



МИНОБРАЗОВАНИЯ РОССИИ  
федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Национальный исследовательский университет «МЭИ»

Институт

ИВТИ

Кафедра

УИТ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
(БАКАЛАВРСКАЯ РАБОТА)

Направление

27.03.04 Управление в технических системах

(код и наименование)

Образовательная  
программа

Управление и информатика в технических  
системах

Форма обучения

очная

(очная/очно-заочная/заочная)

Тема:

Обнаружение DDoS-атак в HTTPS-трафике по журналам web-сервера

Студент

A-01-18

группа

подпись

Соловьев Д.Р.

фамилия и инициалы

Руководитель  
ВКР

уч. степень

старший  
преподаватель

должность

подпись

Козлюк Д.А.

фамилия и инициалы

Консультант

уч. степень

должность

подпись

фамилия и инициалы

Внешний  
консультант

уч. степень

должность

подпись

фамилия и инициалы

организация

«Работа допущена к защите»

Заведующий  
кафедрой

Д.Т.Н.

уч. степень

доцент

звание

подпись

Бобряков А.В.

фамилия и инициалы

Дата

Москва, 2022





МИНОБРАЗОВАНИЯ РОССИИ  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Национальный исследовательский университет «МЭИ»

Институт  
Кафедра

ИВТИ  
УИТ

ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ  
(БАКАЛАВРСКУЮ РАБОТУ)

Направление

27.03.04 Управление в технических системах  
(код и наименование)

Образовательная  
программа

Управление и информатика в технических  
системах

Форма обучения

очная

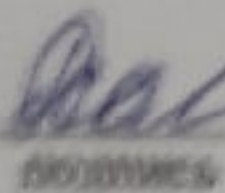
(очная/очно-заочная/заочная)

Тема:

Обнаружение DDoS-атак в HTTPS-трафике по журналам web-сервера

Студент

А-01-18  
группа

  
подпись

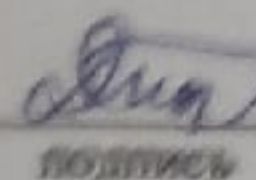
Соловьев Д.Р.  
фамилия и инициалы

Руководитель  
ВКР

уч. степень

старший  
преподаватель

должность

  
подпись

Козлюк Д.А.  
фамилия и инициалы

Консультант

уч. степень

должность

подпись

фамилия и инициалы

Внешний  
консультант

уч. степень

должность

подпись


фамилия и инициалы

организация

Заведующий  
кафедрой

д.т.н.  
уч. степень

доцент  
звание

  
подпись

Бобряков А.В.  
фамилия и инициалы

Место выполнения работы

ФГБОУ ВО «НИУ «МЭИ»



# СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

1. Исследование предметной области и исходных данных
2. Реализация программы
3. Тестирование программы

## ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов

40

Количество слайдов в презентации

14

## РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

### Примечания:

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

## **АННОТАЦИЯ**

Решаемая задача — выявить адреса источников DDoS-атаки, состоящей из зашифрованных HTTP-запросов (HTTPS flood). Предлагается классифицировать записи в журнале запросов (access.log) защищаемого web-сервера на принадлежащие атакующим и легитимным клиентам. Настройка классификатора производится по выборке, размеченной по информации с действующей системы защиты, работающей на сетевом и транспортном уровне. Эксперименты проводятся на данных реальных сайтов и атак.

## СОДЕРЖАНИЕ

.....	1
Аннотация.....	2
Введение.....	6
1 Исследование предметной области и исходных данных.....	9
1.1 HTTPS flood атака.....	9
1.2 Разбор средств защиты от DDoS-атак .....	9
1.3 Анализ исходных данных модели.....	10
1.3.1 Происхождение исходных данных модели .....	10
1.3.2 Разбор признаков запроса.....	10
1.3.3 Разметка данных перед обучением.....	13
1.3.4 Подготовка входных данных перед обучением .....	13
1.4 Разбор входных данных нейронной сети .....	17
1.5 Выбор метода машинного обучения для классификатора .....	17
1.6 Выбор структуры нейронной сети .....	18
1.7 Процесс обучения нейронной сети .....	20
2 Реализация программы.....	22
2.1 Обзор требований .....	22
2.2 Выбор средств .....	23
2.3 Обзор программы.....	25
3 Тестирование программы.....	27
3.1 Тестирование модуля извлечения данных .....	27
3.2 Тестирование модуля подготовки данных .....	29
3.3 Тестирование модуля обучения сети.....	30

3.4 Сквозное тестирование.....	35
3.5 Проблема введения системы в эксплуатацию .....	36
Заключение .....	38
Список использованных источников.....	39



## ВВЕДЕНИЕ

Доступность информационных систем через интернет критически важна в современной жизни [1]. Из атак на доступность распространены DDoS-атаки, заключающиеся в том, что большое количество клиентов перегружает сервер формально легитимными запросами. Из-за того, что их количество слишком велико, у сервера заканчиваются вычислительные ресурсы и он становится недоступен для пользователей. Интенсивность трафика каждого отдельного клиента может быть низкой, поэтому, чтобы отличить запросы атакующих, нужно анализировать содержимое пакетов вплоть до прикладного уровня.

Для того, чтобы не тратить ресурсы сервера на обработку трафика атаки, применяют защитные решения, на которые поступает весь входящий трафик (смесь легитимного трафика и трафика атаки), а к защищаемому серверу пропускается только легитимный, то есть защита работает подобно фильтру (рисунок 1). Он может представлять собой либо облачный сервис (cloud-based service provider, CBSP), либо программно-аппаратный комплекс для организации защиты (ПАК, on-premise)

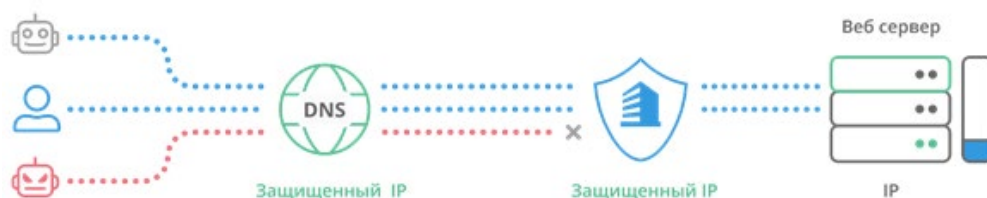


Рисунок 1 — Схема защиты от DDoS-атак [2]

Одним из актуальных типов DDoS-атак является HTTPS flood [3]. В этих атаках трафик прикладного уровня зашифрован. Фильтру доступны только заголовки пакетов, чтобы блокировать трафик с адресов атакующих, но он не может обнаружить атаку, так как ему недоступно для анализа содержимое пакета. Данные будут расшифрованы только на сервере, но обработка запросов, атакующих перегружает его.

Система защиты нуждается в информации о запросах и их параметрах. Один из способов — передать защитной системе ключи шифрования

защищаемого сервера. Однако расшифровка трафика на промежуточных узлах может быть нежелательна. В случае CBSP это означает раскрытие трафика легитимных клиентов третьей стороне, что может быть незаконно или нести риски для защищаемого бизнеса. Расшифровка требует больших вычислительных мощностей, что означает рост себестоимости для CBSP или рост цены для on-premise решения. По этим причинам и CBSP, и on-premise решения предлагают схемы защиты, когда сервер-жертва передает записи из журнала web-сервера (access.log) защитной системе, которая анализирует их и фильтрует входящий трафик [4, 5].

Проприетарные CBSP и on-premise решения не раскрывают полностью используемых алгоритмов, так как это является основой бизнеса их разработчиков. Однако описаны случаи, когда для анализа access.log успешно применялись искусственные нейронные сети [6]. Это целесообразно, поскольку записи обладают большим количеством признаков, которые в случае каждого конкретного сайта могут быть связаны различным образом.

Задача состоит в классификации запросов на запросы атаки и легитимные. Обучение классификатора требует размеченной выборки. Образцы легитимных запросов легко могут быть взяты администратором защищаемого сервера, когда атаки не происходит. Однако разделить запросы, полученные во время атаки, на легитимные и нет, администратор не может. Мы предлагаем воспользоваться данными, которые имеются у защитной системы. Некоторые клиенты-участники атаки могут быть заблокированы на основе объема трафика, частоты подключений и иных признаков, не требующих расшифровки запросов. Сопоставив данные о блокировках (время и IP-адрес) и записи access.log, можно выполнить разметку.

Мы будем работать с системой продукта «BIFIT MITIGATOR». Она блокирует IP-адреса, которые атакуют и заносит их в журнал блокировок, благодаря этому сервер будет защищен пока модель не обучится и признаки будут размечены для обучения. Сначала с помощью мы него разметим наши



данные и обучим модель под работу на конкретном сервере. После модель сможет работать параллельно с сетевым фильтром, анализируя записи access.log и указывая защитной системе блокировать адреса, с которых происходит атака.

В первой главе рассматриваются исходные данные и предлагаются методы их предварительной обработки. Во второй главе выбираются средства для реализации предложенных методов. В третьей главе описан процесс тестирования системы и его результат.

# **1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ИСХОДНЫХ ДАННЫХ**

## **1.1 HTTPS flood атака**

HTTP flood – это атака на уровне приложения(L7-атаки). Нападающий использует большое количество HTTP запросов, на обработку которых сервер тратит вычислительные ресурсы. Из-за того, что таких заголовков слишком много сервер перегружается и это приводит к отказу системы.

При HTTPS flood атаке данные поступают на сервер в зашифрованном виде, что служит только на пользу атакующему, так как они не могут быть расшифрованы до поступления на сервер, что усложняет работу средств защиты.

## **1.2 Разбор средств защиты от DDoS-атак**

Защита представляет собой последовательность контрмер, каждая из которых может быть включена или выключена. Включенные контрмеры влияют на то, при каком поведении клиентов их IP-адреса будут заблокированы и сообщены анализатору для разметки выборки. Релевантные по порядку:

- TBL (temporary black list): в контрмере заносится адрес с интервалом блокировки, в течение заданного интервала трафик с адреса блокируется.
- CRB (connection rate blocking): если с определенного адреса количество пакетов TCP с флагом SYN, означающих попытку начала соединения, выше порога, адрес отправителя заносится в TBL.
- LCON (connection limiter): похож на CRB, но отслеживает не скорость установления новых соединений, а количество существующих соединений, не позволяет создать больше указанного.
- ATLS (active TLS protection): анализирует параметры TLS (протокола в стеке HTTPS, отвечающего за шифрование), у некоторых ботов они отличаются, поэтому их адреса можно заносить в TBL до того, как они смогут начать атаку. Web-сервер этих соединений не видит.



- SORB (source rate blocking): если с определенного адреса трафик (пакетов/бит в секунду = pps/bps) больше порога, адрес отправителя заносится в TBL, в некоторых случаях скорость ограничивается.

### 1.3 Анализ исходных данных модели

#### 1.3.1 Происхождение исходных данных модели

Входные данные, на основе которых будет строиться работа, берутся из двух источников. Первый это журнал запросов, поступающих на сервер (access.log), а второй это журнал результатов работы сетевого фильтра во время атаки на сервер, в который записываются все IP-адреса, заблокированные на некоторое время.

#### 1.3.2 Разбор признаков запроса

Журнал запросов представляет собой документ, каждая строка которого содержит информацию об одном запросе. Там может содержаться большое количество различных параметров в зависимости от того, как ее настроил сетевой администратор.

Пример записи в access.log (для удобства разбита на строки):

```
185.144.28.57 - - [03/Nov/2021 06:27:27 +0300]  
"GET / HTTP/1.0" 200 28941 "-"  
"Mozilla/5.0 (CHE; +https://hosters.ru/uptime-checker.html) "
```

Запись включает:

- 185.144.28.57 — IP-адрес клиента;
- 03/Nov/2021 06:27:27 +0300 — время, в которое поступил запрос, с точностью до секунд;
- GET — метод (глагол) HTTP;
- / — идентификатор запрошенного ресурса (URI);
- HTTP/1.0 — версия протокола;
- 200 — код статуса, с которым завершен запрос [7];
- 28941 — размер ответа на запрос в байтах;

— "Mozilla/5.0 (CNE; +https://hosters.ru/uptime-checker.html)" — HTTP-заголовок User-Agent, определяющий программу или библиотеку, которая используется клиентом.

По IP-адресу можно с помощью специальных баз данных GeoIP приблизительно определить местоположение клиента. Если место нехарактерно для посетителей сайта, это может свидетельствовать об атаке. Например, большинство российских сайтов редко посещают пользователи из Китая.

Поле времени важно в сочетании с другими параметрами, так как те или иные запросы могут быть характерны в разное время суток. Например, пользователи с IP-адресами московских провайдеров могут посещать сайт преимущественно тогда, когда в Москве рабочий день.

Сочетание метода HTTP и URI может быть характерным или нет. Например, может быть нетипичным использование метода GET и URI, который в норме принимает данные с методом POST. Нехарактерным может быть и URI в отдельности.

Версия HTTP может быть нехарактерна для сайта в целом либо в сочетании с конкретным URI или значением заголовка User-Agent.

По коду состояния можно судить об успешности выполнения запроса. Например, при атаке может быть сгенерирована ссылка, которая похожа на реально существующие на сайте, но некорректная, поэтому код статуса будет ошибочный (404), что отличает запрос атаки от легитимных запросов с похожими URI.

Размер ответа может свидетельствовать об атаке, если его рассматривать в совокупности с методом запроса и признаками, полученными при разборе ссылки, так как для каждого метода и некоторых ссылок характерен определенный размер ответа, который возвращается клиенту.

Поле содержащее заголовок HTTP-запроса агента пользователя является необходимым, так как по нему видно с помощью какой программы был сделан запрос на сервер. Заметим, что эту информацию сообщает клиент, поэтому



атакующий волен указать, что запросы от программы-бота исходит от браузера. Однако ботнеты не всегда состоят из специальных ботов, способных на подделку User-Agent, в этом случае атакующий заголовки контролировать не может.

Все перечисленные поля по одиночке или в совокупности будут использоваться для обучения. Но перед этим из поля содержащего заголовок HTTP-запроса агента пользователя, IP-адреса, метода HTTP и URI необходимо извлечь признаки:

1. Из поля `\"%User-agent{i\"` извлечем 4 категориальных признака:
  - общий маркер, который говорит, что браузер совместим с Mozilla и является общим практически для каждого браузера сегодня
  - операционную систему устройства, с которого произошел запрос
  - модель устройства
  - программу, обычного браузера, которая использовалась для совершения запроса.
2. Используя поле `%h` обогатим наши данные, получив два категориальных признака:
  - Страну, из которой пришел запрос
  - Континент, с которого пришел запрос
3. Поле `%r` разберем на два категориальных признака:
  - метод запроса
  - версию протокола

Также разберем ссылку из данного поля, на слова, получив еще один признак. Также будем смотреть на наличие цифр в ссылке и в случае их наличия будем это помечать 1 в ином случае 0. Это будет последним признаком, полученным из данного поля.

### 1.3.3 Разметка данных перед обучением

Журнал работы сетевого фильтра представлен в виде CSV таблицы со следующими полями: `instance_id`, `instance_name`, `created_at`, `action`, `ip`, `source`, `country`, `city`, `as_number`, `as_name`. Нас интересуют 3 поля `created_at`, `action`, `ip` и два признака, которые мы получили из `access.log` по ним мы можем разметить наши данные.

Так сетевой фильтр сначала при необходимости блокирует `ip`, а потом его разблокирует, то можно составить временные интервалы, в которые адрес пользователя был заблокирован. Для этого можно посмотреть на запись в поле `action`, оно может содержать два значения `added` и `remove` первое обозначает, что `ip` был заблокирован, а второе что его разблокировали. Соответственно для формирования временного интервала сначала смотрим на статус, и если он соответствует блокировке, то используем поле времени, как нижнее значение временного диапазона и наоборот с верхним. Также при его формировании нужно учитывать, что сетевой фильтр не всегда реагирует сразу, поэтому зная примерное время нахождения одного `ip` адреса в блокировке, можно вручную рассчитать недостающие границы временного диапазона.

Дальше мы сверяем время поступления запроса на сервер от конкретного `ip` адреса с временными интервалами, и в случае, если он попадает в него, то помечаем данный запрос как атакующий.

### 1.3.4 Подготовка входных данных перед обучением

Данные, полученные из журналов, были записаны в таблицу формата `csv`. Некоторые из них требуется подготовить перед использованием их в обучении. Всего у нас 14 столбцов, которых необходимо проанализировать и при необходимости обработать.

```
'country', 'continent', 'timezone', 'datetime',  
'method', 'url_path', 'number_availability', 'http',  
'status_code', 'size_object', 'exit_system', 'os',  
'browser', 'device'.
```



Столбцы: 'country', 'continent', 'timezone', 'method', 'http', 'exit\_system', 'os', 'browser', 'device' являются категориальными и представляют из себя слова, модель не может работать с признаками, представленными в таком формате. Поэтому их необходимо изменить, закодировав их в числа. Двумя более популярными методами являются Ordinal Encoding и One-Hot Encoding. Так как в данных не существует порядковых отношений, первый метод может не подойти, так как может ввести модель в заблуждение, чтобы преобразовать каждую строку столбца в вектор чисел состоящий из 0 и 1. Длина вектора равняется количеству уникальных значений в столбце. Все значения, кроме одного являются 0, кроме того, которое является категориальным признаком данной строки [2].

Рассмотрим выше озвученные высказывания на примере столбца 'method', содержащим методы запросов. Всего в нем 3 уникальных значения (рисунок 2).

```
3    ['GET' 'POST' 'HEAD']
GET      11524
POST       115
HEAD         4
Name: method, dtype: int64
```

Рисунок 2 — Уникальные значения в столбце «method»

После применения метода *one hot encoding*, видно, что мы получили список векторов длиной равной 3, и в котором все значения равняются 0, кроме одного (Выведено также количество значений в столбце, чтобы не складывалось мнение о том что, в других столбцах содержаться только 0).

Теперь рассмотрим столбцы содержащие числа:

```
'status_code', 'size_object'
```

Рассмотрим 'size\_object' как видно из статистики по данному столбцу и из графика данные в этом столбце имеют большой разброс, поэтому их необходимо масштабировать, так как без этого результат работы модели может быть хуже (рисунок 3).

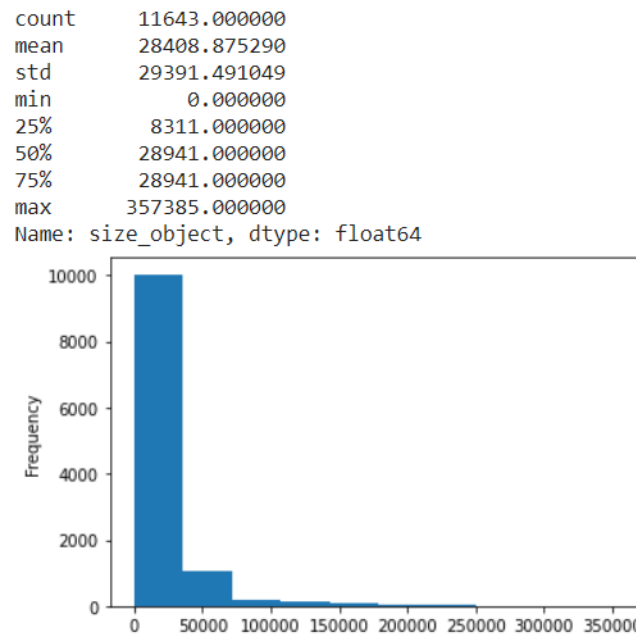


Рисунок 3 — Статистические данные и гистограмма столбца size\_object до масштабирования

После масштабирования видно, что данные были взвешены и помножены на определенной коэффициент, что уменьшило значения в данном столбце. Что и видно из гистограмм. (Значения на осях уменьшились, а вид гистограммы нет)

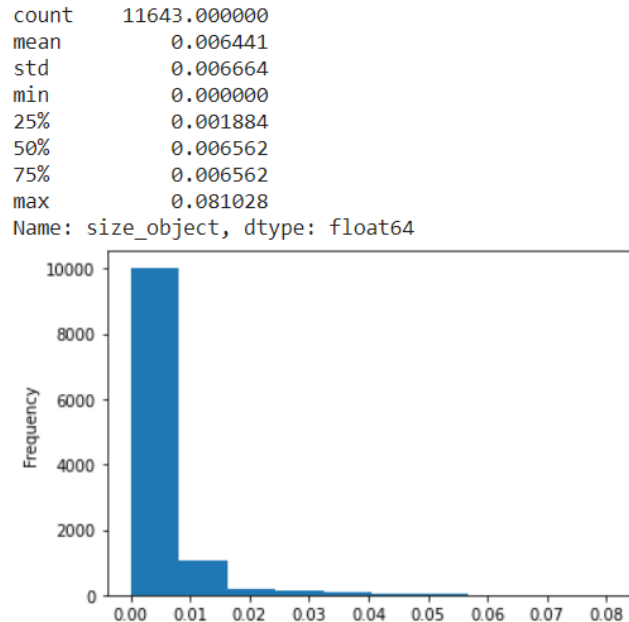


Рисунок 4 — Статистические данные и гистограмма столбца size\_object после масштабирования

Теперь перейдем к рассмотрению 'status\_code' данные в этом столбце не требуют изменений, так как они представляют из себя код состояния значения, которого находятся в интервале от 100 до 526. Убедимся в этом выведя статистическую информацию по нему и построив гистограмму.

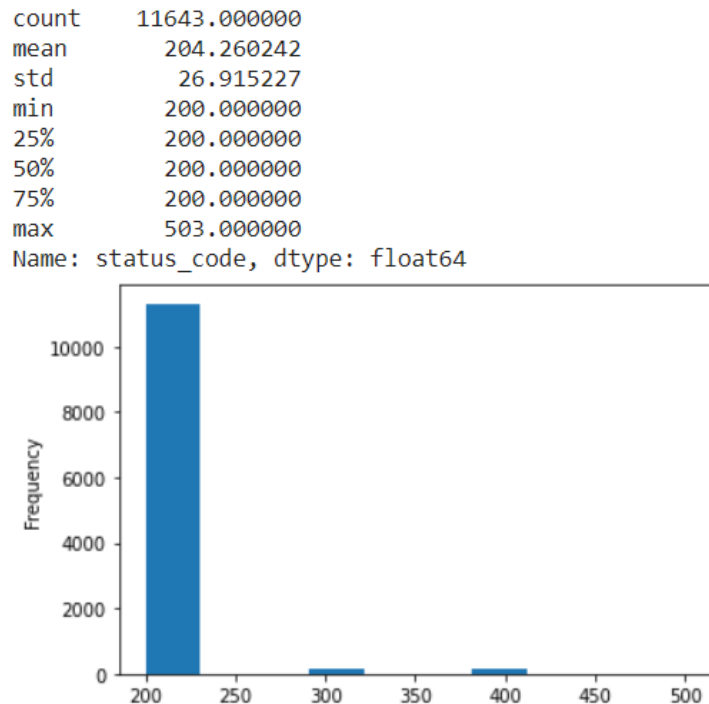


Рисунок 5 — Статистические данные и гистограмма столбца status\_code

Из выведенной информации видно, что предположения подтвердились. Также видно, что в рассматриваемой выборке признак оказался малоэффективен, так как большинство запросов завершается успешно. Однако бывают случаи, когда сервер реагирует на запросы атакующих нетипичным образом, тогда в выборках легитимного трафика и атаки перевес по частоте получают разные коды.

Теперь перейдем к рассмотрению 'number\_availability' данные в ней бинарные, поэтому никакие преобразования тут не требуются.

```
0    10598
1     1045
Name: number_availability, dtype: int64
[0 1]
```

Рисунок 6 — Уникальные данные столбца number\_availability





большим количеством шума. Также они хорошо работают в случаях, когда не известна взаимосвязь между входными и выходными данными.

Все перечисленные преимущества нейронных сетей хорошо подходят для обучения классификатора, который мы хотим получить.

### 1.6 Выбор структуры нейронной сети

Существует ряд архитектур нейронных сетей, которые подходят для задачи классификации, начиная от классической модели Мак-Каллоха-Питса, которая представляет из себя обычный сумматор, заканчивая сверточными нейронными сетями структура, которых намного сложнее. Из этого разнообразия нейронных сетей предстояло сделать выбор, что является достаточно сложной и трудоемкой задачей, так как помимо теоретического подбора архитектур их необходимо было и протестировать, выбрав наилучший результат.

В качестве структуры нейронной сети выбор пал на многослойный перцептрон. Он представляет из себя нейронную сеть с 3 или более слоями, в качестве функции активации используются нелинейные функции (рисунок 8).

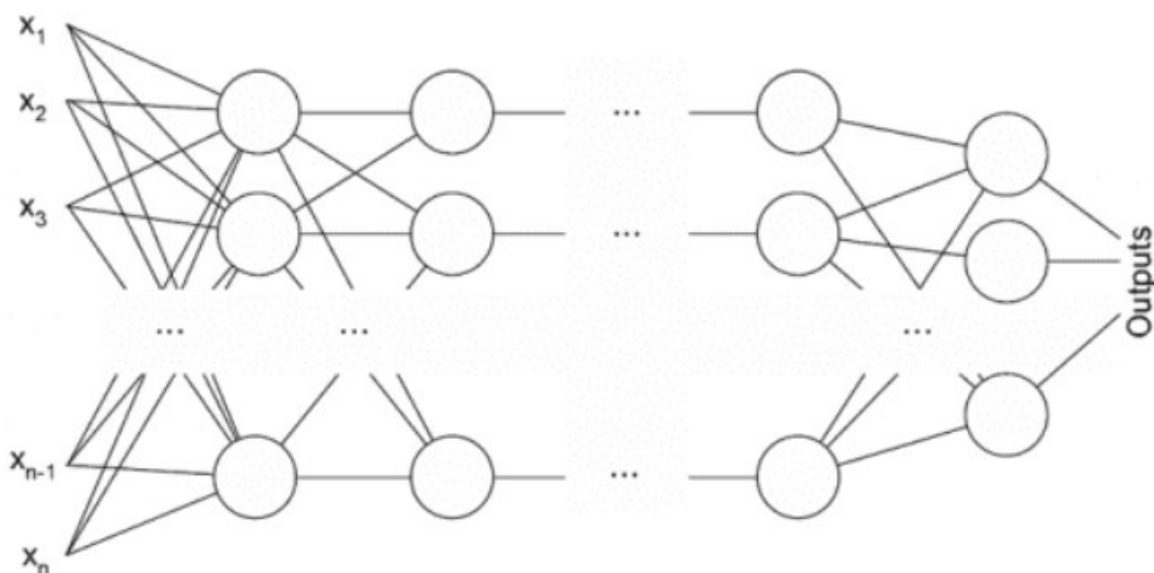


Рисунок 8 – структура многослойного перцептрона

Благодаря тому, что каждый нейрон в одном слое соединен с каждым нейроном в другом слое, сеть является полно связной, а такие сети хорошо

работают при классификации текстов. Наши данные можно отнести к текстовым, так как они преимущественно состоят из слов (страна, континент, название браузера и устройства, с которого был отправлен запрос и т.д)

Также рассматривались следующие архитектуры нейронных сетей:

- Сверточная нейронная сеть
- Рекурсивная нейронная сеть
- Рекуррентная нейронная сеть
- Неглубокие нейронные сети
- Sequence-to-sequence модель
- Сеть долгой краткосрочной памяти

Все они имеют свои особенности и недостатки, но при проведении тестов с ними не получилось достичь желаемого результата (Либо процесс обучения и автоматического подбора гиперпараметров занимал больше времени чем при использовании многослойного перцептрона, либо они не давали желаемого результата точности).

Также перед обучением необходимо подобрать ряд гиперпараметров:

- Функции активации
- Функцию потерь
- Метрику
- Оптимизатор
- Количество эпох
- Архитектуру слоев нейронной сети

В качестве функций потерь для тестов были выбраны следующие функции активации: *sigmoid*, *tanh*, *selu*, *softsing*, так как в качестве архитектуры нейронной сети был выбран многослойный перцептрон, а также они хорошо работают при задачах классификации. На тестах лучшим образом себя показали две функции активации *sigmoid* и *softsin*.

В качестве функции потерь была выбрана *средне квадратичная ошибка*, в качестве метрики была выбрана *accuracy* (доля правильных ответов), они являются стандартными для оценки работы модели. Для цели, стоящей перед нами, они подходят, так как особенностей в оценке качества модели в данной задаче нет.

В качестве оптимизатора был выбран *rmsprop*, так как он позволяет избегать локальных минимумов, а также в отличие от других оптимизаторов, решающих такую же проблему, работает быстрее.

Архитектуру слоев нейронной сети и количество эпох можно подобрать только экспериментальным путем [8].

### **1.7 Процесс обучения нейронной сети**

После того, как экспериментальным путем были подобраны гиперпараметры нейронной сети, ее нужно обучить. На входной слой поступают заранее подготовленные данные, потом в скрытом слое подбирается математическая функция, которая предсказывает ответ. Данный процесс называется прямое распространение ошибки. Затем благодаря корректировке весов и смещению узлов сети минимизируется разница между предсказанным ответом и фактическим результатом. Вышеописанный процесс называется обратное распространение ошибки.

Зачастую прогнать данные один раз через нейронную сеть недостаточно, потому что используется ограниченный набор данных. Мы должны подобрать функцию под них, а это делается с помощью градиентного спуска – интерактивного процесса. Следовательно, обновлять веса после одного раза недостаточно. Данный процесс повторяется некоторое количество раз, равное количеству эпох.

Гиперпараметры подобраны таким образом, что после обучения точность (Precision) модели составляет не меньше 95%, согласно требованию к системе. При данной точности система достаточно обучена, чтобы не классифицировать



легитимный трафик как вредоносный, и при этом не пропускать большое количество атакующих запросов.

## **2 РЕАЛИЗАЦИЯ ПРОГРАММЫ**

### **2.1 Обзор требований**

Полные требования из задания на выпускную работу, уточненные с учетом анализа исходных данных:

1) Загрузка исходных данных в стандартных форматах:

- фрагменты access.log в текстовых файлах или в формате csv;
- журналы защитной системы в формате CSV;
- гибкая настройка параметров, с выбором полей запросов, которые будут использоваться для обучения (в формате txt и csv они имеют разные названия)
- базы данных GeoIP в формате MaxMind (это база данных, где для подсетей отмечены страны местонахождения) [9].

2) Полностью автоматическая настройка для конкретного сайта (выборки).

Подбор полей, на которых будет обучаться модель в зависимости от изначального формата данных, будет выполнен системой автоматически, также система сама определяет доступные мощности машины и использует их максимум, для получения наилучшего результата по времени выполнения.

3) Верификация по тестовой выборке и выдача сведений о качестве обучения. Результат, точности которой составляет не меньше 95%. Это позволит пропускать допустимое количество вредоносного трафика, что существенно снизит нагрузку на сервер и уменьшит вероятность успешной атаки.

4) Алгоритмы работы должны быть выбраны с учетом их потенциальной производительности. Так как в данной системе важна не только точность полученной модели, но и время ее получения, начиная от предварительной обработки данных, заканчивая обучением. Среднее время атаки составляет 15-20 минут, соответственно, чтобы система была эффективной, время получения модели не должно превышать 5-8 минут.

## 2.2 Выбор средств

Перед началом работы требовалось выбрать язык программирования. Выбор пал на Python, поскольку это интерпретируемый язык, что позволяет без особых трудностей тестировать программу, также в нем автоматически отслеживается использование переменных, что позволяет не беспокоиться об освобождении памяти, так как интерпретатор делает это сам. У языка большая пользовательская база, с постоянно пополняющейся базой библиотек для машинного обучения и для считывания исходных данных. Также большим преимуществом является наличие IDE разработанных под машинное обучение. (Jupyter notebook)

Для извлечения из `accessLog` будем использовать библиотеку `apachelogs` [10]. Для обогащения данных по `ip` будем использовать `geoip` [11]. Поле содержащие HTTP заголовков можно обработать, используя библиотеку `user_agents` [12]. Для записи распаршенных данных и вывода статистических данных будем использовать `pandas` [13]. Все выше перечисленные библиотеки кроме `pandas` и `re` узко направлены, и разобраться в них, используя документацию не составит труда, а две другие, наоборот, имеют большое количество учебных пособий.

Для предварительной подготовки данных перед обучением, автоматического подбора гиперпараметров и вывода метрик после обучения для анализа работы классификатора `sklearn` [14]. Она достаточно распространена, поэтому найти по ней учебные пособия не составит труда.

Для формирования и обучения нейронной сети будем использовать `keras` [13]. Библиотека распространена, поэтому найти материалы для ее изучения не составляет труда, она написана по верх библиотеки `tensorflow` [15] инженером Google и имеет постоянную поддержку.

Для запуска нескольких процессов параллельно, для ускорения работы по извлечению данных из текстового файла с логами будем использовать стандартный модуль `multiprocessing` [16]. Он позволяет запустить

несколько процессов и при этом работе не будет мешать GIL, который не позволяет запустить несколько потоков параллельно в прямом понимании. А для параллельной обработки данных из файла формата csv будем использовать библиотеку `pandarallel` [17], но тут стоит отметить что на windows данная библиотека не поддерживается. При этом у нее есть одно достаточно весомое преимущество, она автоматически определяет доступные мощности машины и распределяет выполнение равномерно между ними. Поэтому использование ее не составляет никаких сложностей.

В качестве среды разработки для подбора архитектур нейронных сетей и тестовых функций для подготовки и извлечения данных был выбран google colab, поскольку он является облачным сервисом, что хорошо сказывается на разработке поскольку от компьютера не требуется никаких вычислительных мощностей, так как вся обработка выполняется непосредственно на серверах среды выполнения. Также она направлена на машинное обучение, поэтому не требуется устанавливать библиотеки, направленные на разработку моделей и подготовку данных для их обучения.

После того как удалось подобрать архитектуры нейронных сетей, которые давали необходимую точность (accuracy > 0.95), а функции, отвечающие за подготовку и извлечение данных, начали формировать необходимый датасет, для дальнейшей работы использовался PyCharm. В нем был дописан функционал, отвечающий за настройки, а также весь код был разделен на модули, каждый из которых отвечает за характерные для него задачи. Данная среда разработки была выбрана из-за возможности тщательной настройки виртуального окружения, возможности удаленного тестирования на других устройствах с помощью ssh подключения и возможности автоматического формирования модулей программы (создание файлов не требующих непосредственного кода программы).



## 2.3 Обзор программы

В конечном результате программа представляет из себя прототип вспомогательной системы защиты для «BIFIT MITIGATOR». Она представляет из себя файл запуска и 5 модулей:

- `data_parser` – модуль отвечающий за извлечение данных из двух форматов данных csv и log
- `data_vectorizer` – модуль отвечающий за подготовку данных перед обучением
- `learning_model` – модуль отвечающий за обучение модели, вывод метрик и сохранение обученного классификатора.
- `settings` – модуль отвечающий за общие настройки параметров системы (Пути чтения исходных данных, сохранение данных и т.д)
- `init_modul` – модуль отвечающий за настройки модулей (Название файлов, поля по которым извлекаются данные и т.д)
- файла запуска из себя представляет последовательный запуск функционала классов из модуля `init_modul`

Система выглядит следующим образом (рисунок 9). На ней отображены все модули, кроме модуля, отвечающего за настройки, так как все остальные обращаются к нему практически все время (во время инициализации модуля, загрузки и сохранение данных и т.д).

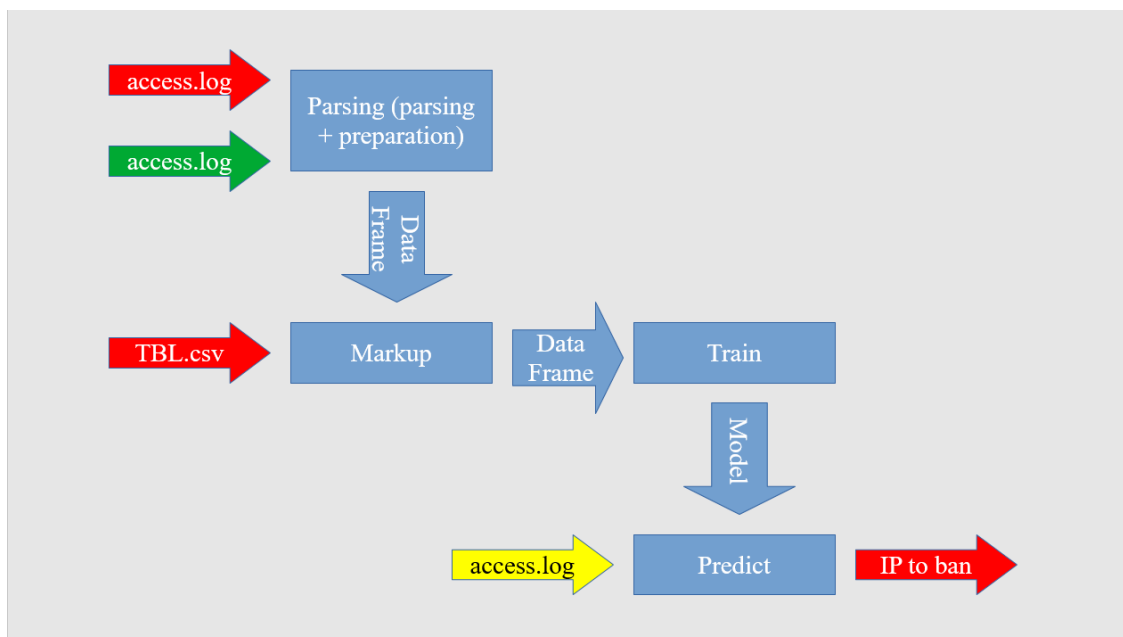


Рисунок 9 — структурная схема системы.

Также благодаря тому, что система работает с использованием нескольких процессов в модулях `data_parser`, `data_vectorizer`, `learning_model` – это позволяет использовать все вычислительные мощности машины (количество доступных процессов определяется автоматически, так же есть возможность указать их вручную), на которой она запущена, что в свою очередь ускоряет работу системы и позволяет обучить модель как можно быстрее. Единственное ограничение – это непосредственно мощность машины, на которой запущена программа.

Реализованная система использует только те поля журнала запросов, которые задал сам пользователь.

Благодаря такой архитектуре программы, система является гибкой, что позволяет индивидуально подстраиваться под различные сервера. Это в свою очередь позволяет использовать все доступные мощности системы, чтобы получить наилучший результат.

### 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

#### 3.1 Тестирование модуля извлечения данных

Тестирование модуля `data_parser` проводилось в два этапа. Сначала на результат корректной работы, а потом на скорость выполнения на различных машинах. Вторая проверка необходима для проверки эффективности системы и для подбора оптимальных комплектующих машины, чтобы обеспечить наилучшую скорость работы системы, что для работы данной системы является одним из основополагающих факторов.

Первая серия тестов проводилась на 6 выборках. Каждая выборка включала фрагмент `access.log` во время атаки, журнал блокировки IP-адресов защитной системой во время атаки и образец `access.log` в период без атаки. Объем выборок был порядка 150-200 тысяч запросов. Критерием проверки была безошибочная загрузка всех данных. В результате тестов было выявлено, что модуль работает без ошибок, но также было выявлено, что только один набор данных подходит для дальнейшей работы.

Вторая серия тестов проводилась на 3 различных системах. (Рисунок 10, рисунок 11, рисунок 12)

```
solovev@nd9:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
Address sizes:          46 bits physical, 48 bits virtual
CPU(s):                 72
On-line CPU(s) list:   0-71
Thread(s) per core:    2
Core(s) per socket:    18
Socket(s):              2
NUMA node(s):          2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                 85
Model name:             Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz

solovev@nd9:~$ cat /proc/cpuinfo|grep processor|wc -l
72
```

Рисунок 10 — характеристики сервера nd9 компании BIFIT.

Элемент	Значение
Имя ОС	Майкрософт Windows 10 Pro
Версия	10.0.19044 Сборка 19044
Дополнительное опис...	Недоступно
Изготовитель ОС	Microsoft Corporation
Имя системы	DESKTOP-R72U310
Изготовитель	Dream Machines
Модель	NHx0DB,DE
Тип	Компьютер на базе x64
SKU системы	Not Applicable
Процессор	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz, 2496 МГц, ядер: 4, логических процессоров: 8
Версия BIOS	INSYDE Corp. 1.07.13TDES, 27.01.2021
Версия SMBIOS	3.2
Версия встроенного к...	7.04
Режим BIOS	UEFI
Изготовитель основно...	Dream Machines
Модель основной пла...	NHx0DB,DE
Версия основной платы	Not Applicable
Роль платформы	Мобильный
Состояние безопасно...	Откл.
Конфигурация PCR7	Для просмотра требуется повышение прав
Папка Windows	C:\Windows
Системная папка	C:\Windows\system32
Устройство загрузки	\Device\HarddiskVolume1
Язык системы	Россия
Аппаратно-зависимы...	Версия = "10.0.19041.1566"
Имя пользователя	DESKTOP-R72U310\danila
Часовой пояс	RTZ 2 (зима)
Установленная операт...	16,0 ГБ

Рисунок 11 — характеристики персонального компьютера.

```

solovev@amd2:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
Address sizes:          43 bits physical, 48 bits virtual
CPU(s):                 128
On-line CPU(s) list:    0-127
Thread(s) per core:     2
Core(s) per socket:     32
Socket(s):              2
NUMA node(s):           2
Vendor ID:              AuthenticAMD
CPU family:              23
Model:                  49
Model name:             AMD EPYC 7502 32-Core Processor

solovev@amd2:~$ cat /proc/cpuinfo|grep processor|wc -l
128

```

Рисунок 12 — характеристики сервера amd2 компании BIFIT.



После того, как мы убедились в эффективности работы модуля в результате первой серии тестов, тесты на скорость выполнения проводились на 3 машинах (на локальной машине и на двух серверах). Тесты на удаленных системах были запущены с помощью подключения по ssh протоколу. Перед запуском на серверах было создано виртуальное окружение и установлены все необходимые для работы библиотеки.

В результате проведения тестов были получены следующие результаты (таблица 1).

Таблица 1 – таблица зависимости скорости работы модуля извлечения данных от устройства, на котором оно было запущено.

Имя устройства	Время, с
Amd2	337.94
Nd9	349.89
Моя машина	257.28

Как видно из тестов скорость выполнения данного модуля на выборке в 200 тыс. запросов составляет большой временной промежуток, что будет сказываться на конечном результате. Модуль работает слишком долго, но так как данная система приступает к распознаванию вредоносного трафика, только после первой атаки – это не является весомой проблемой. Также можно сделать вывод, что время выполнения данного модуля зависит от частоты процессора. Так максимальная частота процессора amd2 составляет 3.35ГГц, а частота на моей машине равна 4.4ГГц.

Результат работы можно ускорить с помощью добавления потоков, но данная работа этого не требует.

### **3.2 Тестирование модуля подготовки данных**

Тестирование данного модуля на корректность работы не требуется поскольку, если данные обработаны неправильно перед обучением, то результат работы классификатора будет не правильным, либо программа при обучении

выдаст ошибку. Поэтому результат работы данного модуля посмотрели эксперты и одобрили результат.

После данный модуль протестировали на скорость работы, для выявления зависимости скорости работы данного модуля от устройства, на котором она запущена.

В результате проведения тестов были получены следующие результаты (таблица 2).

Таблица 2 – таблица зависимости скорости работы модуля подготовки данных перед обучением от устройства, на котором оно было запущено.

Имя устройства	Время, с
Amd2	2,11
Nd9	2,36
Моя машина	1,8

Из результатов, приведенных выше, как и в тестах модуля подготовки данных на быстроту работы модуля влияет частота процессора. Также видно, что количество ядер процессора не влияет на скорость обучения, так как программа в данном модуле работает последовательно. Для достижения лучшего результата можно запустить выполнение данного модуля в несколько потоков.

### **3.3 Тестирование модуля обучения сети.**

Тестирование работы данного модуля проходило в три этапа:

- Оценка результатов работы всего модуля, с использованием всех предложенных архитектур
- Оценка результатов работы каждой из предложенных архитектур
- Оценка времени работы модуля со всеми предложенными архитектурами

Данный модуль автоматически подбирает гиперпараметры из заранее заданных для эффективной работы, поэтому необходимо оценить результат

работы модуля на качество работы со всеми архитектурами. В результате тестирования были получены следующие данные (таблица 3).

Таблица 3 – таблица результатов работы модели с лучшими гиперпараметрами.

Accuracy (аккуратность)	confusion matrix (матрица ошибок)	Precision (точность)	Recall (полнота)	Результат работы модели на тестовых данных
0.998	<code>[2068 2] [ 5 1990]</code>	0.99903	0.997	<code>[0.998021 ] [0.00304052] [0.99764603] ... [0.99534214] [0.00221199] [0.00247964]</code>

Из результатов, полученных выше, видно, что после автоматического подбора гиперпараметров и обучения нейронной сети с их применением, точность работы классификатора составляет 99.7% правильных результатов, что является достаточным результатом для классификатора.

Вторая серия тестов, направленная на демонстрацию качества работы каждой из предложенных архитектур нейронной сети, нужна, так как говорилось уже ранее, для каждого из серверов необходимо будет обучать свой классификатор и, соответственно, набор гиперпараметров может отличаться от того, что был получен на данных, которые использовались для тестирования.

В таблице 4 представлены все архитектуры, которые использовались при подборе гиперпараметров, в дальнейшем будут обозначаться как № архитектуры.

Таблица 4 – таблица архитектур, используемых при подборе гиперпараметров.

№ архитектуры	Структура архитектура
1 архитектура	<code>[Dense(units=100, input_dim=number_input_parameters, activation='softsign'), Dense(units=50, activation='sigmoid'), Dense(units=1, activation='softsign')],</code>
2 архитектура	<code>[Dense(units=100, input_dim=number_input_parameters, activation='sigmoid'), Dense(units=50, activation='softsign'), Dense(units=1, activation='sigmoid')],</code>
3 архитектура	<code>[Dense(units=100, input_dim=number_input_parameters, activation='sigmoid'), Dense(units=50, activation='softsign'), Dense(units=25, activation='sigmoid'), Dense(units=1, activation='softsign')],</code>
4 архитектура	<code>[Dense(units=100, input_dim=number_input_parameters, activation='softsign'), Dense(units=50, activation='sigmoid'), Dense(units=25, activation='softsign'), Dense(units=1, activation='sigmoid')],</code>

Таблица 5 – результаты обученных моделей для каждой из предложенных архитектур

№ архитектуры	Accuracy (аккуратность)	confusion matrix (матрица ошибок)	Precision (точность)	Recall (полнота)	Результат работы модели на тестовых данных
1	0.978	<code>[2069 1] [ 88 1907]</code>	0.9995	0.959	<code>[ 0.9784761 ] [-0.00232134] [ 0.9782206 ] ... [ 0.8136827 ] [-0.00122691] [-0.00136402]</code>
2	0.997	<code>[2068 2] [ 9 1986]</code>	0.99903	0.995	<code>[0.99568164] [0.00192678] [0.9956419 ] ... [0.9754424 ] [0.00124142] [0.00123498]</code>

3	0.997	[2068 2] [ 7 1988]	0.99903	0.996	[9.8393482e-01] [1.8740659e-04] [9.8393357e-01] ... [9.8292738e-01] [1.1089705e-04] [1.3523955e-04]
4	0.998	[2068 2] [ 5 1990]	0.99903	0.997	[0.998021 ] [0.00304052] [0.99764603] ... [0.99534214] [0.00221199] [0.00247964]

Результаты тестирования показали, что каждая из архитектур дает высокий порог точности при использовании их в качестве гиперпараметра. Это показывает, что архитектуры подобраны хорошо, так как их полнота составляет больше 95%. При рассмотрении архитектур такой результат не удалось получить, модель была либо не до обучена, либо переобучена.

Третья серия тестов была на скорость выполнения (таблица 6), она проходила также на 3 машинах. Сначала работу модели запускали с использованием одного ядра процессора, а после с использованием всех доступных ядер машины, это позволило производить вычисления на нескольких ядрах процессора параллельно, что существенно ускорило работу модуля.

Таблица 6 - таблица зависимости скорости работы модуля обучения модели от устройства, на котором оно было запущено и от количества ядер на нем.

	На одном ядре	На всех доступных ядрах	Количество доступных ядер

Amd2	5344.108	520.06	128
Nd9	6186.76	748,7	78
Моя машина	5429.86	1659.45	8

Как видно из результатов тестирования (таблица 6) предположения озвученные выше подтвердились, при использовании всех доступных мощностей процессора скорость увеличивается почти в 5 раз.

Чтобы проверить наличие зависимости между количеством ядер и увеличением скорости работы модуля проведем дополнительную серию тестов.

Таблица 7 – зависимость скорости работы модуля от количества ядер, которые используются

Количество ядер	Скорость выполнения, с
10	660,89
20	633,91
40	498,11
45	550,73
55	570,28
60	563,37
80	555,23
100	550,077
128	520,06



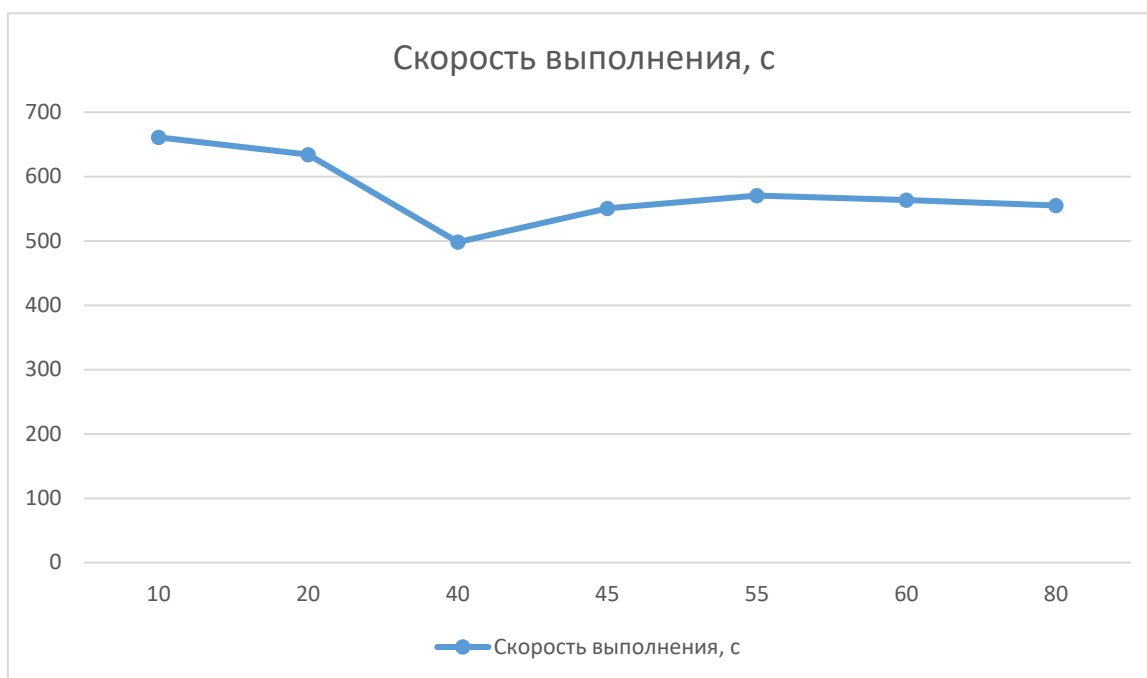


Рисунок 12 — зависимость скорости работы модуля от количества ядер, которые используются.

Как видно из результатов тестирования (таблица 7, рисунок 12) линейной зависимости между увеличением количества ядер и скоростью выполнения нет, поэтому нельзя использовать максимальное количество ядер в надежде получить максимальную скорость. Также видно, что наилучшее время достигается при использовании 40 ядер. Стоит сказать, что для чистоты выполнения во время тестирования были завершены все сторонние процессы.

### 3.4 Сквозное тестирование

Последней серией тестов, которые необходимо провести является проверка последовательного запуска всех написанных модулей. Так как она зависит не только от выше описанных модулей, но и от модулей настройки параметров, инициализация которых тоже занимает некоторое время.

Таблица 8 – зависимость скорости работы всей системы от устройства, на котором она была запущена

Имя устройства	Время выполнения, с
----------------	---------------------

Amd2	945.6
Nd9	1131.994
Моя машина	1865.45

По результатам тестирования видно, что наилучший результат удалось получить на устройстве Amd2 с максимальной частотой процессора 3,35ГГц и 128 ядрами. Это заняло около 15 минут, отсюда можно сделать вывод, что система будет работать только после первой атаки, но при этом с большой эффективностью, определяя не безопасный трафик с точностью 99,7 %.

### **3.5 Проблема введения системы в эксплуатацию**

Несмотря на то, что на тестах система показала неплохие результаты она не может быть введена в эксплуатацию по следующим причинам.

Первой, и одной из наиболее важных причин, является невозможность убедить клиентов передавать данные куда-либо. А предварительно настроить систему и обучить модель без них невозможно, так как под каждый сервер это необходимо делать индивидуально, а используя выборку с другого сервера, получить желаемый результат будет попросту невозможно.

Второй, и не менее важной причиной, является выгрузка данных с сервера, на котором они хранятся, и передача результата работы для внесения его в список блокировок. Данные с запросами пользователей находятся на одном сервере, а результаты работы сетевых фильтров могут быть записаны в разные журналы, так как данные в них записываются от типа запроса, доступ к котором зачастую получить не так просто, к тому же для нового клиента это будет необходимо делать вручную, что не совсем отвечает требованию автоматической системы.

Третьей причиной является то, что зачастую сетевой фильтр хорошо выполняет свою работу и поэтому запросов, содержащих атаку не будет в журнале запросов, а соответственно не на чем будет обучать классификатор.

Последней проблемой данной системы является требование больших мощностей и недостаточно быстрое время подготовки готовой модели, так как несмотря на достаточно быстрое время работы, этого все равно будет недостаточно для успешного отражения атаки.

## ЗАКЛЮЧЕНИЕ

Обосновано применение классификатора HTTP-запросов по данным журнала web-сервера для задачи обнаружения DDoS-атак типа HTTPS flood. Проанализирован состав исходных данных и предложены методы их предварительной обработки.

Проанализированы методы машинного обучения подходящие под задачи классификации, из них был выбран метод под названием нейронные сети, подходящий под данную задачу наилучшим образом. Были проанализированы гиперпараметры, дающие наилучшие результаты модели.

Весь написанный код был оформлен в виде модулей, каждый из которых отвечает за свою цель.

Также проведены тесты по времени и эффективности работы каждого отдельного модуля и всей системы на основе которых можно сделать вывод, что скорость работы можно увеличить, добавив для выполнения обработки данных несколько потоков. По результатам тестов можно сделать вывод, что система сильно зависима от комплектующих машины, на которых запущена, так как в процессе работы проводится большое количество вычислений требующих больших мощностей.

Но в реальных условиях данная система не удалось внедрить по нескольким причинам:

- Клиентов сложно заставить отдать данные
- Подбор гиперпараметров и предварительная подготовка данных требует слишком больших мощностей
- Сложно получать данные с серверов, на которых они сохраняются
- Схема передачи и получения данных получается сложнее чем казалось изначально

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Минкомсвязи РФ. Приказ № 148 от 31.03.2020: [сайт]. URL: <https://digital.gov.ru/uploaded/files/prikaz-148-gv.pdf>
- 2 Защита от DDoS-атак // DDoS Guard: [сайт]. URL: <https://ddos-guard.net/ru/store/web>
- 3 DDoS-атаки и BGP-инциденты третьего квартала 2021 года // Qrator Labs: [сайт]. URL: <https://habr.com/ru/company/qrator/blog/584814/>
- 4 HTTPS filtering without decryption (PCI-DSS ready) // Qrator Labs: [сайт]. URL: <https://qrator.net/en/qrator-technologies/https-ne>
- 5 Анализатор логов web-сервера // BIFIT MITIGATOR: [сайт]. URL: <https://docs.mitigator.ru/v21.10/integrate/log-analyzer/>
- 6 Пример успешного использования машинного обучения для защиты от DDoS атак: [сайт]. URL: <https://habr.com/post/136237/>
- 7 R. Fielding, Ed., J. Reschke, Ed. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content: [страница в интернете]. URL: <https://datatracker.ietf.org/doc/html/rfc7231>
- 8 Выбор метода векторизации для категориальных признаков - Орельен Жерон. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow. Концепции, инструменты. — Диалектика, 2020 г. — 1040 с.
- 9 IP Geolocation and Online Fraud Prevention // MaxMind: [сайт]. URL: <https://www.maxmind.com/>
- 10 Библиотека apachelogs: [сайт]. URL: <https://github.com/jwodder/apachelogs>
- 11 Библиотека geoip: [сайт]. URL: <https://github.com/maxmind/GeoIP2-python>
- 12 Библиотека user\_agents: [сайт]. URL: <https://github.com/selwin/python-user-agents>
- 13 Библиотека pandas: [сайт]. URL: <https://github.com/pandas-dev/pandas>
- 14 Библиотека sklearn: [сайт]. URL: <https://github.com/scikit-learn/scikit-learn>
- 15 Библиотека keras: [сайт]. URL: <https://github.com/keras-team/keras>

- 16 Библиотека tensorflow: [сайт]. URL: <https://github.com/tensorflow/tensorflow>
- 17 Библиотека multiprocessing: [сайт]. URL: <https://github.com/uqfoundation/multiprocess>
- 18 Библиотека pandarallel: [сайт]. URL: <https://github.com/nalepae/pandarallel>