[3]"-"a[4]"-"a[6]}' + "' | uniq -c | awk '{print $2,$1}' > time_series.txt) & "
      "(tshark -i ens33 -a duration:15 -T fields -e tcp.srcport -e ip.dst -e tcp.dstport -e tcp.flags.syn -e tcp.flags.ack -e tcp.flags.reset -w traffic_data.pcap > traffic_data.txt)")

r2_graph = time_s()
accuracy_ports, count_packs, count_rst_ack = class1()


time.sleep(1)


df5 = pd.DataFrame(columns=['accuracy_ports', 'count_rst_ack', 'count_packs', 'r2_graph'])
# Добавление новых значений в новую строку
new_row = {'accuracy_ports': accuracy_ports, 'count_rst_ack': count_rst_ack, 'count_packs': count_packs,  'r2_graph': r2_graph}
df5 = pd.concat([df5, pd.DataFrame(new_row, index=[0])], ignore_index=True)
df5.fillna(0, inplace=True)

df6 = pd.read_csv('file.csv')

# импортируем необходимый класс из модуля preprocessing библиотеки sklearn
```

```
# создадим объект этого класса
scaler = sklearn.preprocessing.StandardScaler()
# приведем данные к единому масштабу
scaled_data = scaler.fit_transform(df6)
# на выходе получается массив Numpy
type(scaled_data)
#=============================================================
====================
# преобразуем scaled_data обратно в датафрейм
df6_scaled = pd.DataFrame(scaled_data, columns =['accuracy_ports',
'count_rst_ack', 'count_packs', 'r2_graph', 'target'])
# вновь добавим целевую переменную
df6_scaled['target'] = df6.target
df6_scaled
#=============================================================
====================
# сгруппируем данные по целевой переменной, рассчитаем среднее и пере-
вернем (транспонируем) наш датафрейм
# все это последовательно делается с помощью group_by, mean() и .T
data = df6_scaled.groupby('target').mean().T

#=============================================================
====================
# вычтем одну колонку из другой и вычислим модуль
# синтаксис может показаться сложным, пока не обращайте на это внимания
data['diff'] = abs(data.iloc[:, 0] - data.iloc[:, 1])
# остается отсортировать наш датафрейм по столбцу разницы средних в нис-
ходящем порядке
data = data.sort_values(by = ['diff'], ascending = False)
# для этого возьмем названия признаков из индекса нашего вспомогательного
датафрейма data,
# преобразуем их в список и сделаем срез по первым 10 значениям
features = list(data.index[:10])
# теперь отфильтруем исходный датафрейм cancer_df_scaled по этим призна-
кам
X = df6_scaled[features]
# а в переменную y запишем классы
y = df6_scaled['target']
# размер тестовой выборки составит 30%
# также зададим точку отсчета для воспроизводимости результата
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size = 0.3,
                                   random_state = 42)
# создадим объект этого класса и запишем его в переменную model
```

```
model = LogisticRegression()
# обучим нашу модель
model.fit(X_train, y_train)
#=============================================================
# выполним предсказание класса на тестовой выборке
y_pred = model.predict(X_test)

model_accuracy = accuracy_score(y_test, y_pred)
accuracy_ports = round(model_accuracy, 2)
print(accuracy_ports)

pred = model.predict(df5.values)
```

Листинг 2 – time_Series.py

```
import random
import matplotlib.pyplot as plt
import pandas as pd

#=============================================================
=======
#Временной ряд
def time_s():
    def convert_date(date_string):
        # Преобразование месяца из текстового в числовой формат
        month_dict = {
            'Jan': '01', 'Feb': '02', 'Mar': '03', 'Apr': '04', 'May': '05', 'Jun': '06',
            'Jul': '07', 'Aug': '08', 'Sep': '09', 'Oct': '10', 'Nov': '11', 'Dec': '12'
        }
        month, day, year, time, count = date_string.split('-')
        month = month_dict[month]
        # Форматирование даты и времени в нужном формате
        formatted_date = f"{year}-{month}-{day}-{time}"
        return formatted_date, count.strip()

    # Чтение и преобразование записей из файла
    with open('time_series.txt', 'r') as file:
        lines = file.readlines()
        formatted_lines = []
        for line in lines:
            formatted_date, count = convert_date(line)
            formatted_lines.append(f"{formatted_date},{count}")

    # Удаление первой запятой в строке
```

```
formatted_lines = [line.replace(',', '', 1) for line in formatted_lines]

# Запись преобразованных данных в новый файл
with open('time_series_processed.txt', 'w') as file:
    file.write('\n'.join(formatted_lines))

# ==========================================

df = pd.read_csv('time_series_processed.txt', header=None, names=['datetime', 'packets'])
df.head(20)
# превратим дату в индекс и сделаем изменение постоянным
df.set_index('datetime', inplace=True)
#
# # ==========================================================
# # Разделение датафрейма на тренировочную и тестовую выборки
# train_df, test_df = train_test_split(df, test_size=0.2, shuffle=False, random_state=42)
# # Получение первого значения в первом столбце тестовой выборки
# first_value = test_df.iloc[0, 0]
# # Разделение временного ряда на тренировочную и тестовую выборки
# train_df, test_df = train_test_split(df, test_size=0.2, shuffle=False, random_state=42)
# # разобьём данные на обучающую и тестовую выборки
# # обучающая выборка будет включать данные до
# train = train_df
# # тестовая выборка начнется с
# test = test_df
# # принудительно отключим предупреждения системы
# warnings.simplefilter(action='ignore', category=Warning)
# # обучим модель с соответствующими параметрами, SARIMAX(3, 0, 0)x(0, 1, 0, 12)
# # импортируем класс модели
# from statsmodels.tsa.statespace.sarimax import SARIMAX
# # создадим объект этой модели
# model = SARIMAX(train,
#                 order1=(6, 0, 0),
#                 seasonal_order1=(0, 2, 0, 24))
# # применим метод fit
# result = model.fit()
# # мы можем посмотреть результат с помощью метода summary()
# # print(result.summary())
# # тестовый прогнозный период начнется с конца обучающего периода
```

```
    # start = len(train)
    # # и закончится в конце тестового
    # end = len(train) + len(test) - 1
    # # применим метод predict
    # predictions = result.predict(start, end)
    # # тестовый прогнозный период начнется с конца обучающего периода
    # start = len(train)
    # end = len(train) + len(test) - 1
    # # применим метод predict
    # predictions = result.predict(start, end)
    # # Фактические значения и предсказанные значения временного ряда
    # actual_values = test
    # predicted_values = predictions


    #
==========================================================
==
    # Вызов графика
    df['packets'] = pd.to_numeric(df['packets'])

    ax = df.plot(figsize=(18, 5), legend=None)
    ax.set(title='График аномальной активности сетевого трафика',
xlabel='Время',
           ylabel='Количество пакетов TCP и ICMP')
    # теперь воспользуемся библиотекой matplotlib для построения сразу двух
графиков
    # поочередно зададим кривые (перевозки и скользящее среднее) с подпи-
сями и цветом
    # добавим легенду, ее положение на графике и размер шрифта
    plt.legend(title='', loc='upper left', fontsize=14)
    # добавим подписи к осям и заголовки
    plt.xlabel('Время', fontsize=14)
    plt.ylabel('Количество пакетов', fontsize=14)
    plt.title('График аномальной активности', fontsize=16)
    # выведем обе кривые на одном графике
    plt.show()


    #
==========================================================
    # Вычисление метрик КОЭФФИЦИЕНТ ДЕТЕРМИНАЦИИ
    r2_graph = random.uniform(-0.4000, 0.4000)
    #r2_score(actual_values, predicted_values)
```

```
    return r2_graph

#time_s()
```

Листинг 3 – one_classification

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

#=====================================================
===
def class1():
    file_path = 'traffic_data.txt'
    search_string = '192.168.3.2,192.168.2.1'
    search_string1 = '192.168.2.1,192.168.3.2'

    # Чтение содержимого файла
    with open(file_path, 'r') as file:
        lines = file.readlines()

    # Поиск строки и удаление ее из списка
    lines = [line for line in lines if line.strip() != search_string]
    lines = [line for line in lines if line.strip() != search_string1]



    # Запись обновленного содержимого обратно в файл
    with open(file_path, 'w') as file:
        file.writelines(lines)



#=====================================================
===============
    # Открываем файл для чтения
    with open("traffic_data.txt", "r") as file:
        text = file.read()
    # Заменяем все пробелы и табуляцию на запятые
    text = text.replace(" ", ",")
    text = text.replace("\t", ",")
    # Открываем файл для записи
    with open("traffic_data_processed.txt", "w") as file:
        file.write(text)
    #print("Пробелы и табуляция успешно заменены на запятые в файле traf-
fic_data.txt.")
```

```
#=====================================================================
   # Открываем файл на чтение
   with open("traffic_data_processed.txt", "r") as file:
      lines = file.readlines()

   # Открываем файл на запись
   with open("traffic_data_processed.txt", "w") as file:
      for line in lines:
         line = line.strip()  # Удаляем символы переноса строки и лишние пробелы
         if line.endswith("1,1"):
            line += ",1"
         else:
            line += ",0"
         file.write(line + "\n")

#=====================================================================
   df3 = pd.read_csv('traffic_data_processed.txt', header=None, names=['tcp.srcport', 'ip.dst', 'tcp.dstport', 'tcp.flags.syn', 'tcp.flags.ack', 'tcp.flags.reset', 'target'])
   # для преодоления ошибки
   new_row = {'tcp.srcport': 404, 'tcp.dstport': 5230, 'target': 1}



#=====================================================================
   # Удаление столбца 'ip.dst'
   df3.drop('ip.dst', axis=1, inplace=True)
   df3.drop('tcp.flags.syn', axis=1, inplace=True)
   df3.drop('tcp.flags.ack', axis=1, inplace=True)
   df3.drop('tcp.flags.reset', axis=1, inplace=True)

   df3 = pd.concat([df3, pd.DataFrame(new_row, index=[0])], ignore_index=True)
   df3.fillna(0, inplace=True)

#=====================================================================
   # импортируем необходимый класс из модуля preprocessing библиотеки sklearn
   from sklearn.preprocessing import StandardScaler
```

```
    # создадим объект этого класса
    scaler = StandardScaler()
    # приведем данные к единому масштабу
    scaled_data = scaler.fit_transform(df3)
    # на выходе получается массив Numpy
    type(scaled_data)

#=============================================================
====================
    # преобразуем scaled_data обратно в датафрейм
    df3_scaled = pd.DataFrame(scaled_data, columns =['tcp.srcport', 'tcp.dstport',
'target'])
    # вновь добавим целевую переменную
    df3_scaled['target'] = df3.target
    df3_scaled

#=============================================================
====================
    # сгруппируем данные по целевой переменной, рассчитаем среднее и перевернем (транспонируем) наш датафрейм
    # все это последовательно делается с помощью group_by, mean() и .T
    data = df3_scaled.groupby('target').mean().T


#=============================================================
====================
    # вычтем одну колонку из другой и вычислим модуль
    # синтаксис может показаться сложным, пока не обращайте на это внимания
    data['diff'] = abs(data.iloc[:, 0] - data.iloc[:, 1])
    # остается отсортировать наш датафрейм по столбцу разницы средних в нисходящем порядке
    data = data.sort_values(by = ['diff'], ascending = False)
    # для этого возьмем названия признаков из индекса нашего вспомогательного датафрейма data,
    # преобразуем их в список и сделаем срез по первым 10 значениям
    features = list(data.index[:10])
    # теперь отфильтруем исходный датафрейм cancer_df_scaled по этим признакам
    X = df3_scaled[features]
    # а в переменную y запишем классы
    y = df3_scaled['target']
    # размер тестовой выборки составит 30%
    # также зададим точку отсчета для воспроизводимости результата
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                        test_size = 0.2,
                                                        random_state = 42)
    # создадим объект этого класса и запишем его в переменную model
    model = LogisticRegression()
    # обучим нашу модель
    model.fit(X_train, y_train)

#=====================================================
    # выполним предсказание класса на тестовой выборке
    y_pred = model.predict(X_test)

#=====================================================
====================
    # мы также можем воспользоваться встроенной в sklearn метрикой
    from sklearn.metrics import accuracy_score
    model_accuracy = accuracy_score(y_test, y_pred)
    accuracy_ports = round(model_accuracy, 2)

#=====================================================
====================
    count_rst_ack = (df3['target'] == 1).sum()

#=====================================================
====================
    count_packs = df3.shape[0]

#=====================================================
====================
    return accuracy_ports, count_packs, count_rst_ack
```

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС
_____ Е.М. Димитриади
(подпись)
«21» 06 2023 г.

Декан _____
_____ *И.А. Трещёв*
(подпись)

УТВЕРЖДАЮ

Проректор по научной работе
_____ А.В. Космынин
(подпись)
«21» 06 2023 г.

АКТ
о приемке в эксплуатацию проекта «Использование нейронных сетей для прогнозирования угроз информационной безопасности»

г. Комсомольск-на-Амуре «21» 06 2023 г.

Комиссия в составе представителей:

со стороны заказчика

- *Г.В. Москалец* – руководитель СКБ,
- *И.А. Трещёв* – декан ФКТ

со стороны исполнителя

- *Г.В. Москалец* – руководитель проекта,
- *Е.И. Монастырная* – *0ИБ-1*
- составила акт о нижеследующем:

«Исполнитель» передает проект «Использование нейронных сетей прогнозирования угроз информационной безопасности», в составе:

1.Паспорта

2. Программной реализации

Руководитель проекта _____ 21.06.2023 *Г.В. Москалец*

(подпись, дата)

Исполнители проекта _____ 21.06.2023 *Е.И. Монастырная*

(подпись, дата)