DOI 10.36622/1682-7813.2024.27.3.003
УДК 349:004.056

НЕЙРОСЕТЕВОЙ СЕРВИС РЕГЛАМЕНТАЦИИ МЕР ПРОТИВОДЕЙСТВИЯ КИБЕРАТАКАМ (ЧАСТЬ II)

Г.А. Остапенко, А.П. Васильченко, А.А. Остапенко, А.А. Ноздрюхин, А.Г. Остапенко, И.Л. Батаронов, Д.С. Печкин

Рассматривается нейросетевая реализация сервиса автоматизированного противоборства в части регламентации мер противодействия компьютерным атакам. В этой связи предлагается программная реализация сервиса нейросетевой регламентации мер противодействия кибератакам, включая скрипты определения ip-адресов, взаимодействие с языковой моделью, модуль машинного обучения, скрипты подготовки базы знаний и осуществления машинного обучения, бэкенд-сервис, фронтенд-сервис. Демонстрируются примеры практического использования сервиса в режимах регистрации вторжения, интеллектуальных подсказок, в комбинированном режиме.

Ключевые слова: сервис, нейросетевая реализация сервиса, кибератака, скрипт, веб-приложение, бэкенд-сервис, фронтенд-сервис.

Введение

Сформированные в предыдущей части методическое и алгоритмическое обеспечения позволяют провести программную реализацию модуля регламентации мер противодействия. С учетом имеющихся алгоритмов, описанных ранее, и программных возможностей, позволяющих их реализовать, можно выделить 4 основные сервиса, которые должны быть реализованы для обеспечения функционирования модуля регламентации мер противодействия, к их числу относятся:

− скрипт определения ip-адресов локальной сети;

− API, необходимое для взаимодействия с локальной языковой моделью;

− веб-приложение администратора безопасности, которое в том числе включает в себя поток, принимающий пакеты с полезной информацией о сценарии атаки;

− модуль машинного обучения.

При этом для успешного развертывания и запуска приложения необходимо будет произвести следующие действия:

1. Запустить скрипт определения ip-адресов, которой запишет ip-адреса работающих автоматизированных информационных систем в отдельный файл ip.txt иподаст на вход веб-приложению.

2. Открыть LM Studio, выбрать в ней заранее прошедшую дополнительное машинное обучение модель saiga_mistral_7b_lora и запустить ее на локальном сервере [1].

3. Последним шагом является запуск веб-приложения, в котором при входе в левой части экрана будут отображаться ip-адреса автоматизированных информационных систем из локальной сети, при переходе на каждый из которых будет открыт доступ к чату, содержащий информацию о проведенных атаках и мерах противодействия для конкретного хоста. Здесь администратор имеет возможность ведения диалога с нейросетью для корректировки выданных мер. Приведем подробное описание программной реализации каждого сервиса.

Программная реализация скрипта определения ip-адресов в локальной сети

В процессе разработки были рассмотрены различные варианты реализации данного скрипта и предполагалось сделать его автоматически запускаемым при старте веб-приложения до его инициализации. Однако такой вариант имеет ряд проблем, главная из которых упирается в наличие на хосте нескольких

сетевых интерфейсов, и неопределенности того, по какому интерфейсу компьютер администратора безопасности соединен с локальной сетью. В связи с этим возникла необходимость реализовать скрипт таким образом, чтобы администратор сам мог выбирать сетевой интерфейс, по которому его компьютер соединен с локальной сетью.

Стоит заметить, что в различных системах сетевые интерфейсы называют по-разному. Например, в системе Linux в качестве нейминга сетевых интерфейсов используют такие слова как «enp0s3», чаще всего обозначающий сетевой кабель или «lo», именующий loopback-интерфейсы компьютера [2]. Однако в системе Windows нейминг интерфейсов осуществляется по-другому, и в названиях часто отказываютсяот использования стандартизованных слов, по которым можно понять к какому типу данный интерфейс относится. В качестве примера, можно привести такие названия как «0F0EFA7E-4CB5-46DB-B0E5-DC4D152EB81A» или «C7B97CC7-3E3C-4173-A6EB-382D40003A4B». В связи с чем для понимания роли данного интерфейса администратору безопасности нужна дополнительная информация, в виде ip-адреса и маски сети. В таком случае при виде сетевого интерфейса с ip-адресом «127.0.0.1» администратор будет четко понимать, что это loopback-интерфейс и его выбирать не стоит [3].

С учетом всего вышесказанного сформируем перечень требований, которым должен соответствовать данный скрипт:

1. Обеспечение возможности выбора администратором безопасности выбора сетевого интерфейса. Вывод подробной информацией о сетевом интерфейсе, включающий в себя не только его название, но и ip-адрес и маску подсети.

2. Формирование файла со списком ip-адресов локальной сети по окончанию выполнения скрипта.

3. Обеспечение удобства запуска посредством реализации данного скрипта в виде .exe файла, для того чтобы администратору безопасности не пришлось использовать командную строку.

Для программной реализации данного скрипта был выбран язык программирования python, в виду его простоты установки дополнительных библиотек, которые можно скачать и добавить в проект, используя команду «pip install [названиебиблиотеки] == [версия]», причем последняя часть необязательна, и без ееиспользования будет скачана последняя версия необходимой библиотеки. В рамках данного скрипта используются следующие библиотеки:

1) netifaces, предоставляющая разработчику удобные средства для получения информации о сетевых интерфейсах и их конфигурации [4];

2) ipaddress, необходимая для создания, манипулирования и анализа IPv4 и IPv6 адресов и сетей на уровне приложения [5];

3) ping3, которая используется для отправки ICMP-запросов на указанные IP-адреса или доменные имена с целью получения информации о времени отклика и доступности узлов [6].

Начало программной реализации скрипта характеризуется подключением перечисленных библиотек при помощи утилиты «import». Далее происходит вывод имеющихся на компьютере сетевых интерфейсов, получаемых при помощи метода interfaces() библиотеке netifaces [4]. Стоит отметить, что вывод информации о сетевых интерфейсах содержит не только название интерфейса, но и его ip-адрес и маску подсети. После этого выводится системное сообщение пользователю о предложении ввода номера сетевого интерфейса, который он хочет использовать для подключения к локальной сети. Это показано на рис. 1.

```python
# Этот файл необходим для получения ip-адресов всех хостов, находящихся в локальной сети АИС

import netifaces
import ipaddress
import ping3

# Вывод интерфейсов хоста
print("IP address of existing interfaces")
count = 0
for interface in netifaces.interfaces():
    count += 1
    print(
        f'\t{count}) Interface: {interface} {netifaces.ifaddresses(interface)[netifaces.AF_INET][0]["addr"]} '
        f'{netifaces.ifaddresses(interface)[netifaces.AF_INET][0]["netmask"]}')

# Выбираем интерфейс
print(f"Select the interface number (from 1 to {count}): ")
number = int(input())

# Получить IP-адрес хоста, на котором запущен этот скрипт
host_ip = netifaces.ifaddresses(netifaces.interfaces()[number - 1])[netifaces.AF_INET][0]['addr']
host_mask = netifaces.ifaddresses(netifaces.interfaces()[number - 1])[netifaces.AF_INET][0]["netmask"]
print(f"Host ip: {host_ip}")
```

Рис. 1. Выбор сетевого интерфейса

После успешного выбора сетевого интерфейса происходит получение ip-адреса хоста администратора безопасности и создание объекта сети, это необходимо для того, чтобы иметь возможность осуществить отправку ICMP-запросов на другие адреса из локальной сети. Эта информация находит отражение на рис. 2.

```python
# Получить IP-адрес хоста, на котором запущен этот скрипт
host_ip = netifaces.ifaddresses(netifaces.interfaces()[number - 1])[netifaces.AF_INET][0]['addr']
host_mask = netifaces.ifaddresses(netifaces.interfaces()[number - 1])[netifaces.AF_INET][0]["netmask"]
print(f"Host ip: {host_ip}")

# Создать объект сети
ip_interface = ipaddress.ip_interface(f'{host_ip}/{host_mask}')
network = ipaddress.ip_network(f'{str(ip_interface.network.network_address)}/{host_mask}')
print(f"Network: {network}")
```

Рис. 2. Получение сетевых параметров хоста, на котором запущен скрипт

На следующем этапе происходит создание списка ip-адресов локальной сети, который наполняется в цикле. Причем имеется проверка каждого ip-адреса на предмет соответствия ip-адресу хоста администратора безопасности, который был определен ранее. Это необходимо для того, чтобы компьютер администратора не попал в конечный список ip-адресов, по которым будет вестись чат. Также с этим ip-адресом пытается установить связь посредством отправки ICMP-запросов. В случае успешного ответа ip-адрес добавляется в специально заведенный список, а при неудачной попытке цикл переходит на следующую итерацию. Эта часть скрипта отображена на рис. 3.

```
30  # Создать пустой список для хранения IP-адресов
31  ip_list = []
32
33
34  # Перебрать все IP-адреса в сети
35  for ip in network.hosts():
36      # Преобразовать IP-адрес в строку
37      ip_str = str(ip)
38
39      # Проверить, не равен ли текущий IP-адрес IP-адресу хоста
40      if ip_str != host_ip:
41          # Отправить пакет ICMP echo request на текущий IP-адрес
42          if ping3.verbose_ping(ip_str, count=1):
43              # Если ответ положительный, добавить IP-адрес в список
44              ip_list.append(ip_str)
45
```

Рис. 3. Формирование списка активных устройств в локальной сети

Финальным этапом реализации данного скрипта является запись полученного списка в файл ip.txt. Стоит отметить, что для этого в функционале записи выбрана опция «w», которая позволяет создавать файл в случае отсутствия и обновлять при егоналичии. Это показано на рис. 4.

```
48  # Записать ip-адреса в файл
49  with open('ip.txt', 'w') as file:
50      for ip in ip_list:
51          file.write(f"{ip}\n")
52          print(f"\t{ip}")
53
```

Рис. 4. Запись активных устройств из локальной сети в файл

Как было сказано ранее, для более удобного взаимодействия администратора безопасности с данным скриптом нужно, чтобы он имел расширение .exe. Для этого в python существует библиотека pyinstaller, которую можно скачать при помощи команды «pip install pyinstaller». И для создания файла .exe достаточно выполнить команду «pyinstaller localNetworkFuncs.py». В результате чего в проекте появляется новые папки и нужный .exe файл лежит по адресу «/dist/localNetworkFuncs/localNetworkFuncs.exe». На компьютере администратора безопасности его необходимо переносить вместе с папкой «dist», так как в ней лежат все библиотеке и зависимости,используемые скриптом [7].

Программная реализация API для взаимодействия с языковой моделью Достаточно важную в рамках правильного функционирования веб-приложения играет API, обеспечивающее взаимодействие с локальной языковой моделью. Главной задачей данного API является отправка запросов в языковую модель и прием ответов, сгенерированных нейросетью. Причем должна быть проработана возможность отправки запроса пользователя с уже имеющейся историей чата, включающей не только пользовательские сообщения, но и ответы нейросети. Для фокусирования внимания нейросети на сфере информационной безопасности, а именно регламентации мер противодействия в виде выдачи интеллектуальных подсказок в формате мер реагирования и ликвидации последствий, необходим системный запрос, конкретизирующий поведение модели.

Для удовлетворения потребностей, описанных выше, был создан файл lmAPI.py. В своей реализации он использует лишь одну библиотеку – openai [8], которая позволяет обеспечить удобное взаимодействие не только с моделями одноименной компании,

но локальными нейросетевыми реализациями, развернутыми в LM Studio [1]. В начале скрипта инициализируется константа NEIRO_SERVER_ADDRESS, которая описывает адрес сервера, на котором развернута модель. Далее происходит создания клиента OpenAI, которому в качестве базового адреса присваивается константа NEIRO_SERVER_ADDRESS, а в качествеключа API-фраза «lm-studio». На самом деле вместо этой фразы можно было написать что угодно, так как этот параметр необходим для взаимодействия с API от OpenAI, чтоявляется небесплатной услугой.

Данный скрипт имеет лишь один метод «get_response()», который получает на вход историю чата администратора безопасности с локальной языковой моделью и возвращает ответ нейросети в текстовой форме. Для выполнения требования по соответствию языковой модели специфике регламентации мер противодействия в необходимом формате инициализируется системный запрос, содержащий достаточно длинную инструкцию, описывающую поведение нейросети. Этот запрос добавляется на первое место в список «messages», после чего к данному списку в цикле присоединяется вся история чата. Далее формируется запрос в модель, где имя модели указывается как «local». В качестве истории чата идет заполненный список «messages» и выбирается параметр температура, который отвечает за оригинальность ответов и варьируется в пределах от 0 до 1. Так как мы пытаемся достичь стандартизации ответов, с учетом тех данных, на основании которых была обучена модель, в значение температуры необходимо поставить маленькое значение, такое как или 0.1. После отправки запроса ожидается ответ, причем он приходит в json формате, который необходимо подвергнуть небольшому парсингу, обращаясь к полю content переменной message, лежащей в объекте выбора ответов «choices». Так как модель настроена под выдачу одного ответа, то необходимо обращаться к первому и единственному элементу объекта choices [8]. Программная реализация описанных действия изображена на рис. 5.

```python
from openai import OpenAI


NEIRO_SERVER_ADDRESS = "http://localhost:1234/v1"

client = OpenAI(base_url=NEIRO_SERVER_ADDRESS, api_key="lm-studio")


 Aleksandr Nozdryuhin *
def get_response(chat_history):
    messages = [{"role": "system",
                 "content": "Ты специалист по информационной безопасности, который выдает интеллектуальные "
                            "подсказки для регламентирования мер противодействия кибератакам, состоящие из мер "
                            "реагирования и мер ликвидации последствий. Меры генерируются на основании входных "
                            "данных: атаки и уязвимости. Если тебе на вход поступают неизвестные для тебя "
                            "данные об атаке или уязвимости, то ты генерируешь новые меры по имеющейся у "
                            "тебя информации. Также тебе могут задавать вопросы для корректировки этих мер, выдавай "
                            "скорректированные ответы четко, пользуясь при этом информацией о мерах противодействия, "
                            "на которых ты обучен"}]
    for val in chat_history:
        messages.append(val)
    response = client.chat.completions.create(
        model="local",
        messages=messages,
        temperature=0,
    )
    return response.choices[0].message.content
```

Рис. 5. Программная реализация API для взаимодействия с языковой моделью

Необходимо сказать, что правильное функционирование данного API возможно лишь в том случае, если перед его использованием была запущена языковая модель на локальном или удаленном сервере LM Studio [1]. В случае запуска на сервере необходимо поменять значение константы NEIRO_SERVER_ADDRESS.

Программная реализация модуля машинного обучения

Как было сказано ранее, выдача

валидных ответов в соответствие с необходимыми форматами, включающими в себя меры как реагирования, так и ликвидации последствий, невозможна без предварительного проведения дополнительного машинного обучения языковой модели, которое в свою очередь невозможно без наличия заранее заготовленной базы знаний, приведенной под определенной формат. Решение этих двух задач обеспечивает обобщенный сервис машинного обучения, разделяющийся на две части:

− скрипт формирования валидной базы знаний для машинного обучения;

− скрипт, отвечающий за дополнительное машинное обучение модели.

Приведем подробное описание каждого скрипта из этого списка с точки зрения его программной реализации.

Скрипт подготовки базы знаний

Как было выяснено ранее, перед дополнительным машинным обучением необходимо создать валидную базу знаний. При этом она должна соответствовать некоторым требованиям:

− формат файла – json;

− каждая порция данных, подаваемая на вход машинного обучения, должнасодержать системное сообщение, описывающее поведение нейросети (роль system), запрос пользователя (роль user) ответ нейросети (роль bot);

− каждая новая порция данных должна записываться с новой строки и заключаться в фигурные скобки;

− информация, но основание которой формируется база знаний хранится в файле ML.xlsx, формат которого описан в алгоритмическом обеспечении.

Наиболее простая реализация данного скрипта с учетом перечисленных требований осуществляется при помощи языка программирования python ввиду того, что в нем содержится высокоуровневая библиотека pandas, позволяющая удобно манипулировать с данными различных форматов.

Программное обеспечение скрипта подготовки базы знаний машинногообучения находит свое отражение в файле mlAPI.py. Помимо pandas в скрипте используется библиотека json, позволяющая заворачивать данные в json-объекты и записывать их в файл одноименного формата. По сути, программную реализацию скрипта подготовки базы знаний можно разделить на две логические части:

− инициализация данных;

− парсинг данных и формирование json-файла.

В первой части происходит создание констант, отражающих названия колонок из первичной базы данных ML.xlsx, список вопросов пользователя и системный запрос, который специализирует нейросеть под регламентацию мер противодействиясетевым вторжениям. Эти константы необходимы для удобства формирования json- подобных сущностей [9]. После инициализации переменных происходит чтение всей информации из базы знаний ML.xlsx по названию колонок и запись этих данных в локальные переменные. Это показано на рис. 6.

```
import pandas as pd
import json

COLUMNS_NAMES = ['Наименование атаки', 'Название атаки', 'Семейство уязвимостей с общей характеристикой',
                 'Меры реагирования на регистрируемые инциденты, порожденные парой атака-уязвимость',
                 'Меры ликвидации последствий инцидентов, порожденных парой атака-уязвимость']

QUESTIONS = ['Напиши меры противодействия для атаки']

SYSTEM_PROMPT = "Ты специалист по информационной безопасности, который выдает интеллектуальные подсказки для " \
                "регламентирования мер противодействия кибератакам, состоящие из мер реагирования и мер ликвидации " \
                "последствий. Меры генерируются на основании входных данных: атаки и уязвимости. Если " \
                "тебе на вход поступают неизвестные для тебя данные об атаке или уязвимости, " \
                "то ты генерируешь новые меры по имеющейся у тебя информации. Также тебе могут задавать вопросы " \
                "для корректировки этих мер, выдавай скорректированные ответы четко, пользуясь при этом информацией  " \
                "о мерах противодействия,на которых ты обучен"

excel = pd.read_excel('ML.xlsx')
data_excel = pd.DataFrame(excel, columns=COLUMNS_NAMES)
```

Рис. 6. Инициализация констант и чтение данных из ML.xlsx

Следующая часть скрипта производит формирование запросов на основе конкатенации констант с полезной информацией из ML.xlsx наряду с их записью в json файл. В начале открывается файл ML.json, причем в параметрах открытия задается флаг ‘w’, позволяющий создавать файл при отсутствии и обновлять его при наличии. Далее происходи обход в цикле списка со значением порций данных, взятыхиз файла ML.xlsx построчно. Внутри этого цикла создается еще один, уже со спискомзаранее инициализированных вопросов, по которому также происходит обход. На основании инициализированных в константах вопросов формируются два запроса пользователя: один с необходимостью регламентации мер по названию атаки, другой по ее идентификатору CAPEC. После этого конструируется строка ответа бота с мерами реагирования и ликвидации последствий. Далее создается json-словарь, содержащий наименования ролей, системный запрос, запрос пользователя и ответ нейросети, который переводится в строку и с помощью функции write записывается в файл ML.json [9]. Это показано на рис. 7.

```
# Открываем файл для записи
with open('ML.json', 'w', encoding='utf-8') as file:
    # Перебираем строки из DataFrame
    for de in data_excel.values:
        for q in QUESTIONS:
            user_msg_capec = f"{q} {de[0]}, учитывай при этой уязвимости: \'{de[2]}\'"
            user_msg_name = f"{q} {de[1]}, учитывай при этой уязвимости: \'{de[2]}\'"
            bot_msg = f"Для {de[1]}({de[0]}) уместны следующие меры реагирования: \n{de[3]}\nДля ликвидации негативных " \
                      f"последствий,связанных с реализацией данного сценария стоит применить следующие мероприятия:\n{de[4]}\n"

            # Создаем словарь для хранения одного JSON-элемента
            json_elem = {
                "request": [
                    {"role": "system", "text": SYSTEM_PROMPT},
                    {"role": "user", "text": user_msg_capec}
                ],
                "response": bot_msg.replace("\n", " \n ")
            }

            # Переводим словарь в JSON-строку
            json_elem_str = json.dumps(json_elem, ensure_ascii=False)
```

Рис. 7. Запись json-подобного объекта данных в файл

Стоит сделать замечание о том, что формируется два таких словаря, один из них содержит запрос пользователя с названием атаки, а другой с идентификатором. Оба этих словаря записываются в файл, причем производится добавление знака переноса строки для обеспечения соответствия выходного файла требуемому формату.

В результате запуска данного скрипта формируется база знаний ML.json, которую уже можно подавать на вход программного обеспечения, позволяющего произвести дополнительное машинное обучение.

Скрипт, осуществляющий машинное обучение

Получив в итоге выполнения предыдущего скрипта валидную базу знаний, можно на ее основании реализовать дополнительное машинное обучение методом LoRA. Стоит отметить, что машинное обучение является довольно ресурсоемким процессом, требующим серьезные вычислительные мощности, которые, как правило, больше мощностей, использующихся для обеспечения непосредственного функционирования крупных языковых моделей. Так как, в рамках проекта используется семимиллиардная языковая модель, то для ее машинного обучения необходимо выделить как минимум 16 гигабайт видеопамяти. В связи с этим требованием, машинное обучение необходимо проводить либо на специализированном сервере, имеющем серьезные вычислительные мощности, либос помощью таких платформ как Kaggle или Google Collab, которые предоставляют

возможность аренды GPU и CPU на определенное время [10, 11].

Проведем машинное обучение на платформе Kaggle ввиду того, что по сравнению с Google Collab данная платформа предоставляет больший объем GPU (2 видеокарты на 16 гигабайт). В рамках данного скрипта используются следующие библиотеки:

− os, которая нужна для выполнения команд операционной системы, в случае реализуемого скрипта для перезапуска ядер и создания/удаления каталогов [12];

− logging, ведущая журналы событий, что позволяется диагностировать ошибки при выполнении кода [13];

− pathlib, предоставляющая объектно-ориентированный интерфейс для манипулирования путями, что делает код более читаемым и удобным для написания [14];

− datasets, необходимая для загрузки и манипулирования данными машинного обучения [15];

− pandas, используемая для чтения и обработки данных из датасета [9];

− torch, которая предоставляет широкий спектр инструментов и функций для создания нейронных сетей, обучения моделей и их последующего использования[16];

− peft, используемая для более тонкой настройки языковых моделей, аименно изменения их архитектуры, добавления оптимизаторов и др. [17];

− transformers, работающая с моделями на основе трансформеров, разработанных для решения задач в области обработки естественного языка (NLP) [18].

Первым шагом скрипта дополнительного обучения является загрузка модели и установка конфигурационных настроек среды обучения. В качестве загружаемой модели выступает «IlyaGusev/saiga_mistral_7b_lora», а устройством обучения выставляется «cuda» в случае наличия GPU. В противном случае используется «cpu» и далее загружается адаптер LoRA, в котором свойство is_trainable необходимо выставить в True, для того чтобы была возможность проведения обучения. Это показано на рис. 8.

```
MODEL_NAME = "IlyaGusev/saiga_mistral_7b_lora"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

st_time = time.time()

config = PeftConfig.from_pretrained(MODEL_NAME)
model = AutoModelForCausalLM.from_pretrained(
    config.base_model_name_or_path,
    load_in_8bit = True,
    torch_dtype=torch.float16,
    device_map="auto"
)
model = PeftModel.from_pretrained(
    model,
    MODEL_NAME,
    torch_dtype=torch.float16,
    is_trainable = True
).to(device)

model.eval()

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, use_fast=False)
generation_config = GenerationConfig.from_pretrained(MODEL_NAME)
```

Рис. 8. Загрузка языковой модели и адаптера LoRA с указаниемконфигурационных параметров

На следующем этапе происходит загрузка датасета с использованием библиотеки dataset load_dataset. При этом датасет разбивается на две части – тренировочную и валидационную. Такое разделение нужно для того, чтобы избежать перетренированности модели. Это показано на рис. 9.

```
data = load_dataset(
    "json",
    data_files={'train' : '/kaggle/input/saigamistral/ML.json' ,
                'validation' : '/kaggle/input/saigamistral/Val.json'
               }
)
```

Рис. 9. Загрузка датасета

Далее необходимо произвести предобработку датасета, чтобы каждый элемент был словарем с ключами input_ids, attention_mask, labels, что показано на рис. 10.

```
CUTOFF_LEN = 3584
def generate_prompt(data_point):
    promt = f"""<s>system
{data_point['system']}</s><s>user
{data_point['user']}</s><s>bot
{data_point['bot']}</s>"""
    return promt
def tokenize (prompt, add_eos_token=True):
    result = tokenizer(
        prompt,
        truncation=True,
        max_length=CUTOFF_LEN,
        padding=False,
        return_tensors=None,
    )
    if (
        result["input_ids"][-1] != tokenizer.eos_token_id and len(result["input_ids"]) < CUTOFF_LEN
        and add_eos_token
    ):

        result["input_ids"].append(tokenizer.eos_token_id)
        result["attention_mask"].append(1)

    result["labels"] = result["input_ids"].copy()

    return result
def generate_and_tokenize_prompt(data_point):
    full_prompt = generate_prompt(data_point)
    tokenized_full_prompt = tokenize(full_prompt)
    return tokenized_full_prompt

train_data = (
    data["train"].map(generate_and_tokenize_prompt)
)
val_data = (
    data['validation'].map(generate_and_tokenize_prompt)
)
```

Рис. 10. Предобработка данных машинного обучения

Далее необходимо задать параметры машинного обучения, к их числу относятся следующие:

– per_device_train_batch_size - размер набора, обрабатываемого в одном цикле на каждую видеокарту;

– gradient_accumulation_steps - накапливание градиента для экономии видеопамяти;

– max_steps - количество шагов обучения;

– learning_rate - скорость обучения;

– logging_steps - вывод метрики потерь через данное количество шагов, которое здесь зададим;

– optim – оптимизатор;

– evaluation_strategy - стратегия вычислений, так как у нас стоят шаги(steps), то записываем steps;

− eval_steps - подсчет метрики потерь каждые N шагов;

− save_steps - параметр сохранения контрольных точек обучения черезкаждые N раз;

− output_dir - параметр для сохранения модели и контрольных точек;

− save_total_limit - количество сохранённых контрольных точек;

− load_best_model_at_end - загрузка лучшей модели после обучения;

− report_to - сохранение истории обучения на какой-либо сервис;

− overwrite_output_dir - перезапись директории для сохранения модели.

Результат выставления этих параметров представлен на рис. 11.

```
BATCH_SIZE = 4
MICRO_BATCH_SIZE = 2
GRADIENT_ACCUMULATION_STEPS = BATCH_SIZE // MICRO_BATCH_SIZE
LEARNING_RATE = 3e-4
TRAIN_STEPS = 100
OUTPUT_DIR = "/kaggle/working/tmp"

training_arguments = transformers.TrainingArguments(
            per_device_train_batch_size=MICRO_BATCH_SIZE,
            gradient_accumulation_steps=GRADIENT_ACCUMULATION_STEPS,
            max_steps=TRAIN_STEPS,
            learning_rate=LEARNING_RATE,
            fp16=True,
            logging_steps=10,
            optim="adamw_torch",
            evaluation_strategy="steps",
            save_strategy="steps",
            eval_steps=10,
            save_steps=10,
            output_dir=OUTPUT_DIR,
            save_total_limit=10,
            load_best_model_at_end=True,
            report_to=None,
            overwrite_output_dir=True,
)
```

Рис. 11. Выставление параметров машинного обучения

На следующем этапе выбирается data_collator, производящий пакетную обработку данных и после его выбора запускается скрипт машинного обучения, что показано на рис. 12.

```
data_collator = transformers.DataCollatorForSeq2Seq(
    tokenizer, pad_to_multiple_of=8, return_tensors="pt", padding=True
)

device = torch.device('cpu')
peft_config = LoraConfig(
    r = 16,
    lora_alpha = 32,
    lora_dropout = 0.05,
    bias = "none"
)
model.to(device)
trainer = transformers.Trainer(
    model=model,
    train_dataset=train_data,
    eval_dataset=val_data,
    args=training_arguments,
    data_collator=data_collator
)
model.add_adapter(peft_config)
model = torch.compile(model)
trainer.train()
model.save_pretrained(OUTPUT_DIR)
```

Рис. 12. Запуск машинного обучения

Результатом машинного обучения является появление файла .py, однако для использования в рамках проекта этого недостаточно, так как LM Studio принимает формат gguf, в связи с чем необходима конвертация. Это можно осуществить с помощью проекта llama.cpp. Перед конвертацией нужно сохранить токенизатор в любую папку, после чего необходимо перейти в папку llama.cpp и задать необходимые параметры для конвертирования модели. К их числу относятся:

− model_dir - путь до модели;

− checkpoint - путь до сохранённого токенизатора;

− output_model – место для сохранения модели gguf [19-22].

После инициализации этих параметров необходимо запустить скрипт с помощью команды «python convert.py {model_dir} --vocab-dir {checkpoint} --outfile {output_model} --outtype f16 --ctx 4096». В результате этого будет сформирован файл gguf, соответствующей модели, прошедшей дополнительное обучение, которую уже можно развернуть в LM Studio, и настраивать для взаимодействия с веб-приложением.

Программная реализация веб-приложения

Веб-приложение представляет собой функционал, позволяющий администратору безопасности удобно взаимодействовать с нейросетевой реализацией, вести чаты по каждому хосту, сохраняя при этом их историю в отдельные списки. Ввиду того, что API, позволяющее получать ответы нейросети реализовано на языке python, для наилучшего взаимодействия стоит использовать веб-фреймворки именно этого языка. При этом существует два наиболее популярных фреймворка, позволяющих реализовать веб-приложение на python, это Flask [23], который больше подходит для проектирования небольших приложений, и Django [24], используемый в крупных проектах. Исходя из функционала веб-приложения администратора, его нельзя отнести к крупным проектам, задействующим множество технологий, таких как, например, подключение базы данных или платежной системы, поэтому логично произвести программную реализацию с помощью фреймворка Flask.

В любом приложение программное обеспечение можно логически разделить на две части: бэкенд, отвечающий за взаимодействие внутренних компонентов приложения, которое скрыто от пользователя, и фронтенд, представляющий собой презентационную часть веб-приложения, то, с чем напрямую будет взаимодействовать администратор безопасности. При этом, в рамках проекта на Flask,принято выносить весь код, связанный с фронтендом в отдельную папку templates, гдепрописывается логика сервиса (код html и java script), и static, в которой хранятся стили (в то время как файлы backend хранят в главной папке проекта). В проектируемом модуле было произведено такое же разделение. Опишем программное обеспечение каждого из приведенных сервисов [23].

Бэкенд-сервис

Перед реализацией данного сервиса необходима определить перечень задач и требований. Исходя из функционала, который должно обеспечивать веб-приложения, данный перечень состоит из следующих пунктов:

− обеспечение прослушивания в отдельном потоке входящих соединений локальной сети с целью получения полезных данных о сценарии атаки, произошедшем на конкретном хосте;

− получение данных от фронтенд-сервиса и формирование запроса локальной языковой модели;

− настройка взаимодействия с языковой моделью, развернутой на локальном или удаленном сервере;

− прием сообщений от языковой модели и передачи их в фронтенд-сервис;

− получение данных от скрипта определения ip-адресов активных устройств локальной сети и автоматическое формирование списка чатов, на основании этой информации;

− обеспечение хранения истории чата для каждого хоста по-отдельности.

Весь бэкенд сервис реализован в виде

файла main, который логически можно разбить на следующие части:

− инициализация констант и глобальных переменных;

− часть, отвечающая за прослушивание входящих соединений локальной сети с целью получения информации о сценарии атаки;

− обработчики http-запросов;

− метод, отвечающий за взаимодействие с локальной языковой моделью. При этом к данном файлу необходимо подключить следующие библиотеки:

− threading, которая отвечает за создание потоков, в данном случае она будет обеспечивать бесконечное прослушивание входящих соединений [25];

− requests, необходимая для осуществления приема и передачи http-запросов, что поможет обеспечить удобное взаимодействие с фронтенд-сервисом;

− lmAPI, являющаяся самописной библиотекой для взаимодействия с языковой моделью;

− apiFuncs, которая также была написана самостоятельно в рамкахдипломного проекта с целью обеспечения более удобного парсинга, чтения и записиразличных данных, использующихся в приложении;

− flask, подключающая некоторые классы бекенд-фреймворка, необходимые для отправки http-запросов с помощью обработчиков, создания динамической генерации html-страниц и запуска самого приложения [23].

В самом начале программной реализации бэкенд-сервиса идет инициализация переменной, отвечающей за запуск веб-приложения, и создание констант, применяемых для прослушивания входящих соединений (app.UDP_PORT_LOCAL, app.UDP_PORT_LISTEN), определяющих адрес сайта и его порт (app.SITE_ADDRESS), хранящих список ip-адресов автоматизированных информационных систем из локальной сети (app.HOST_IPS). Кроме того, создается динамическая переменная, в которую можно добавлять сообщения чата, зная ip-адрес хоста этого чата. Эта инициализация показана на рис. 13.

```
8   app = Flask(__name__)
9
10  app.UDP_PORT_LOCAL = 12345
11  app.UDP_PORT_LISTEN = 54321
12  app.SITE_ADDRESS = "http://127.0.0.1:5000"
13  app.HOST_IPS = af.read_ip()
14  app.chat_history = []
15
16  for i in app.HOST_IPS:
17      app.chat_history.append({
18          "ip": i,
19          "messages": []
20      })
```

Рис. 13. Инициализация системных параметров бэкенд-сервиса

Следующей частью сервиса является создания потока, который находится в бесконечном прослушивании входящих соединений. Создается сокет, который привязывается к локальному хосту с портом 12345, который ждет в бесконечном цикле прихода пакета из локальной сети. По приходу UDP-пакета данные подвергаются парсингу и в файл udp.txt помимо сценария атаки записывается и ip- адрес, с которого был получен пакет. После чего происходит отправка post запроса для вызова обработчика, отправляющего запросы в нейросеть [27]. Это показано на рис. 14.

```
def listen_udp_socket():
    # Создаем сокет
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Привязываем сокет к IP и порту
    sock.bind(('0.0.0.0', app.UDP_PORT_LOCAL))

    while True:
        # Получаем данные от клиента
        data, address = sock.recvfrom(app.UDP_PORT_LISTEN)
        # Записываем данные в файл
        af.write_udp(f"{address[0]}+++{data.decode()}")
        requests.post(app.SITE_ADDRESS, data=address[0])
app.udp_thread = threading.Thread(target=listen_udp_socket)
app.udp_thread.start()
```

Рис. 14. Прослушивание входящих соединений

Далее происходит инициализация обработчиков. Первый обработчик передает в веб-приложение данные о списке ip-адресов локальной сети, на основании которых автоматически формируются чаты, это отображено на рис. 15.

```
@app.route("/")
def index():
    return render_template('chat.html', host_ips=app.HOST_IPS)
```

Рис. 15. Обработчик главной страницы

Наибольшее значение в рамках функционирования веб-приложения играет второй обработчик,ибо он отвечает за обработку get и post запросов. На выходе он выдает get-запрос в фронтенд-сервис с ответом нейросети. Также он обеспечивает работу приложения в двух режимах:

− в режиме регистрации вторжения;

− в режиме интеллектуальных подсказок.

Это достигается за счет введения условия, согласно которому в случае наличия данных о сценарии атаки в файле udp.txt на их основании формируется запрос в нейросеть, после чего файл очищается. В противном случае идет чтение из поля ввода веб-приложения и в качестве запроса нейросети подается сообщение пользователя без дополнительных обработок. Также стоит отметить, что в данном методе существует проверка, согласно которой, если было совершено обновление страницы, то необходимо очистить историю чата. Проверка осуществляется за счет получения post-запроса со стороны фронтенд-сервиса, причем результат наличия обновления страницы описывает переменная update. Если она равна 0, то обновление былосделано и стоит очистить истории всех чатов [28]. Программная реализация данной части бэкенд-сервиса показана на рис. 16.

```
@app.route("/get", methods=["GET", "POST"])
def chat():
    global index
    global input
    udp_packet = af.read_udp()
    print(udp_packet)
    if len(udp_packet) != 0:
        host_address = af.get_ip_from_udp()
        attack_data = af.get_data_from_udp()
        attack = af.get_attack_number(attack_data)
        vulnerabilities = af.get_vulnerabilitys_list(attack_data)
        input = f"Напиши меры противодействия для атаки {attack}, , учитывай при этом уязвимость: " \
                f"\'{vulnerabilities[0]}\'"
        count = 0
        for ip in app.HOST_IPS:
            if ip == host_address:
                index = count
            count += 1
        af.clear_udp()
    else:
        msg = request.form["msg"]
        index = int(request.form["index"])
        check_update = int(request.form["update"])
        if check_update == 0:
            af.clear_chat_history(app.chat_history)
        input = msg
    print(f"input: {input}")
    return get_Chat_response(input, index)
```

Рис. 16. Обработчик get и post - запросов

Причем стоит отметить, что обработчик get и post - запросов в своем выводе обращается к функции get_Chat_response(), которая занимается формированием промтов на вход нейросетевой реализации. Опишем более подробно как это происходит. Для начала информация, которая была получена из обработчика сохраняется в локальную историю хоста по ip-адресу под ролью «user». Причем ip- адрес определяется на основании индекса, который подается на вход метода наряду с пользовательским сообщением. Далее происходит передача всей локальной истории в функции формирования промта языковой модели, которая дополняется системнымсообщением для форматирования ввода и концентрации внимания на специфике регламентации мер противодействия. После того как ответ от нейросети был получен, он также записывается в локальную историю чата хоста с ролью «assistant». Далее ответ дополняется значением индекса, что обеспечивает вывод сообщения в нужныйчат, и подается на выход функции [28]. Это показано на рис. 17.

```
def get_Chat_response(text, index):

    local_history = app.chat_history[index]["messages"]
    app.chat_history[index]["messages"].append({"role": "user", "content": text})
    response = lapi.get_response(local_history)
    # ждем ответ от нейросети
    app.chat_history[index]["messages"].append({"role": "assistant", "content": response})
    response = f"{index}+++{response}"
    return response
```

Рис. 17. Метод взаимодействия с языковой моделью

Кроме описанных функций существует еще одна – main, которая отвечает за запуск приложения. Программное обеспечение бэкенд-сервиса позволяет реализовать функционал взаимодействие компонентов, которое скрыто от пользователя. Однако для создания полнофункционального веб-приложения необходимо провести реализацию фронтенд-сервиса.

Фронтенд-сервис

Приведем описание фронтенд-сервиса, необходимого для формирования презентационной части веб-приложения, которая позволит администратору безопасности удобно взаимодействовать с компонентами бэкенд-сервиса, регламентируя при этом меры противодействия сетевым вторжениям. К основным задачам данного сервиса относятся следующие:

− прием сообщений от пользователя и передача их в бэкенд-сервис;

− вывод ответов нейросетевой реализации в диалоговом окне веб-приложения;

− обеспечение вывода сообщений в чате таким образом, чтобы можно было отличить запросы пользователя от ответов языковой модели;

− прием данных от бэкенд-сервиса об ip-адресах автоматизированных информационных систем локальной сети, и создание на основе этих сведений кнопок,по нажатию на которых можно было бы перейти в чат регламентации мер противодействия для каждого хоста;

− удаление сообщений чата по событию обновления страницы.

Программное обеспечение для фронтенд-сервиса, согласно правилам проектирования подобных приложений, принято хранить в папках «static» и «template». Причем в папке «static» хранятся стили классов html в формате css и картинки, применяемые в веб-приложения, а в «template» - верстка сайта и логика функционирования его динамических элементов, что представляет наибольший интерес в рамках проектируемого сервиса [29].

В папке «template» находится всего один файл, отвечающий за разметку и функционирование единственной страницы chat.html. Его можно разделить на две части – html, которая описывает статическую составляющую веб-приложения, и javascript, отвечающая за работу динамических компонентов. Приведем подробное описание каждой из них.

Кроме файла style.css, находящегося в папке «static» стилизацию позволяют обеспечить интернет-стили, которые подключаются в элементе с тэгом «head», как показано на рис. 18 [30].

```
<div class="col-md-8 col-xl-6 chat">
    <div class="card">
        <div class="card-header msg_head">
            <div class="bd-highlight">
                <div class="img_cont">
                    <img src="https://i.ibb.co/fSNP7Rz/icons8-chatgpt-512.png"
                        class="rounded-circle user_img">
                    <span class="online_icon"></span>
                </div>
                <div class="user_info">
                    <span>SIBGPT</span>
                    <p>Задай мне вопрос!</p>
                </div>
            </div>
        </div>
        <div id="messageFormeight" class="card-body msg_card_body">
```

Рис. 18. Подключение интернет-стилей

Далее инициализируются основные элементы веб-приложения. По сути, к ним относятся: карточка чата, текстовое поле, в которое пользователь вводит данные, кнопка отправки, кнопки чатов с ip-адресами хостов локальной сети. Сначала создаются кнопки чатов, причем, так как нет фиксированных ip-адресов, эта информация берется из переменных, передаваемых бэкенд-сервисом, которые выводятся в цикле, создавая при этом кнопки, соответствующие чатам. Такой способ достаточно удобен, так как при добавлении нового компьютера в локальную сеть будет автоматически создан новый чат для этого хоста. Создание кнопок чатов хостов локальной сети проиллюстрировано на рис. 19.

```
<div class="d-flex host-ips">
    {% for ip in host_ips %}
        <button class="ip-button" name="host-button" value="{{ip}}">хост {{ ip }}</button>
    {% endfor %}
</div>
```

Рис. 19. Создание кнопок чатов хостов локальной сети

Следующим компонентом является карточка, необходимая для вывода текстовых данных пользователя и бота на основе искусственного интеллекта. Стоит отметить, что она является единой для каждого чата, а обновление сообщений в ней осуществляется за счет скрипта js. Вверху данной карточки присутствует иконка нейросети с названием «SIBGPT» и предложением задать ей вопрос, подтягивающаяся автоматически из Интернета [30]. Это информация находит отражение на рис. 20.

```
<div class="col-md-8 col-xl-6 chat">
    <div class="card">
        <div class="card-header msg_head">
            <div class="bd-highlight">
                <div class="img_cont">
                    <img src="https://i.ibb.co/fSNP7Rz/icons8-chatgpt-512.png"
                        class="rounded-circle user_img">
                    <span class="online_icon"></span>
                </div>
                <div class="user_info">
                    <span>SIBGPT</span>
                    <p>Задай мне вопрос!</p>
                </div>
            </div>
        </div>
        <div id="messageFormeight" class="card-body msg_card_body">
```

Рис. 20. Создание карточки вывода сообщений

И последним важным элементом является форма «messageArea», необходимая для обеспечения приема сообщений пользователя. Она содержит элемент ввода данных «textarea», который был выбран для возможности записи в запрос знака переноса строки, что не позволяет произвести стандартный элемент «text» [29]. Наряду с элементом ввода в данной форме присутствует кнопка, позволяющая отправить данные в бэкенд-сервис. Программная реализация формы ввода данных представлена на рис. 21.

```
<form id="messageArea" class="input-group">
    <textarea type="text" id="text" name="msg" placeholder="Type your message..."
              autocomplete="off" class="form-control type_msg" required></textarea>
    <div class="input-group-append">
        <button type="submit" id="send" class="input-group-text send_btn">
            <i class="fas fa-location-arrow"></i>
        </button>
    </div>
</form>
```

Рис. 21. Форма ввода данных

Как было отмечено ранее, наряду с html-разметкой в данном файле присутствует js скрипт. Для возможности его интеграции в файл html необходимо использовать тэг «script». Говоря про его структуру, можно отметить, что он содержит несколько обработчиков и функций, а в самом начале инициализируется двумерный массив, хранящий информацию об истории чата. К числу функций относятся следующие:

− checkMessageLengths, необходимая для проверки длины всех подмассивов массива, хранящего историю чатов каждого хоста;

− extractCountAndData – парсинг принятой информации из бэкенд-сервиса и возврат структуры, содержащей порядковый номер чата, в который необходимо добавить сообщение и полезных данных, и само сообщение.

Кроме того, js скрипт содержит 3 обработчика:

− обработчик события «submit» (нажатие на кнопку в форме message area), необходимый для работы веб-приложения в режиме интеллектуальных подсказок, когда принимаются и выводятся сообщения администратора, а затем отображается ответ нейросети;

− обработчик кнопки «f2», покрывающий работу приложения в режиме регистрации вторжения, при котором формируется скрытый запрос пользователя, не отображающийся в карточке сообщений, и выводится ответ языковой модели с регламентами противодействия;

− обработчик кликов на кнопки чатов, нужный для того, чтобы при нажатии на какую-либо кнопку чата она выделялась другим цветом, что позволяет пользователю понимать в каком чате он работает (стандартные механизмы html позволяют сделать такое выделение только на время нажатия на кнопку, то есть когдапользователь ее отжимает кнопка возвращает прежний свой прежний цвет) [31].

Взаимодействие с бэкенд-сервисом также реализовано в js-скрипте и оно обеспечивается за счет применения ajax-запросов. Когда пользователь вводит данныеи производит их отправку, то вызывается ajax-запрос типа post, передающий сообщение пользователя, номер чата, в котором было сделано это сообщение, и переменную, хранящую информацию о том, было ли сделано обновление или нет на локальный адрес веб-приложения «/get» [32]. На стороне бэкенд-сервиса в обработчики эти данные принимаются, записываются в локальные переменные, на основание которых формируется запрос локальной языковой модели. По пришествии ответа осуществляется http-запрос типа get, который передается обработчик выполнения ajax-запроса фронтенд-сервиса. В нем данные подвергаются парсингу и выводятся на экран с иконкой нейросети [33].

Примеры практического использования сервиса в различных режимах его работы

Проведя программную реализацию сервиса регламентации мер противодействия, произведем демонстрацию эффективности его работы на реальныхпримерах.

Так как данный модуль используется в рамках общей автоматизированной системы агрегирования данных, обнаружения, идентификации сетевых вторжений и регламентации мер противодействия, то для эксперимента необходимо воссоздать локальную сеть, содержащую персональные

компьютеры, на которые совершаются кибератаки, поместив туда ПК администратора безопасности. Пусть компьютер администратора безопасности имеет ip-адрес 192.168.1.51, а пул адресов автоматизированных информационных систем имеет следующий диапазон – от 192.168.1.55 – 192.168.1.57. Схема данной сети приведена на рис. 22.

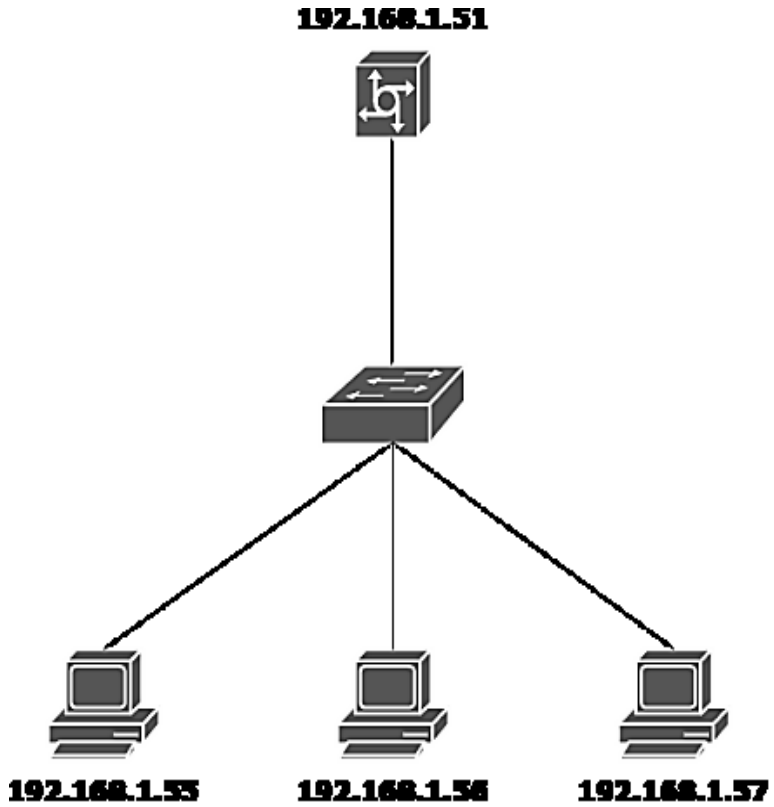

Рис. 22. Схема демонстрационного стенда

Перед демонстрацией работы сервиса необходимо развернуть модуль идентификации сетевых вторжений на каждой автоматизированной информационной системе. Далее необходимо произвести развертывание прошедшей дополнительное машинное обучение языковой модели в рамках LM Studio и запуск серверного варианта ее функционирования. В случае, если модель была запущена на локальном хосте, вносить изменения в код не стоит, однако, если запуск производился на удаленном специализированном сервере, то придется в файле проекта lmAPI.py поменять константу NEIRO_SERVER_ADDRESS на статический адрес сервера.

После проведения подобных манипуляций можно свободно пользоваться функционалом модуля регламентации мер противодействия. Сначала необходимо запустить скрипт определения ip-адресов активных устройств локальной сети. Это показано на рис. 23, 24.

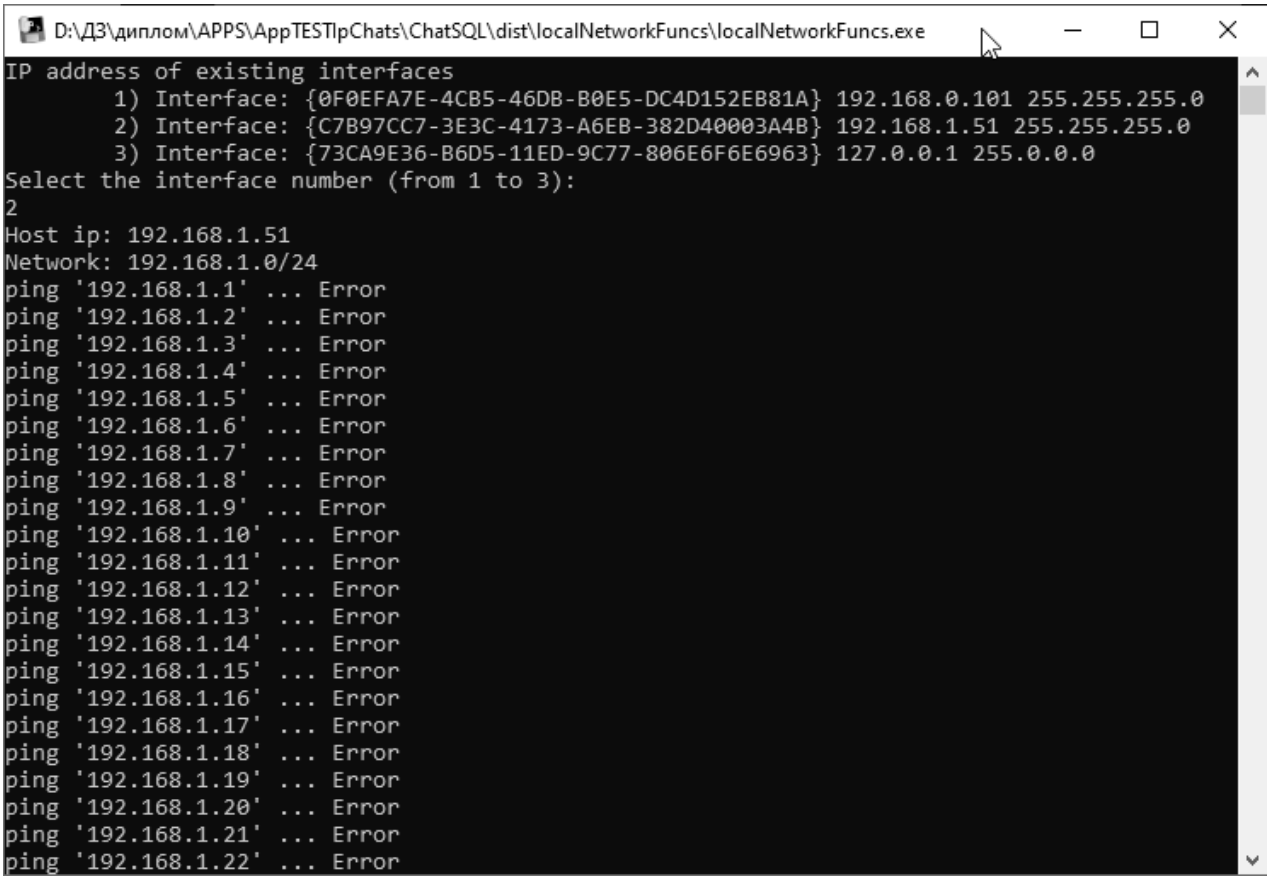

Рис. 23. Тестирование скрипта определения списка ip-адресов в сети

Рис. 24. Фиксация ip-адресов активных хостов локальной сети

Для того, чтобы избежать проверки связи «сам с собой», ip-адрес компьютера администратора исключается из списка адресов, подлежащих пингованию.

В результате выполнения данного скрипта формируется файл ip.txt, содержание которого можно видеть на рис. 25.

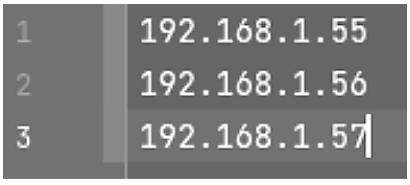

Рис. 25. Фиксация ip-адресов активных хостов локальной сети

Данный файл продемонстрирован в среде разработки PyCharm, так как его необходимо помещать в тело проекта веб-приложения, обеспечивая при этом удобное взаимодействие со всеми его сервисами.

После того как модуль был развернут в рамках локальной сети можно провести его тестирование. На основании файла ip.txt, полученного в результате выполнения скрипта определения списка ip-адресов активных хостов локальной сети, формируется интерфейс веб-приложения, а именно вкладки чатов, которые соответствуют выявленным адресам. Интерфейс веб-приложения при его запуске показан на рис. 26.

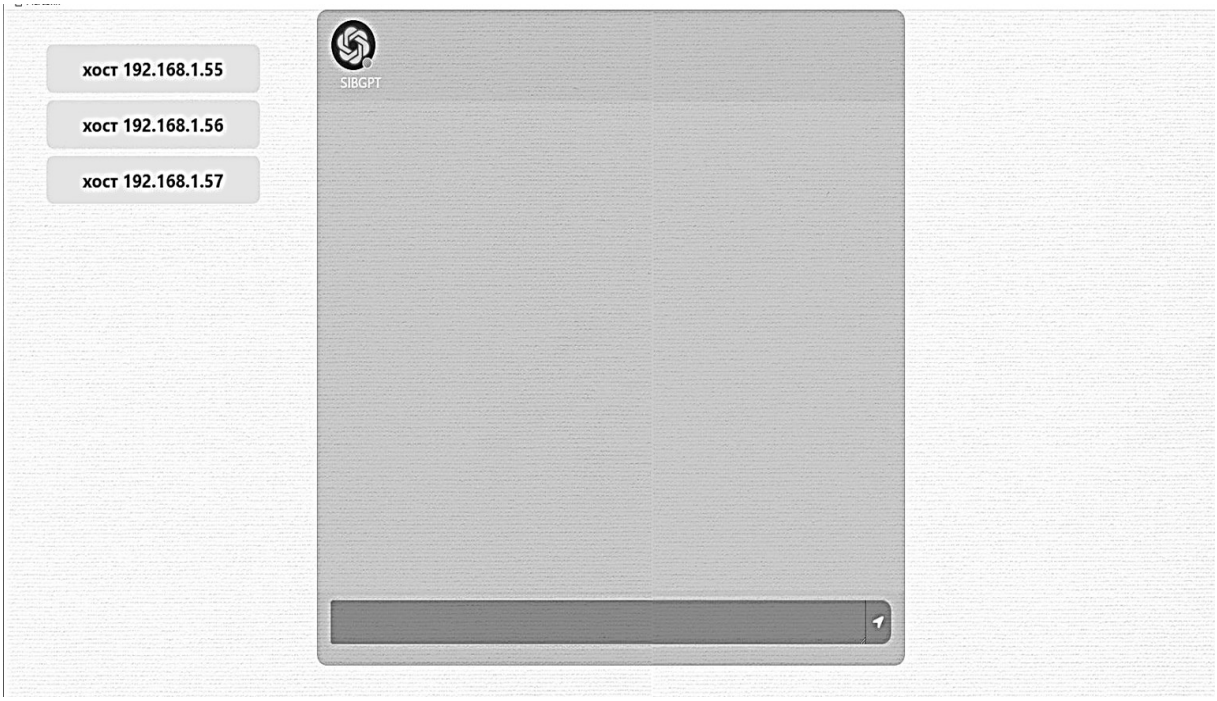

Рис. 26. Интерфейс веб-приложения при первом запуске

Согласно методическому обеспечению, нейросетевой модуль регламентации мер противодействия способен работать в двух режимах:

− в режиме регистрации вторжения;

− в режиме интеллектуальных подсказок.

Для наиболее наглядного представления о возможностях модуля в рамках регламентации мер противодействия проведем демонстрацию его работы в двух режимах. В конце покажем работу модуля в комбинации режимов регистрации вторжения и интеллектуальных подсказок. Причем данное тестирование будем проводить наряду со сравнением результатов, выдаваемых аналогами. Ввиду большого количество аналогов в сформированных группах, возьмем по одному из каждой и проведем сравнение лишь в режиме интеллектуальных подсказок, в связи с невозможностью их работы в других режимах. Стоит отметить также, что данное тестирование в режимах регистрации вторжения и комбинированном режиме будем проводить вместе с модулем идентификации сетевых вторжений. В общем случае в качестве атак, подаваемых в качестве входных данных для тестирования, выберем несколько атак из различных классов согласно классификации CAPEC. Данный выбор обусловлен относительной простотой их реализации на стендовой установке и проиллюстрирован в табл. 1.

Таблица 1

Выбор атак для демонстрации работы сервиса

Идентификатор CAPEC	**Название атаки**	**Класс атаки согласно CAPEC**
CAPEC-482	TCP-наводнение (флуд)	Злоупотребление функционалом
CAPEC-297	Сканирование портов (TCP ACK)	Анализ целевого объекта
CAPEC-49	Подбор паролей методом грубой силы	Использование вероятностных методов

Во избежание одинаковых результатов в рамках тестирования, необходимо провести распределение входных данных для каждого режима работы сервиса. Так как режим интеллектуальных подсказок предполагается протестировать наряду с аналогами, то в данном режиме тестирование будет проводиться для всех атак, причембез участия модуля идентификации сетевых вторжений. Для режима регистрации вторжения используем CAPEC-49 и CAPEC-482. В рамках тестирования комбинированного режима возьмем атаку CAPEC-297. Более наглядное сопоставление режимов сервиса и его входных данных представлено в табл. 2.

Таблица 2

Сопоставление режимов работы сервиса и входных данных

Режим работы сервиса	**Входные данные**	**Использование модуля идентификации сетевых вторжений**
Режим регистрации вторжения	CAPEC-49, CAPEC-482	Используется
Режим интеллектуальных подсказок	CAPEC-49, CAPEC-297, CAPEC-482	Не используется
Комбинированный режим	CAPEC-49, CAPEC-297	Используется

Также стоит отметить, что для эмуляции атак на стороне модуля идентификации сетевых вторжений будут использованы следующие приложения:

− patator для реализации CAPEC-49,
− nmap/zenmap для CAPEC-297,
− hoic для CAPEC-482.

Консультационный режим в момент регистрации вторжения

Первым режимом является режим регистрации вторжения. Модуль должен принимать udp-пакет о сценарии атаки, производить его парсинг, и на основании полученных сведений выдавать меры противодействия. Стоит отметить, что в таком режиме меры выдаются лишь на основании сценария атаки и предполагается, что администратор безопасности не будет их корректировать. Такой подход можно использовать лишь для консультационных потребностей администратора безопасности, причем меры будут не таким и точными, так как не учитывается специфика инфраструктуры предприятия и возможности имеющихся защитных механизмов. Протестируем модуль в данном режиме, взяв в качестве тестовых данных следующие атаки:

− CAPEC-482, реализованную на хосте 192.168.1.55,
− CAPEC-49, реализованную на хосте 192.168.1.56.

Для успешного обнаружения данных необходимо зафиксировать их траффик с помощью Wireshark сохранить его в файл .pcap. Результаты перехвата показаны на рис. 27, 28.

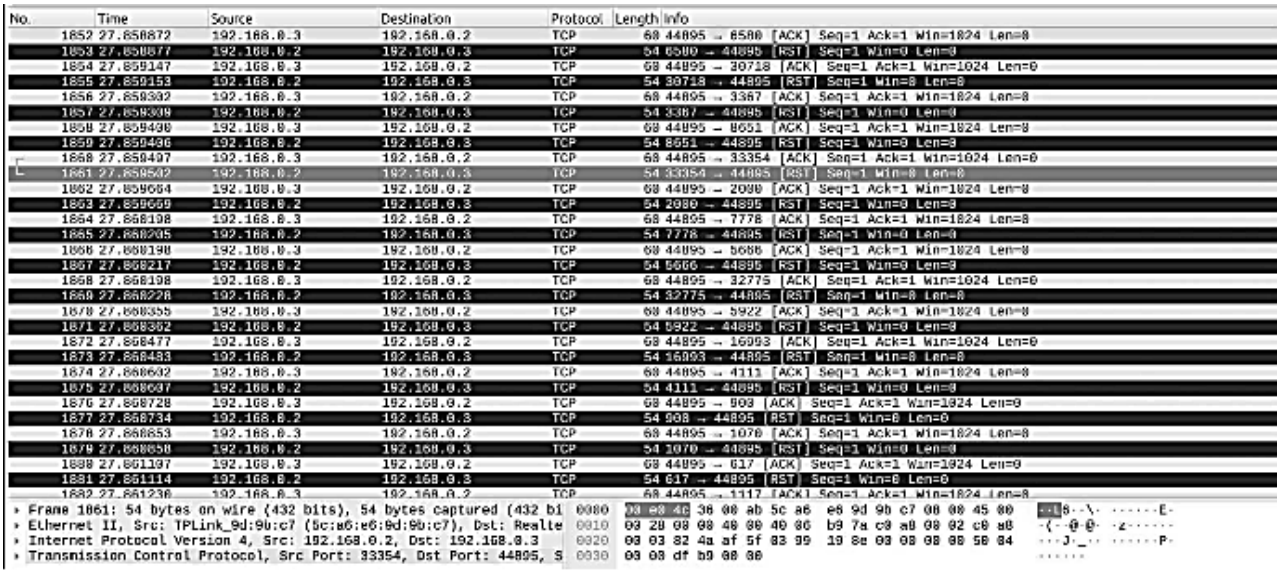

Рис. 27. Результата перехвата траффика для CAPEC-482

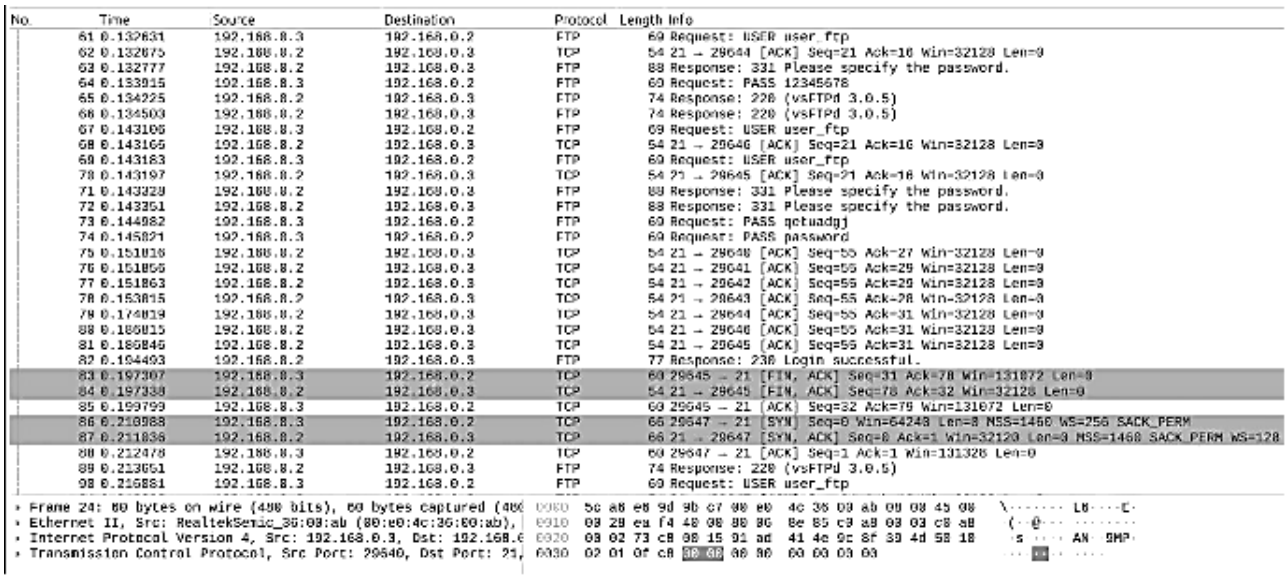

Рис. 28. Результат перехвата траффика для CAPEC-49

На следующем этапе файлы перехваченного траффика проходят конвертацию из формата .pcap в .flows, что необходимо для корректной работы нейросети, идентифицирующей атаку.

Далее происходит подача полученных файлов в нейросеть, идентифицирующую атаку. Результаты работы данной нейросетевой реализации показаны на рис. 29, 30.

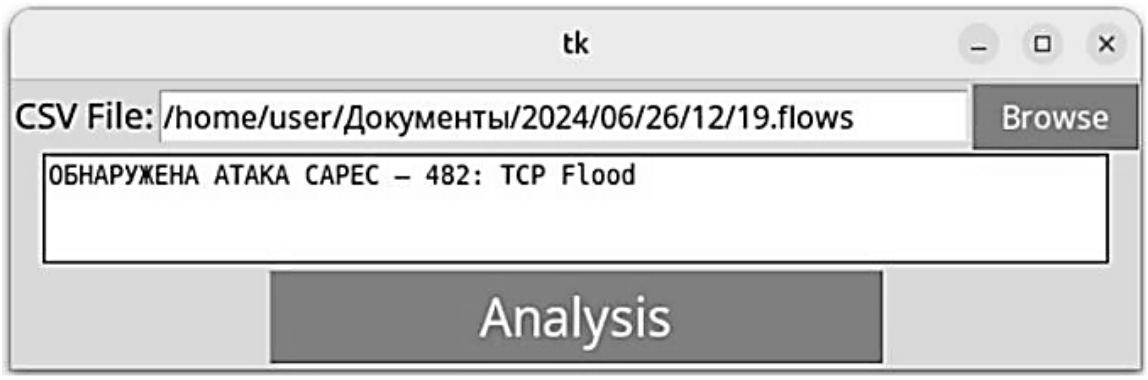

Рис. 29. Идентификация CAPEC-482

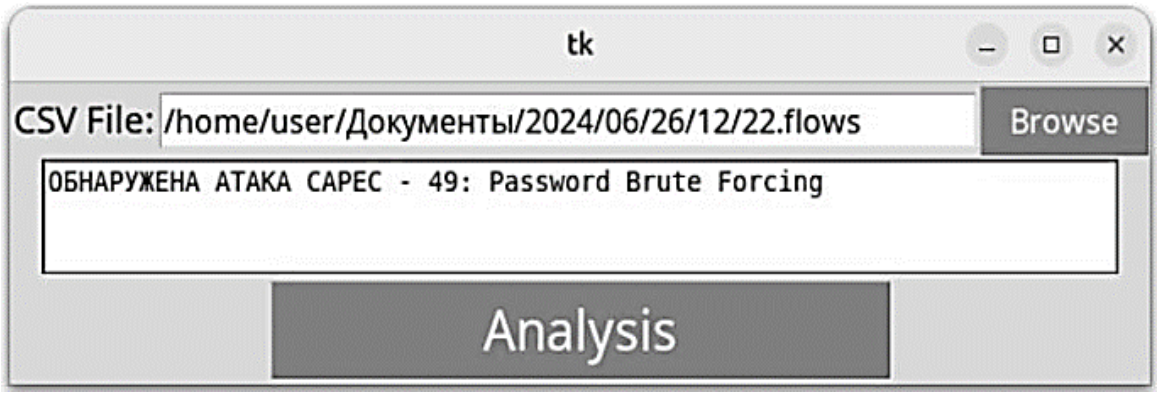

Рис. 30. Идентификация CAPEC-482

Согласно последним двум рисункам, атаки были правильно идентифицированы, в результате чего произошла отправка UDP-пакетов с полезными сведениями осценарии в модуль регламентации мер противодействия. Непосредственная регламентация этих мер приведена на рис. 31, 32.

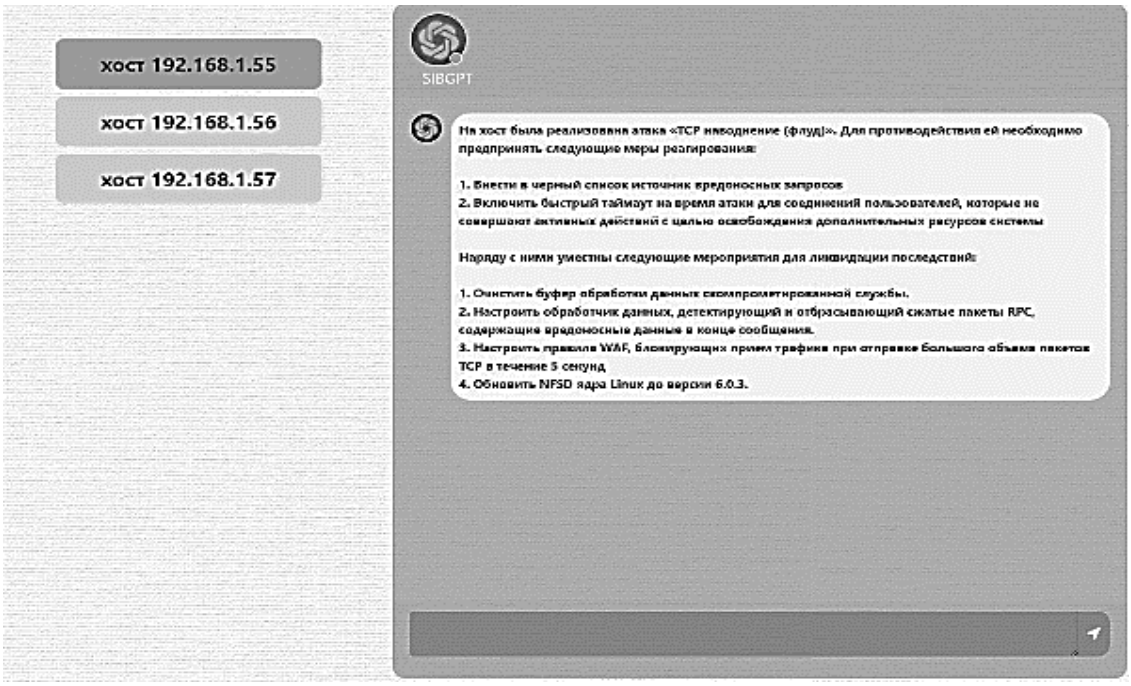

Рис. 31. Выдача мер противодействия для CAPEC-482 в режиме регистрации вторжения

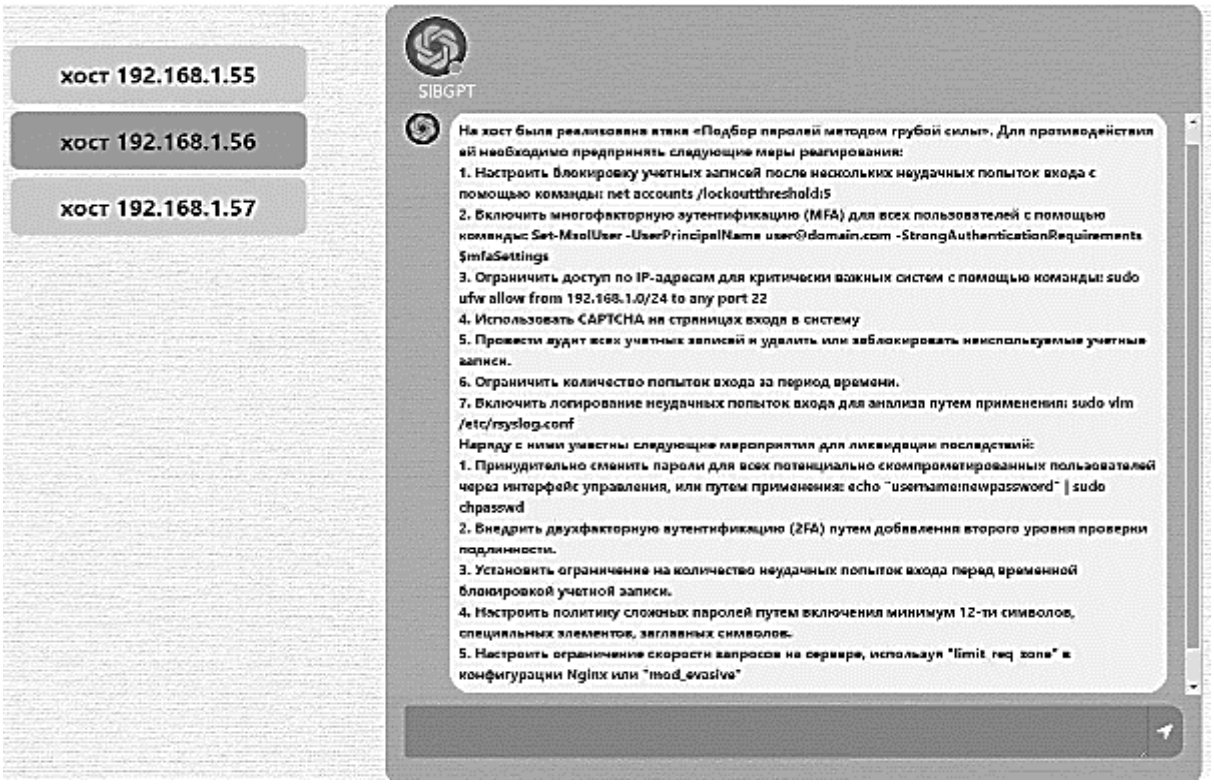

Рис. 32. Выдача мер противодействия для CAPEC-49 в режиме регистрации вторжения

Исходя из результатов, отраженных на рис. 31, 32, отметим, что были правильно идентифицированы хосты, на которых реализовывались атаки, кроме того, выданы валидные меры противодействия, с правильным названием атаки по идентификатору CAPEC.

Режим интеллектуальных подсказок

Второй режим работы модуля, в отличие от первого, предусматривает прием сведений о сценарии атаки от пользователя, при этом сообщение лишено дополнительного парсинга, без обработки подаваясь на вход языковой модели. После получения мер противодействия администратор безопасности имеет возможность их скорректировать или получить дополнительное разъяснение по применению мероприятия, с помощью отправки наводящих вопросов. Также стоит отметить, что в данном режиме изначально можно вводить не только сценарий атаки, но и любой вопрос, связанный с информационной безопасностью, что является хорошим подспорьем для использования в образовательных целях, обучая при этом либо сотрудников компании, либо студентов по специальностям, связанным с ИБ. Также можно использовать режим в консультативных целях, однако выходные меры могут быть несовсем точными, так как при таком варианте не будет достоверно известен сценарийатаки. Как было оговорено ранее, протестируем в этом режиме спроектированный модуль наряду с аналогами для наглядной демонстрации отличия выдаваемых результатов. Возьмем из категории аналогов, основанных на OpenAI API, MITREGPT и Soc copilot, а из группы собственных нейросетевых реализаций Threat Intel Bot. Протестируем модуль в данном режиме, взяв в качестве тестовых данных следующиеатаки:

− CAPEC-482, реализованную на хосте 192.168.1.55,

− CAPEC-49, реализованную на хосте 192.168.1.56,

− CAPEC-297, реализованную на хосте 192.168.1.57.

Протестируем выдачу ответов проектируемого сервиса в рамках регламентации мер противодействия для атаки CAPEC-482 в сравнении с результатами работы Soc Copilot. Результаты работы Soc Copilot приведены на рис. 33-35.

Рис. 33. Вывод мер противодействия Soc Copilot для CAPEC-482

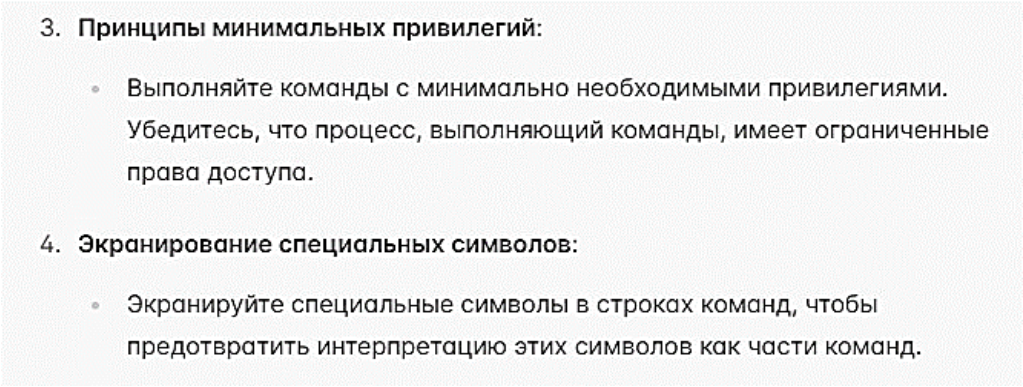

Рис. 34. Вывод мер противодействия Soc Copilot для CAPEC-482

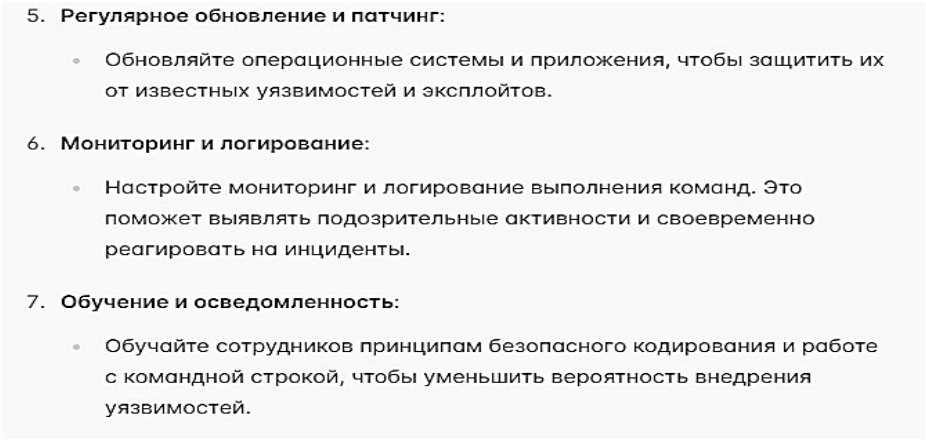

Рис. 35. Вывод мер противодействия Soc Copilot для CAPEC-482

Результаты вывода мер противодействия для атаки TCP наводнение (флуд) сервиса нейросетевой регламентации представлены на рис. 36.

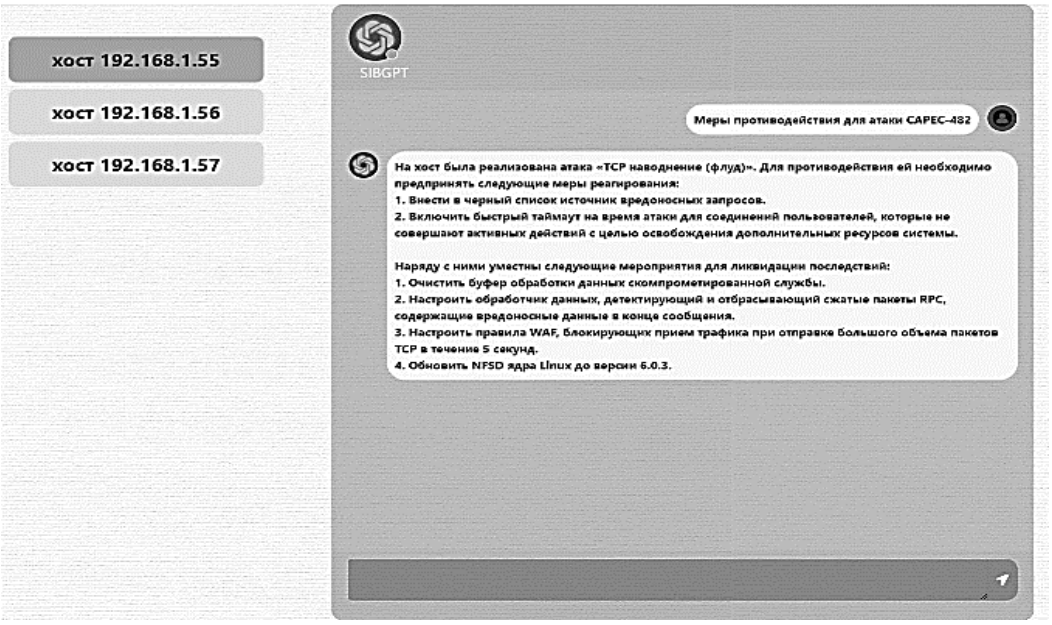

Рис. 36. Вывод мероприятий по борьбе с CAPEC-482 сервиса нейросетевой регламентации мер противодействия в режиме интеллектуальных подсказок

Сравнивая эти два вывода, можно отметить, что Soc Copilot не смог правильно определить атаку по ее CAPEC-идентификатору, что делает дальнейший вывод мер противодействия некорректным. Также при выводе нет разделения на меры реагирования и ликвидации последствий, само содержание мероприятий недостаточно раскрыто, нет конкретной последовательности действий, позволяющей пользователю эффективно среагировать на сетевое вторжение. Эти недостатки минимизируются при регламентации мер противодействия спроектированным сервисом. Далее для, того чтобы убедиться в корректности работы сервиса при выдачи интеллектуальных подсказок и корректировки мер, зададим нейросетевой реализации наводящие вопросы. Результаты работы модуля в рамках выдачи интеллектуальных подсказок показаны на рис. 37-40.

Рис. 37. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-482

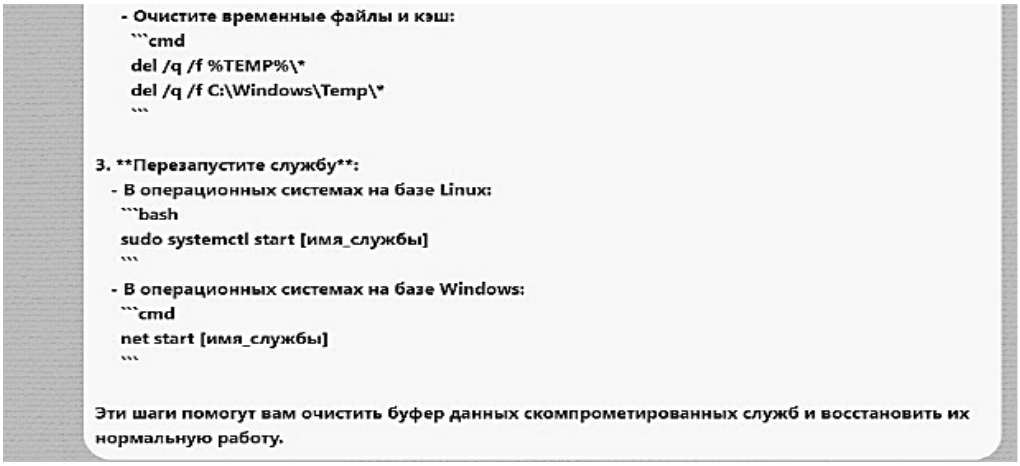

Рис. 38. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-482

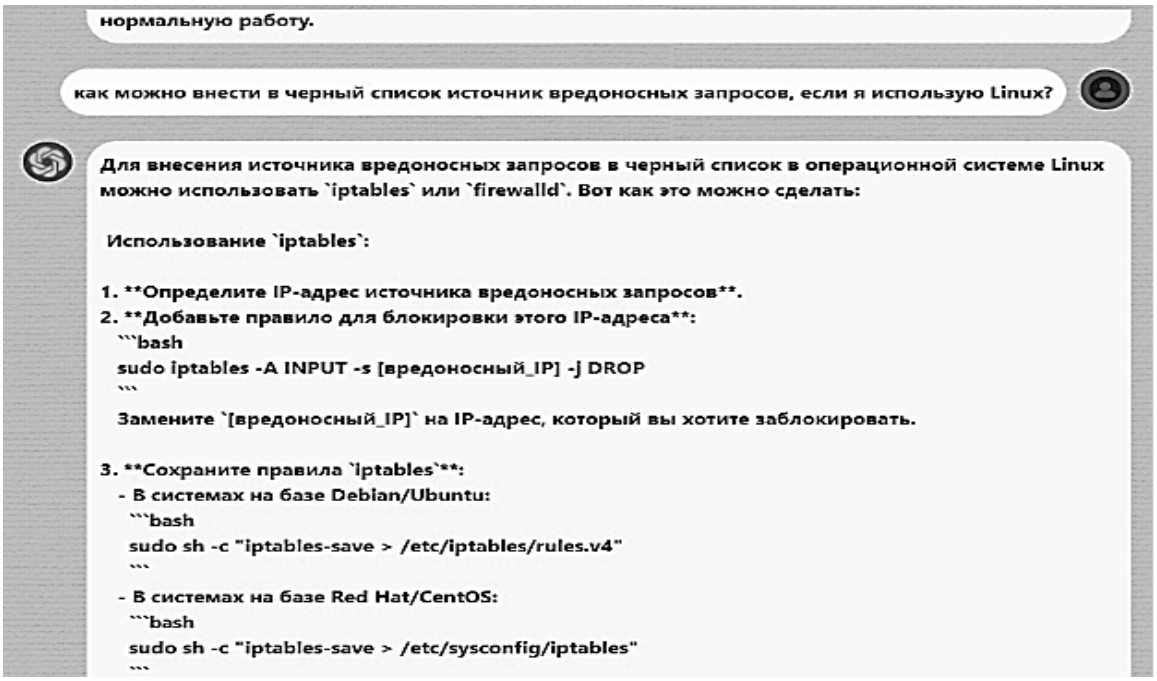

Рис. 39. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-482

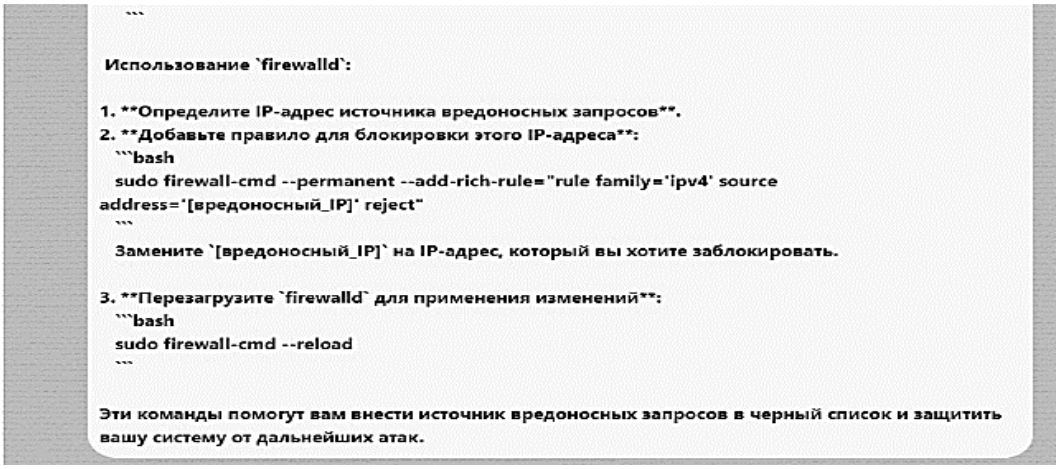

Рис. 40. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-482

Из рис. 37-40 видно, что подсказки были выданы довольно подробно с предоставлением конкретной последовательности действий для пользователя, что схоже с изначальной выдачей мер противодействия.

На следующем тесте сравним вывод спроектированного нейросетевого сервиса с результатами работы MITREGPT. Вывод MITREGPT для атаки CAPEC-49 представлен на рис. 41.

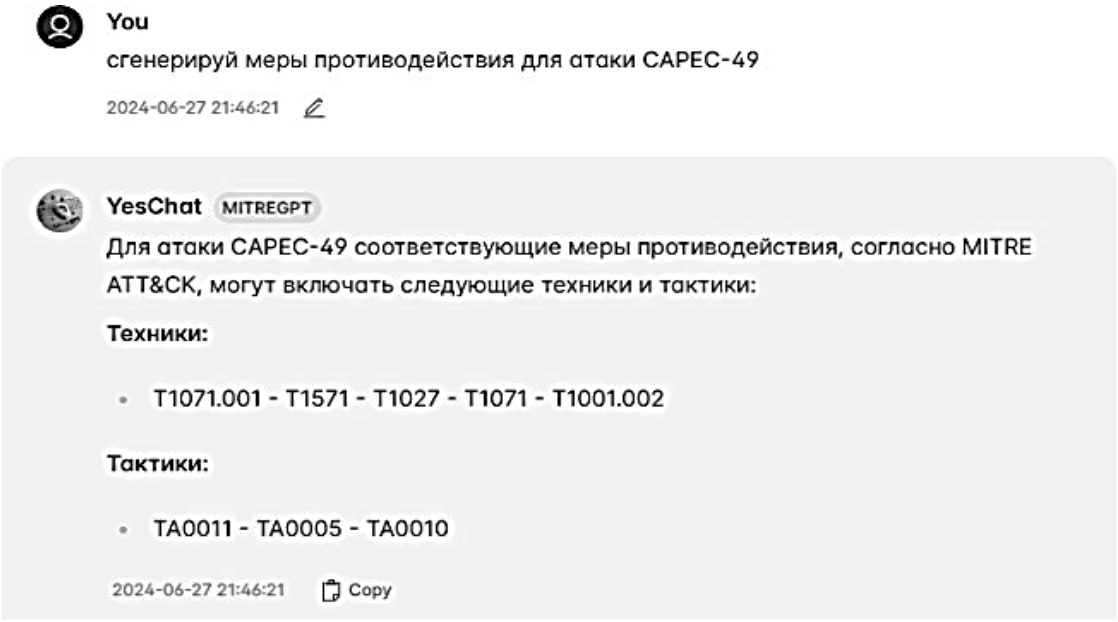

Рис. 41. Вывод мер противодействия MITREGPT для атаки CAPEC-49

Вывод довольно неудобен для пользователя, так как нет четкой последовательности действий, которые он должен принимать для смягчения ущербапри реализации атаки. Для вывода более подробной информации касаемо мер противодействия зададим MITREGPT наводящий вопрос, это показано на рис. 42.

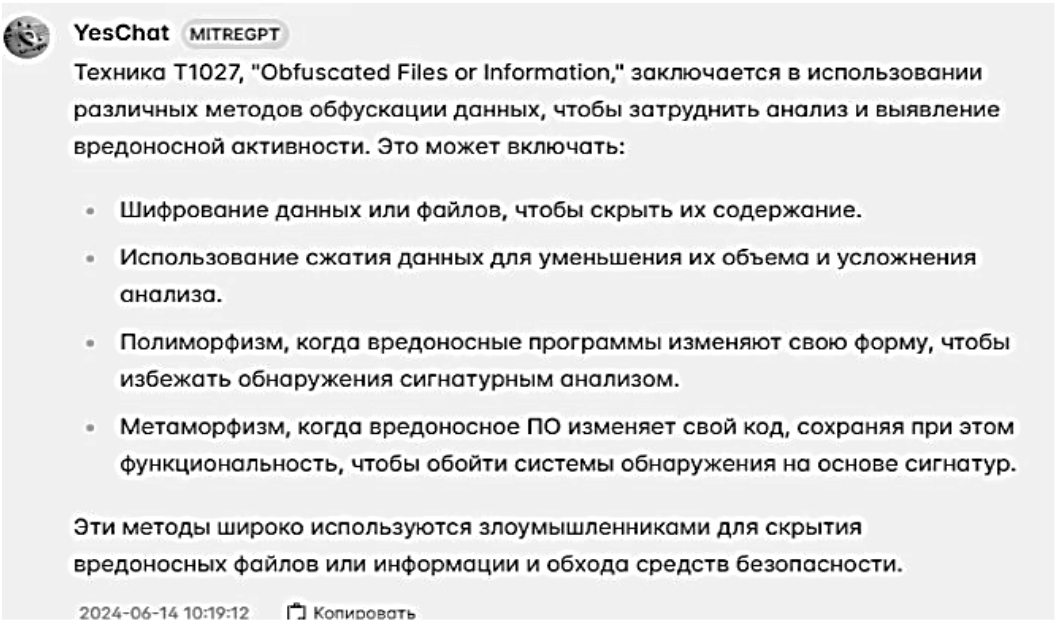

Рис. 42. Вывод интеллектуальных подсказок для мер, сгенерированных MITREGPT для атаки CAPEC-49

Ответ был довольно исчерпывающим и релевантным, однако уточнять каждую технику таким образом не очень удобно. Кроме того, вывод будет общим и конкретных мероприятий, которые пользователь должен сделать, получить с помощью данного GPT-агента не представляется возможным.

Проверим работоспособность спроектированного сервиса на входных данных, содержащих тот же класс атаки. Результаты выдачи мер противодействия наряду с интеллектуальными подсказками приведены на рис. 43-47.

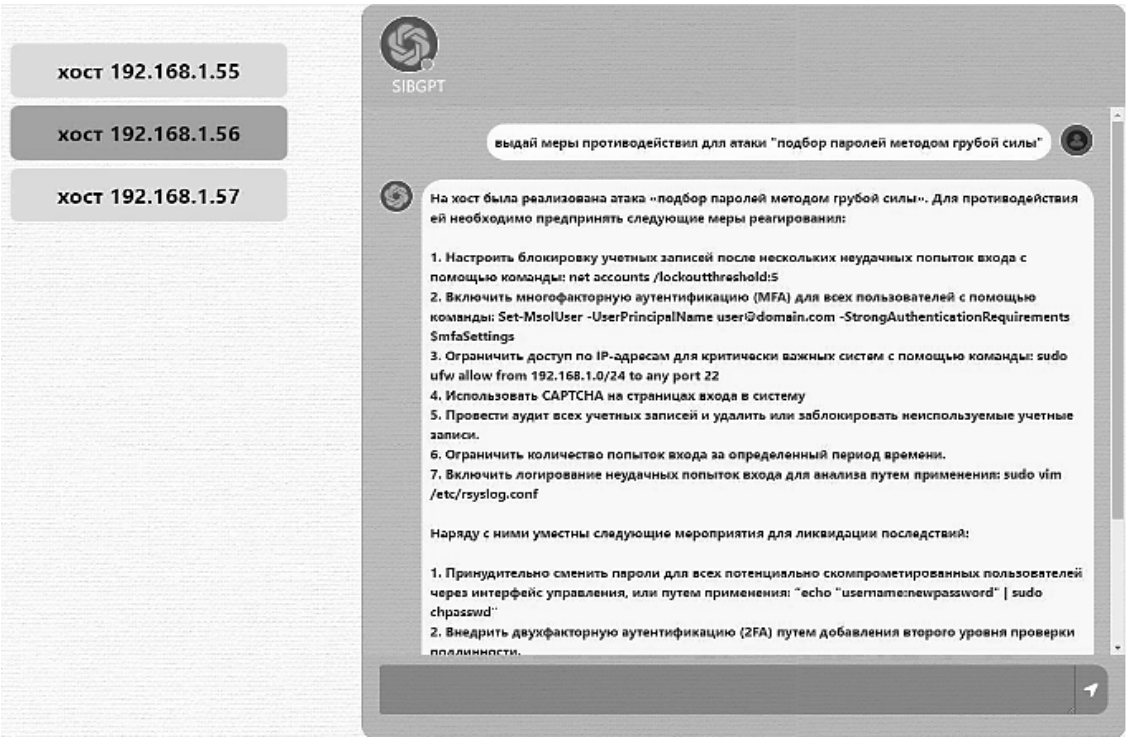

Рис. 43. Вывод мероприятий по борьбе с CAPEC-49 сервиса нейросетевой регламентации мер противодействия в режиме интеллектуальных подсказок

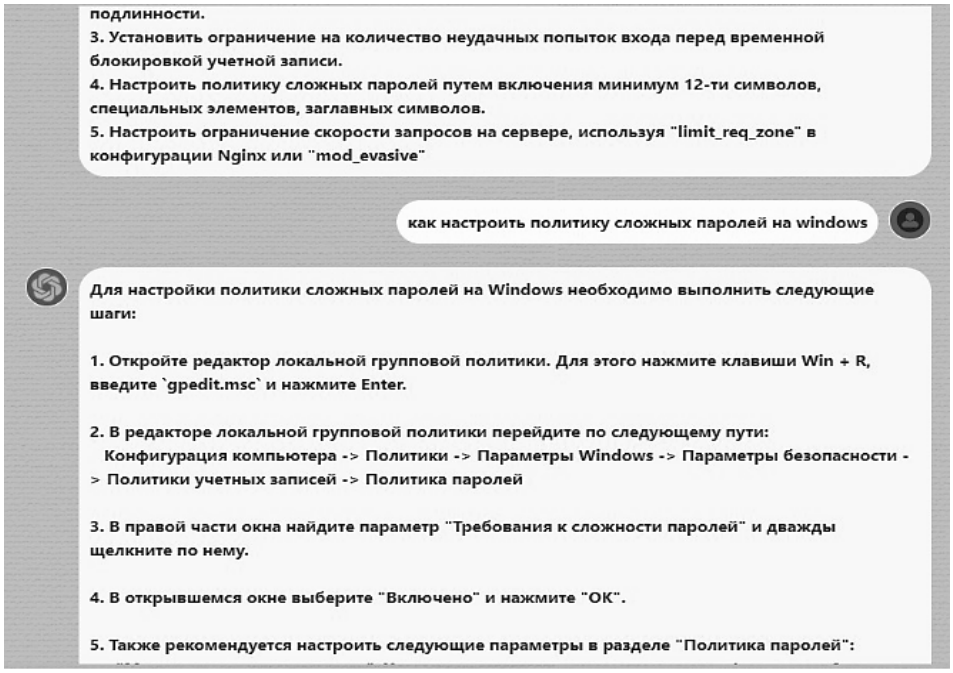

Рис. 44. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-49

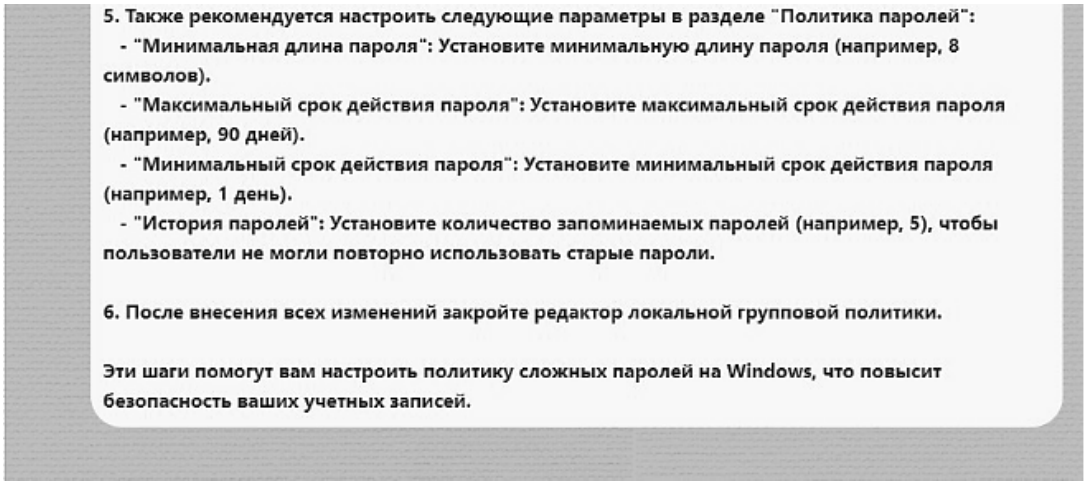

Рис. 45. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-49

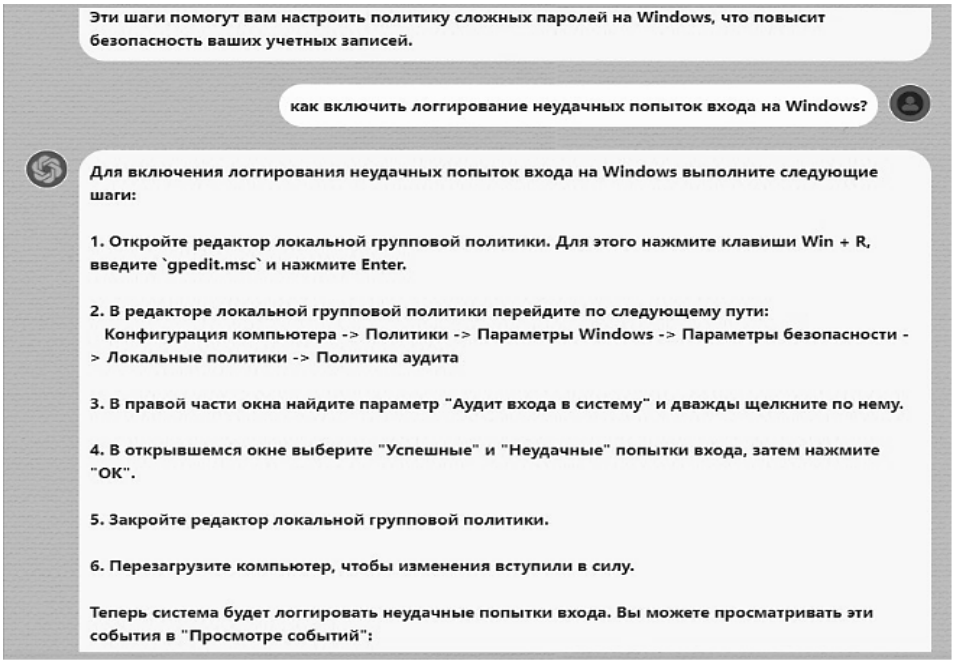

Рис. 46. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-49

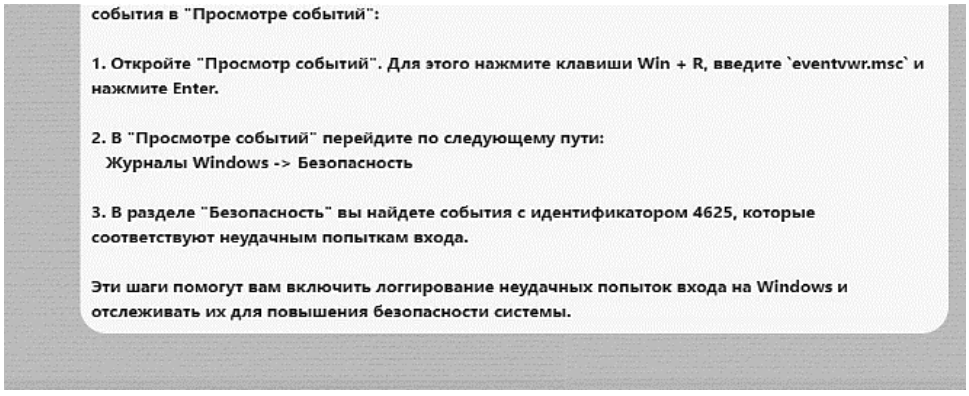

Рис. 47. Выдача интеллектуальных подсказок по сформированным мерам противодействия для CAPEC-49

Таким образом, вывод с помощью спроектированного сервиса более удобен для пользователя, так как содержит более конкретные действия, которые он должен сделать для смягчения ущерба, наряду с исчерпывающим пояснением того, как произвести непонятные администратору безопасности мероприятия. Также немаловажной особенностью является тот факт, что вывод содержит команды для различных ОС, что позволит администратору удобно применять меры противодействия, обеспечивая при этом кроссплатформенность

На последнем этапе тестирования в данном режиме произведем сравнение с выводом аналога Threat Intel Bot для атаки CAPEC-655. Результаты вывода Threat Intel Bot приведены на рис. 48, 49.

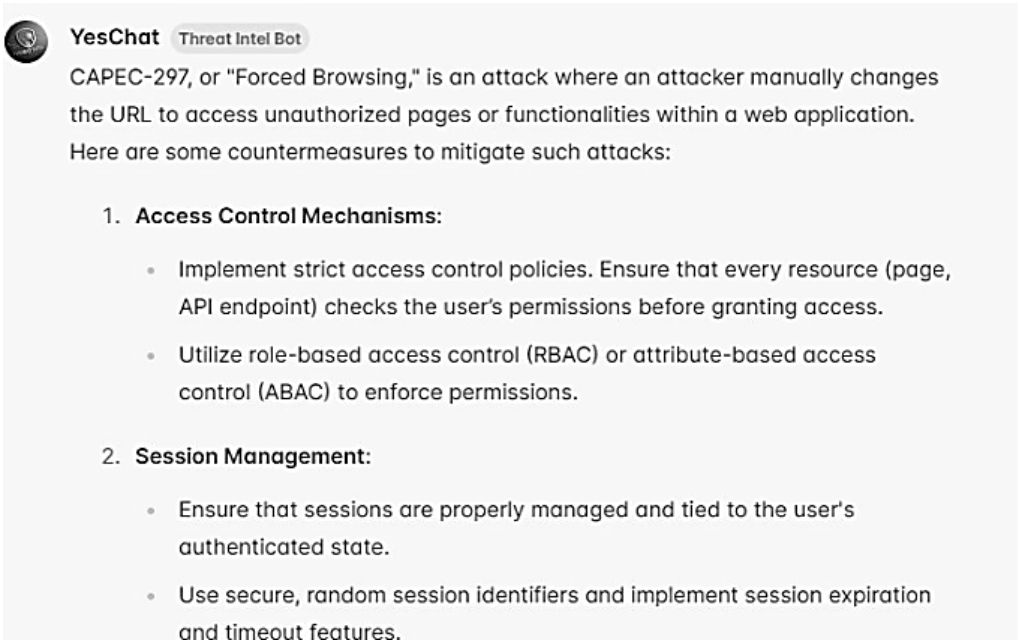

Рис. 48. Вывод мер противодействия Threat Intel Bot для атаки CAPEC-297

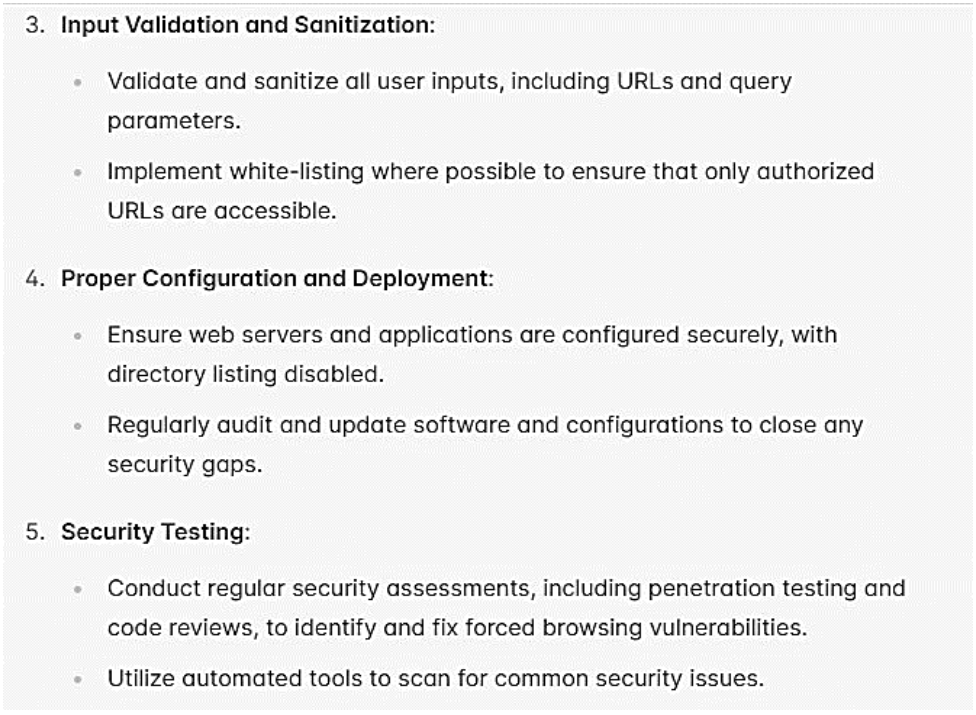

Рис. 49. Вывод мер противодействия Threat Intel Bot для атаки CAPEC-297

Вывод данного помощника похож на Soc Copilot. Однако его недостатком является невозможность вывода на русском языке. В плане конкретике мероприятийони мало чем отличаются друг от друга, ибо вывод происходит в формате ChatGPT. Еще одним серьезным недостатком является то, что неправильно определилось название атаки по ее CAPEC-идентификатору, в связи с чем регламентация мер также некорректна. Проведем тестирование спроектированного сервиса с теми же входными данными. Результаты выдачи мер противодействия наряду с интеллектуальными подсказками показаны на рис. 50-56.

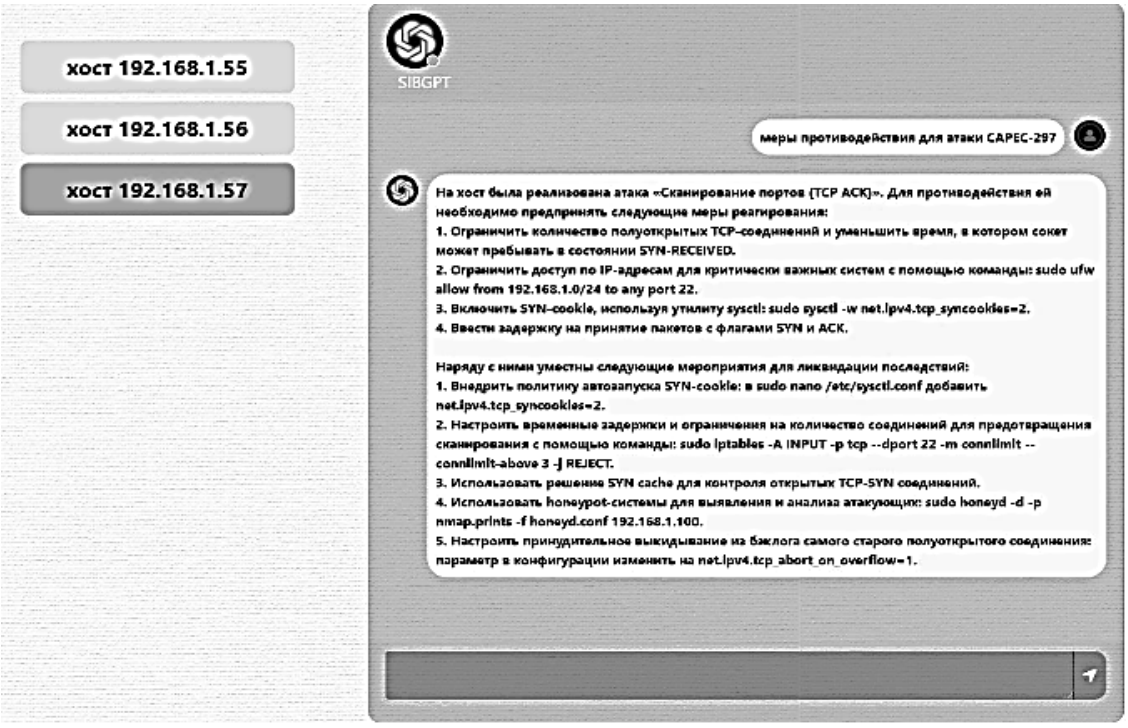

Рис. 50. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

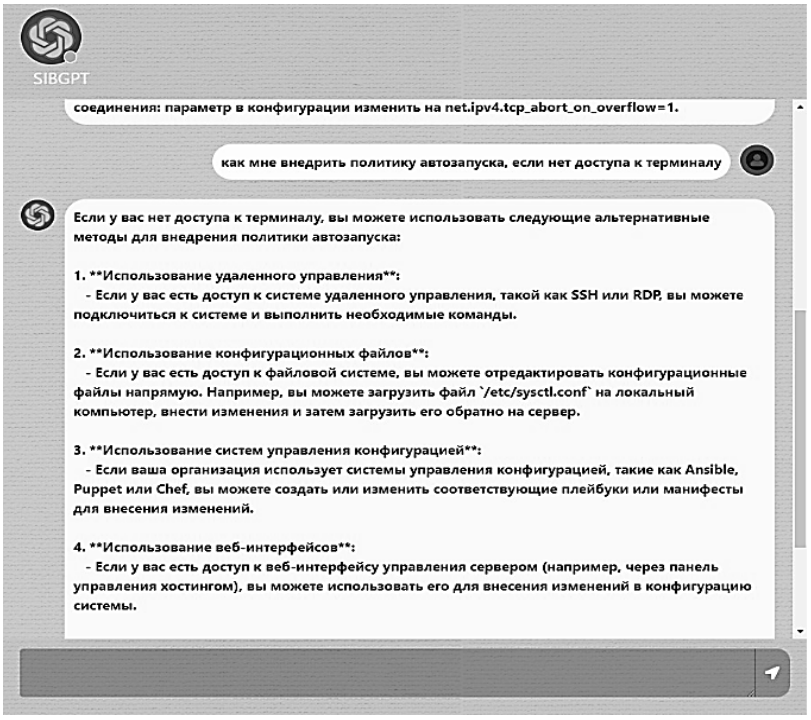

Рис. 51. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

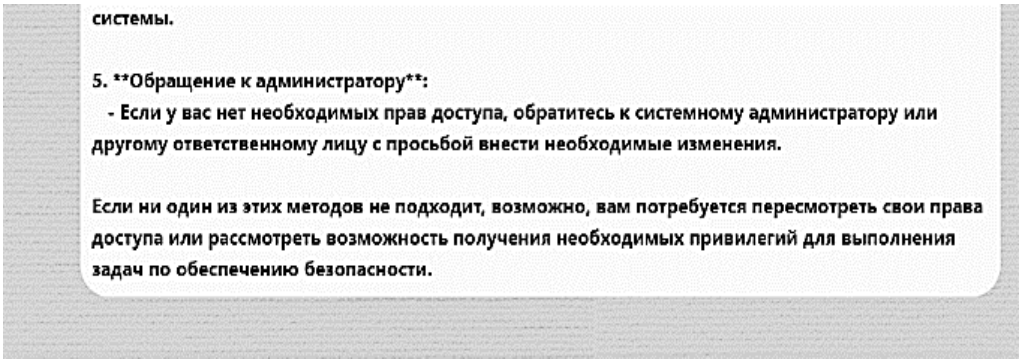

Рис. 52. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

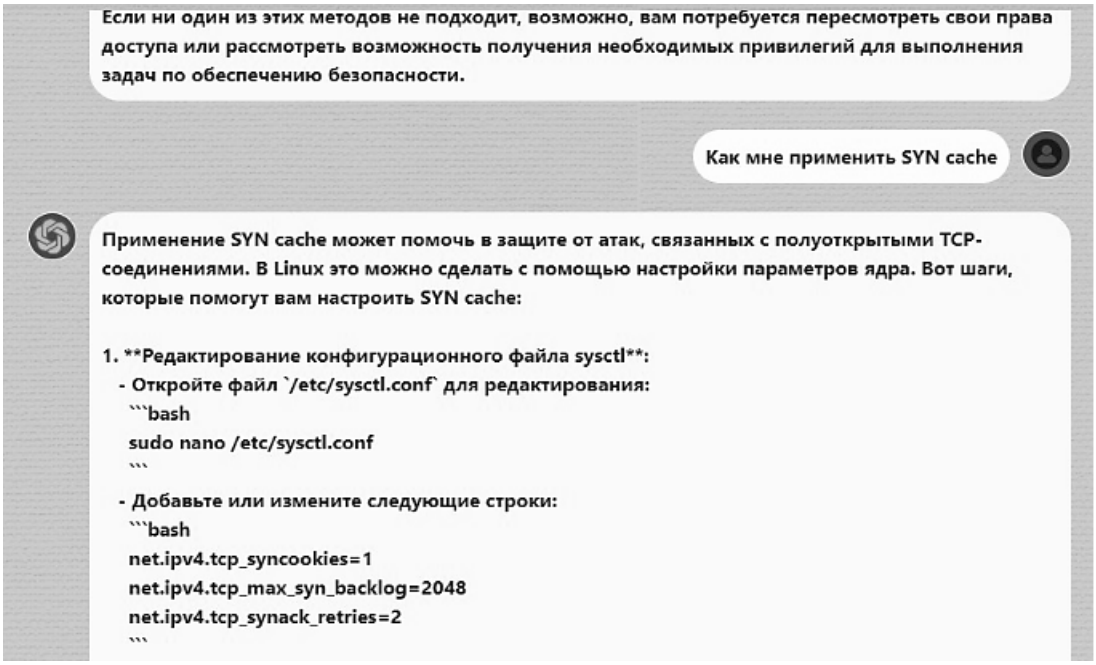

Рис. 53. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

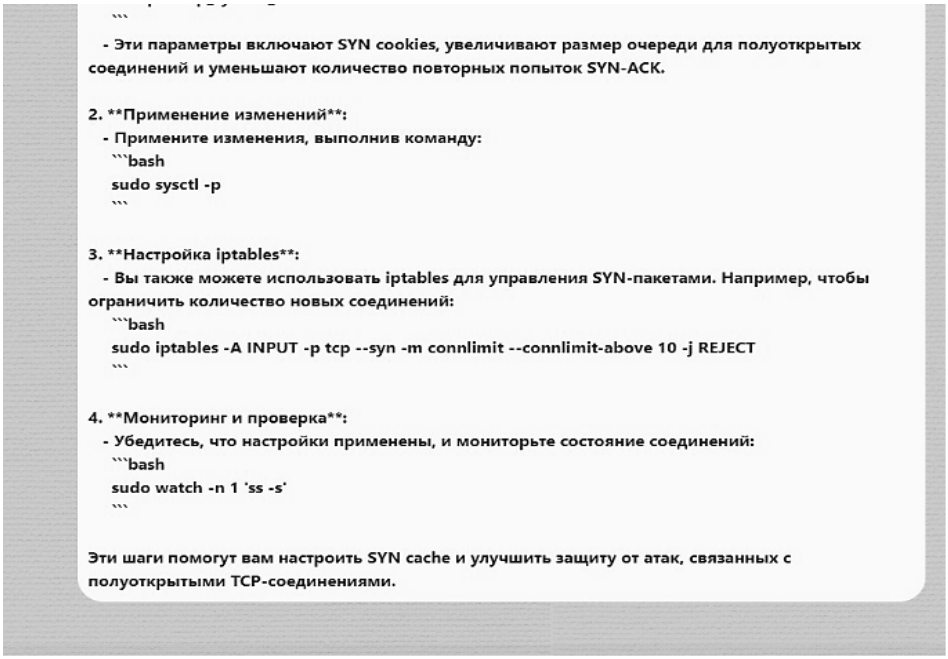

Рис. 54. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

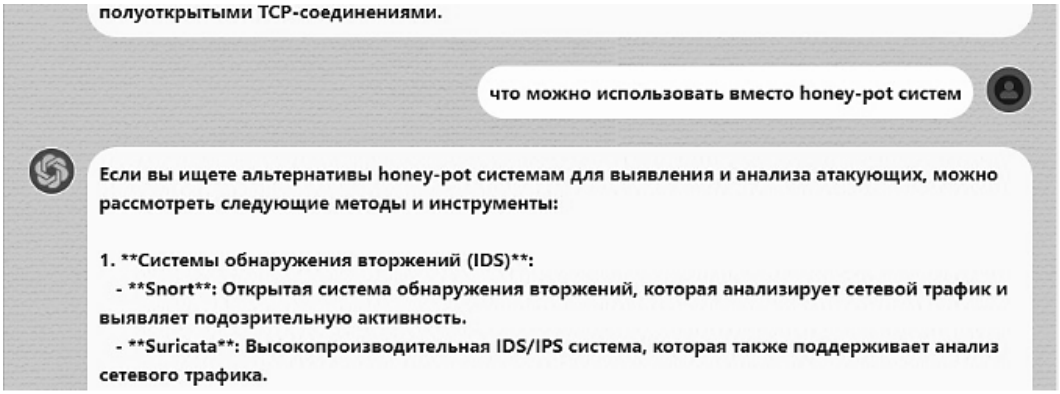

Рис. 55. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

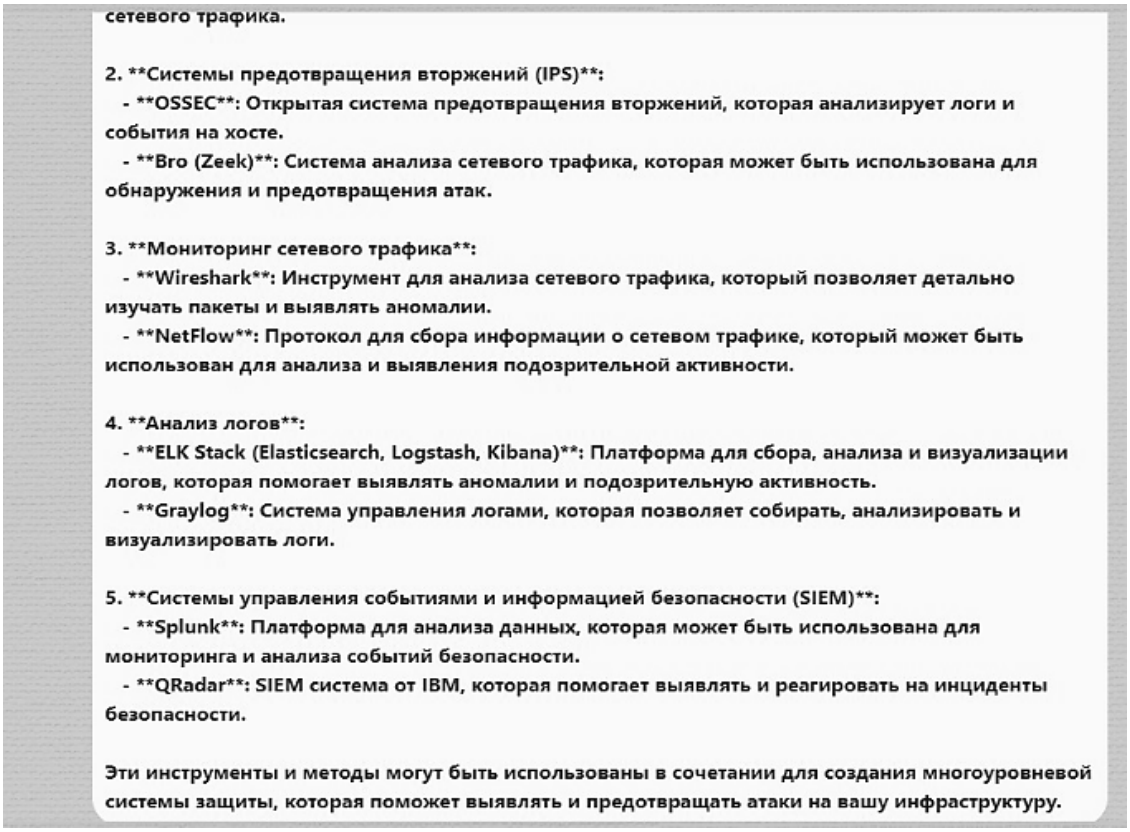

Рис. 56. Вывод интеллектуальных подсказок по сгенерированным мерам дляатаки CAPEC-297

Таким образом, спроектированный сервис регламентирует меры достаточно конкретно, разделяя вывод на меры реагирования и ликвидации последствий, не совершает ошибок в определении названия атаки по ее идентификатору согласно CAPEC, и выдает подробные интеллектуальные подсказки, позволяющие более детально раскрыть какое-либо из выданных мероприятий по смягчению ущерба, что в совокупности являются серьезными преимуществами по сравнению с результатами работы приведенных аналогов (Threat Intel Bot, MitreGPT,Soc Copilot).

Комбинированный режим

Последний режим предусматривает комбинирование первых двух. При этом приложение начинает работать по следующей схеме:

1. Приложение ожидает сигнал об атаке, приходящий из локальной сети автоматизированных информационных систем.

2. По приходу сигнала, сведения udp-пакета подвергаются парсингу. На их

основании формируется запрос в языковую модель, которая выдает в качестве ответа меры реагирования ликвидации последствий, что по сути является первым режимом работы.

3. После выдачи первичных мероприятий противодействия администратор имеет возможность их скорректировать, на данном этапе включается в работу второйрежим.

При таком способе происходит наиболее точная регламентация мер противодействия, что хорошо подходит для удовлетворения консультационных потребностей администратора безопасности при интеграции данного модуля в локальную сеть предприятия, так как меры выдаются на основании не только сценария атаки, но и специфики информационной системы и имеющегося защитного инструментария у специалиста в области ИБ. Протестируем модуль в данном режиме,показывая при этом возможность получения интеллектуальных подсказок, возьмем в качестве тестовых данных атаку CAPEC-297, реализованную на хост 192.168.1.57.

Проведем тестирование, задействуя при этом полный цикл от идентификации атаки до регламентации мер противодействия на основание входных данных. На первом этапе происходит захват траффика и его сохранение в файл .pcap при помощи функционала, предоставляемого приложением Wireshark, что показано на рис. 57.

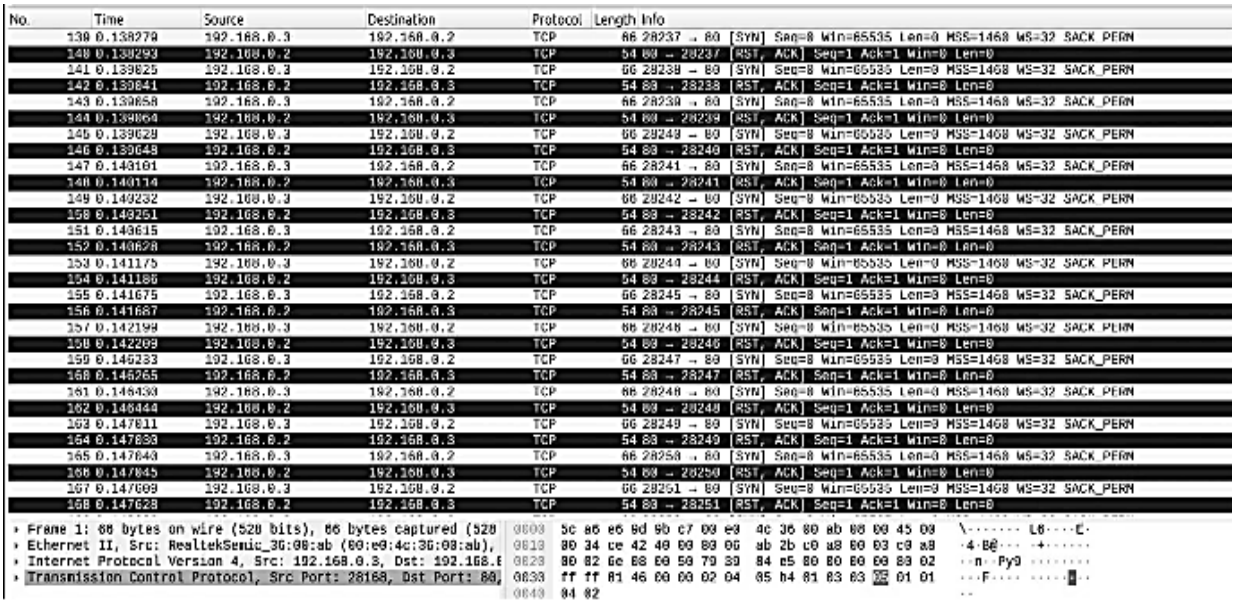

Рис. 57. Результата перехвата траффика для CAPEC-297

На следующем этапе полученный файл .pcap конвертируется в .flows для дальнейшей подаче в нейронную сеть идентификации атаки. При обнаружении вторжения нейросетевой реализацией происходит автоматическая отправка на хост администратора безопасности UDP-пакета с полезными данными о сценарии. Идентификация атаки нейронной сетью показана на рис. 58.

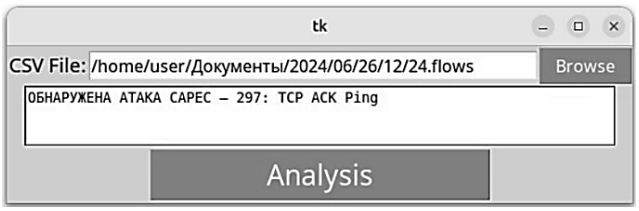

Рис. 58. Идентификация атаки CAPEC-297

Затем, при получении UDP-пакета с полезными данными о сценарии атаки происходит регламентация мер противодействия, что показано на рис. 59.

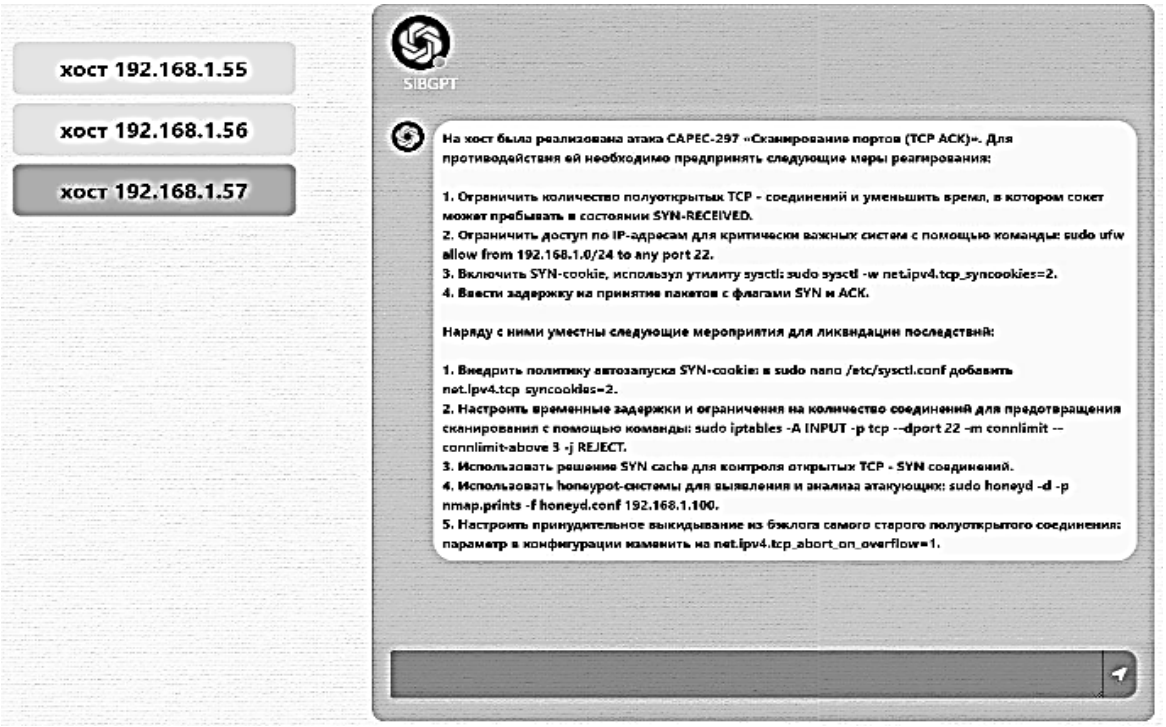

Рис. 59. Выдача мер противодействия для CAPEC-297 в комбинированном режиме

Исходя из рис. 59, можно сделать вывод, о том, что выдача мер в данном режиме совпадает с аналогичными результатами в режиме интеллектуальных подсказок, когда пользователь напрямую задавал вопрос о необходимости получения защитных мероприятий для CAPEC-297, что свидетельствует о правильности работы нейросетевой реализации. Для проверки корректности вывода интеллектуальных подсказок зададим языковой модели два вопроса по поводу конкретизации мер противодействия. Результаты ответа нейросети на поставленные вопросы показаны на рис. 60-61.

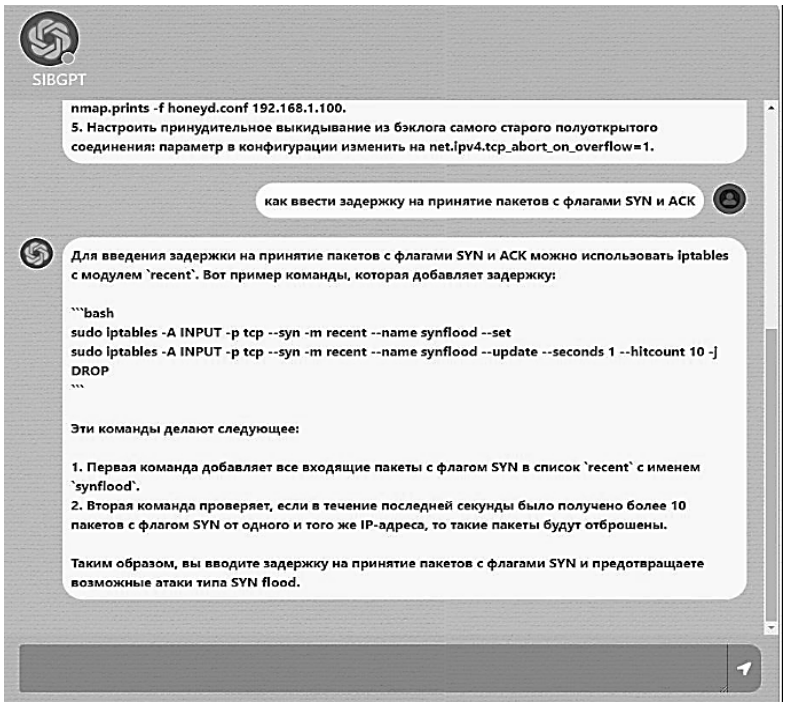

Рис. 60. Вывод интеллектуальных подсказок по сгенерированным мерам для атаки CAPEC-297 в комбинированном режиме

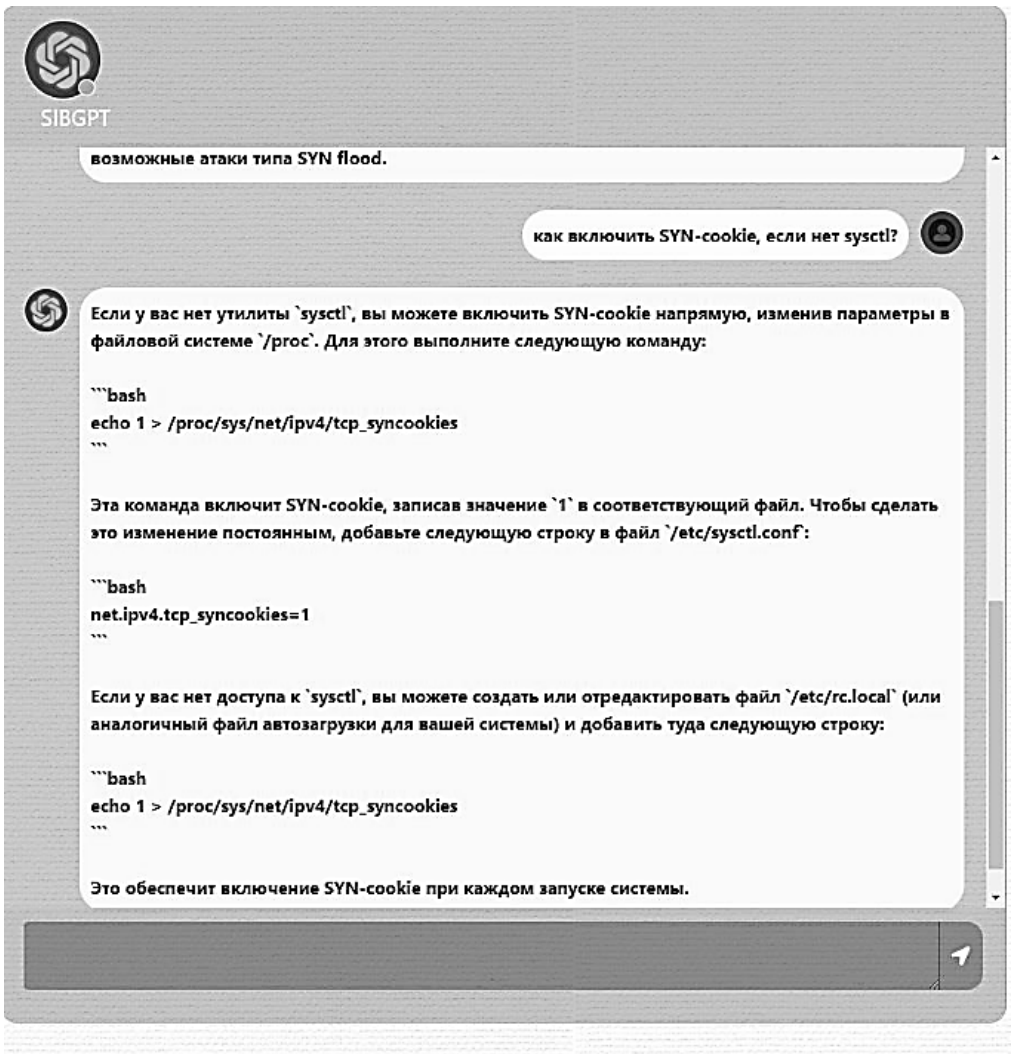

Рис. 61. Вывод интеллектуальных подсказок по сгенерированным мерам для атаки CAPEC-297 в комбинированном режиме

Исходя из рис. 60-61 можно сделать вывод о том, что меры расписаны довольно подробно, что по качеству вывода соответствует режиму интеллектуальных подсказок. Таким образом, данный режим впитал в себя преимущества двух предыдущих, что обеспечивает его удобство использования веб-приложения при интеграции системы идентификации сетевых вторжений и регламентации мер противодействия в локальную сеть предприятия.

Заключение

Была получена программная реализация разработанных методик и алгоритмов в виде нейросетевогомодуля, в отношение которого осуществлено машинное обучение агрегированными знаниями, наряду с тестированием его работы применительно к различным видам сетевых вторжений, которые обладают следующими новизной, практической ценностью и теоретической значимостью:

− программное обеспечение в отличие от аналогов актуализирует методологию построения приложений с использованием нейросетевых реализаций, покрывая при этом требования по обеспечению изолированности и возможности развертывания в рамках инфраструктуры локальной сети. При этом с помощью данного модуля впервые получены интеллектуальные подсказки по противодействию сетевым вторжениям, включающие себя меры как реагирования, так и ликвидации последствий, а также их возможные корректировки с учетом проведенного машинного обучения на базе знаний, сформированной благодаря агрегации открытых источников.

− программное обеспечение открывает возможности эффективной автоматизации процесса регламентации мер противодействия сетевым вторжениям, а также их корректировки с учетом специфики инфраструктуры, в рамках которой развернут модуль, и запросов администратора безопасности. При этом примеры регламентации мер противодействия, взятые из демонстрационных материалов проектируемого модуля, с успехом могут быть применены на практике в локальной сети, подверженной сетевым вторжениям.

− программное обеспечение, полученное в результате проектирования, может быть доведено до автоматизации регулярного дополнительного машинное обучение, расширяя при этом свой функционал под выполнение новых задач (атаки с помощью нейросетей). Причем формат демонстрационных материалов, полученных в результате запуска модуля регламентации мер противодействия, включающий в себя меры реагирования и ликвидации последствий, может быть использован при формировании корпоративных политик по обеспечению сетевой информационной безопасности.

Список литературы

1. LM Studio Documentation // Lm Studio URL: https://lmstudio.ai/docs/welcome (дата обращения: 06.06.2024).

2. The name game: Naming network interfaces in Linux // RedHat URL: https://www.redhat.com/sysadmin/network-interfaces-linux (дата обращения: 06.06.2024).

3. Что такое сетевые интерфейсы и как они устроены // Академия Selectel URL: https://selectel.ru/blog/network-interfaces/ (дата обращения: 06.06.2024).

4. Netifaces 0.11.0 // PyPI URL: https://pypi.org/project/netifaces/ (дата обращения: 06.06.2024).

5. paddress — IPv4/IPv6 manipulation library // Python docs URL: https://docs.python.org/3/library/ipaddress.html (дата обращения: 06.06.2024).

6. ping3 4.0.8 // PyPI URL: https://pypi.org/project/ping3/ (дата обращения: 06.06.2024).

7. PyInstaller Manual // PyInstaller URL: https://pyinstaller.org/en/stable/index.html (дата обращения: 06.06.2024).

8. OpenAI Python API – Complete Guide // GeeksforGeeks URL: https://www.geeksforgeeks.org/openai-python-api/ (дата обращения: 06.06.2024) –

9. Документация pandas // Pandas URL: http://pandas.geekwriter.ru/ (дата обращения: 06.06.2024).

10. Level up with the largest AI & ML community // Kaggle URL: https://www.kaggle.com/ (дата обращения: 06.06.2024).

11. Google Colaboratory // Google URL: https://colab.google/ (дата обращения: 06.06.2024).

12. Os // Python Docs URL: https://treasuremaster.gitbook.io/python-docs/moduli- standartnoi-biblioteki-1/obshie-sluzhby-operacionnoi-sistemy/os (дата обращения: 06.06.2024).

13. Logging HOWTO // Docs Python URL: https://docs.python.org/3/howto/logging.html (дата обращения: 06.06.2024).

14. Pathlib — Object-oriented filesystem paths // Docs Python URL: https://docs.python.org/3/library/pathlib.html (дата обращения: 06.06.2024).

15. Dataset: databases for lazy people // Dataset URL: https://dataset.readthedocs.io/en/latest/ (дата обращения: 06.06.2024).

16. PyTorch documentation // PyTorch URL: https://pytorch.org/docs/stable/index.html (дата обращения: 06.06.2024).

17. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning. // GitHub URL: https://github.com/huggingface/peft (дата обращения: 06.06.2024).

18. Введение в библиотеку Transformers и платформу Hugging Face // Хабр URL: https://habr.com/ru/articles/704592/ (дата обращения: 06.06.2024).

19. Mistral 7B - лучше в русском чем saiga2_7b // GitHub URL: https://github.com/IlyaGusev/rulm/issues/30 (дата обращения: 06.06.2024).

20. Language modeling and instruction tuning for Russian // GitHub URL: https://github.com/IlyaGusev/rulm (дата обращения: 06.06.2024).

21. LLM inference in C/C++ // GitHub URL: https://github.com/ggerganov/llama.cpp (дата обращения: 06.06.2024).

22. Machine Learning Life Cycle: What Are Its Key Stages? // Plat URL: https://plat.ai/blog/stages-of-machine-learning-lifecycle/ (дата обращения: 06.06.2024).

23. User's Guide // Flask URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 06.06.2024).

24. Django documentation // Django URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 06.06.2024).

25. Threading — Параллелизм на основе потоков // Digitology.tech URL: https://digitology.tech/docs/python_3/library/threading.html (дата обращения:06.06.2024).

26. Requests: HTTP for Humans // Requests URL: https://requests.readthedocs.io/en/latest/index.html (дата обращения: 06.06.2024).

27. Структура и эскиз приложения Flask // PythonRu URL: https://pythonru.com/uroki/19-struktura-i-jeskiz-prilozhenija-flask (дата обращения: 06.06.2024).

28. Проектирование простых приложений в Flask // Хабр URL: https://habr.com/ru/articles/275099/ (дата обращения: 06.06.2024).

29. Справочник HTML // HTML и CSS URL: https://htcdev.ru/html/ (дата обращения: 06.06.2024).

30. jQuery API Documentation // jQuery URL: https://api.jquery.com/ (дата обращения: 06.06.2024).

31. Javascript Documentation // Javascript URL: https://js- documentation.netlify.app/ (дата обращения: 06.06.2024).

32. jQuery.ajax() // jQuery API Documentation URL: https://api.jquery.com/jQuery.ajax/ (дата обращения: 06.06.2024).

33. HTTP-запросы: структура, методы, строка статуса и коды состояния // Академия Selectel URL: https://selectel.ru/blog/http-request/ (дата обращения: 06.06.2024).

Финансовый университет при Правительстве Российской Федерации
Financial University under the Government of the Russian Federation

Воронежский государственный технический университет
Voronezh State Technical University

Поступила в редакцию 17.07.2024

Информация об авторах

Остапенко Григорий Александрович – д-р техн. наук, профессор, Финансовый университет при Правительстве Российской Федерации, e-mail: ost@fa.ru

Васильченко Алексей Павлович – аспирант, Финансовый университет при Правительстве Российской Федерации, e-mail: rainichek@yandex.ru

Остапенко Александр Алексеевич – аспирант, Воронежский государственный технический университет, e-mail: alexostap123@gmail.com

Ноздрюхин Александр Александрович – студент, Воронежский государственный технический университет, e-mail: sfrvvv@yandex.ru

Остапенко Александр Григорьевич – д-р техн. наук, профессор, заведующий кафедрой, Воронежский государственный технический университет, e-mail: alexanderostapenkoias@gmail.com

Батаронов Игорь Леонидович – д-р физ.-мат. наук, профессор, заведующий кафедрой, Воронежский государственный технический университет, e-mail: alexanderostapenkoias@gmail.com

Печкин Дмитрий Сергеевич – студент, Воронежский государственный технический университет, e-mail: alexanderostapenkoias@gmail.com

NEURAL NETWORK SERVICE FOR REGULATING MEASURES TO COUNTER CYBER ATTACKS (PART II)

G.A. Ostapenko, A.P. Vasilchenko, A.A. Ostapenko, A.A. Nozdryukhin, A.G. Ostapenko, I.L. Bataronov, D.S. Pechkin

The neural network implementation of the automated confrontation service is considered in terms of regulating measures to counter computer attacks. In this regard, a software implementation of the neural network regulation service for countering cyber attacks is proposed, including scripts for determining IP addresses, interacting with a language model, a machine learning module, scripts for preparing a knowledge base and implementing machine learning, a backend-service, and a frontend-service. Examples of practical use of the service in intrusion detection modes, intelligent prompts, and combined mode are demonstrated.

Keywords: service, neural network implementation of service, cyberattack, script, web application, backend-service, frontend-service.

Submited 17.07.2024

Information about authors

Grigoriy A. Ostapenko – Dr. Sc. (Technical), Professor, Financial University under the Government of the Russian Federation, e-mail: ost@fa.ru

Alexsey P. Vasilchenko – graduate student, Financial University under the Government of the Russian Federation, e-mail: rainichek@yandex.ru

Aleksandr A. Ostapenko – graduate student, Voronezh State Technical University, e-mail: alexostap123@gmail.com

Aleksandr A. Nozdriuhin – student, Voronezh State Technical University, e-mail: sfrvvv@yandex.ru

Aleksandr G. Ostapenko – Dr. Sc. (Technical), Professor, Head of the Department, Voronezh State Technical University, e-mail: alexanderostapenkoias@gmail.com

Igor L. Bataronov – Dr. Sc. (Physical and Mathematical), Professor, Head of the Department, Voronezh State Technical University, e-mail: alexanderostapenkoias@gmail.com

Dmitriy S. Pechkin – student, Voronezh State Technical University, e-mail: alexanderostapenkoias@gmail.com