**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Институт (Филиал)** № 8 «Компьютерные науки и прикладная математика» **Кафедра** 806

**Группа** М8О-407Б-19  **Направление подготовки** 01.03.02 Прикладная математика и информатика

**Профиль** Информатика

**Квалификация**  **бакалавр**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

На тему: Разработка интеллектуальной системы для выявления аномалий сетевого трафика

Автор ВКРБ\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляшун Дмитрий Сергеевич (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Руководитель Сошников Дмитрий Валерьевич (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Консультант (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Консультант (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Рецензент (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

**К защите допустить**

Заведующий кафедрой 806 Крылов Сергей Сергеевич (\_\_\_\_\_\_\_\_\_)

(№ каф) (фамилия, имя, отчество полностью)

\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Москва 2023

**РЕФЕРАТ**

Выпускная квалификационная работа бакалавра состоит из 76 страниц, 5 рисунков, 7 таблиц, 48 использованных источников, 6 приложений.

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА, СЕТЕВОЙ ТРАФИК, АНОМАЛИЯ, СЛУЧАЙНЫЙ ЛЕС, ГРАДИЕНТНЫЙ БУСТИНГ, НЕЙРОННАЯ СЕТЬ

Объект исследования – аномалии сетевого трафика в компьютерных сетях.

Цель работы – разработка интеллектуальной системы для выявления аномалий сетевого трафика.

В процессе работы проводились исследования условий применимости различных моделей машинного обучения и нейронных сетей, а также использования простых и производных характеристик сетевого трафика на качество обнаружения аномалий.

Результатом выполнения работы является интеллектуальная система, способная автоматически выявлять отклонения в сетевом трафике локальной сети и предупреждать о возможных угрозах безопасности.

Внедрение результатов данной работы позволяет вывести информационную безопасность предприятия на новый уровень, обнаруживать сложные сетевые атаки в реальном времени.

Дальнейшее развитие интеллектуальной системы представляется в виде реализации надстройки моделей более высокого порядка, производящих уточнение и категоризацию возникших аномалий.

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc135823463)

[ВВЕДЕНИЕ 6](#_Toc135823464)

[1 ОПИСАНИЕ СЕТЕВЫХ АНОМАЛИЙ И ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ИХ ОБНАРУЖЕНИЯ 9](#_Toc135823465)

[1.1 Важность обнаружения аномалий сетевого трафика 9](#_Toc135823466)

[1.1.1 Укрепление безопасности предприятия 9](#_Toc135823467)

[1.1.2 Улучшение производительности приложения 9](#_Toc135823468)

[1.1.3 Оптимизация пользовательского опыта 10](#_Toc135823469)

[1.1.4 Упорядочивание процессов для службы безопасности 10](#_Toc135823470)

[1.1.5 Применение в различных отраслях 11](#_Toc135823471)

[1.2 Интернет-трафик 11](#_Toc135823472)

[1.3 Описание сетевых аномалий 12](#_Toc135823473)

[1.3.1 Классификация аномалий 13](#_Toc135823474)

[1.3.2 Проблема обнаружения аномалий 16](#_Toc135823475)

[1.4 Анализ существующих подходов в обнаружении аномалий 17](#_Toc135823476)

[1.4.1 Сигнатурные и адаптивные (интеллектуальные) методы 17](#_Toc135823477)

[1.4.2 Методы обнаружения аномалий и злоупотреблений 18](#_Toc135823478)

[1.4.3 Гибридные методы 22](#_Toc135823479)

[2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ ОБНАРУЖЕНИЯ АНОМАЛИЙ 23](#_Toc135823480)

[2.1 Постановка задачи 23](#_Toc135823481)

[2.2 Архитектура системы 23](#_Toc135823482)

[2.3 Описание моделей искусственного интеллекта 28](#_Toc135823483)

[2.3.1 Случайный лес 28](#_Toc135823484)

[2.3.2 Бустинг на решающих деревьях 30](#_Toc135823485)

[2.3.3 Многослойная ИНС с прямой связью 34](#_Toc135823486)

[2.4 Стек используемых технологий 40](#_Toc135823487)

[2.4.1 Python 40](#_Toc135823488)

[2.4.2 Tcpdump и Libpcap 45](#_Toc135823489)

[2.4.3 Nfdump 45](#_Toc135823490)

[2.4.4 Clickhouse 45](#_Toc135823491)

[2.5 Описание программной разработки 46](#_Toc135823492)

[2.5.1 Захват и преобразование трафика 46](#_Toc135823493)

[2.5.2 Визуализация трафика 47](#_Toc135823494)

[2.5.3 Подготовка данных для ядра СОА 48](#_Toc135823495)

[2.5.4 Ядро СОА 51](#_Toc135823496)

[3 РЕЗУЛЬТАТЫ И ОЦЕНКА РАБОТЫ СИСТЕМЫ 53](#_Toc135823497)

[3.1 Разведочный анализ данных 53](#_Toc135823498)

[3.1.1 Анализ активности и поведения сетевых узлов 54](#_Toc135823499)

[3.1.2 Заключения разведочного анализа 56](#_Toc135823500)

[3.2 Обучение моделей 57](#_Toc135823501)

[3.3 Оценка качества моделей 58](#_Toc135823502)

[3.4 Предложения по дальнейшим улучшениям 60](#_Toc135823503)

[ЗАКЛЮЧЕНИЕ 62](#_Toc135823504)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 63](#_Toc135823505)

[ПРИЛОЖЕНИЕ А QR-код ссылки на репозиторий проекта в Github 67](#_Toc135823506)

[ПРИЛОЖЕНИЕ Б QR-код ссылки на фрагмент используемого набора данных 68](#_Toc135823507)

[ПРИЛОЖЕНИЕ В Пример графиков параллельных координат активности узлов сети 69](#_Toc135823508)

[ПРИЛОЖЕНИЕ Г Пример графиков интенсивности трафика узлов сети 71](#_Toc135823509)

[ПРИЛОЖЕНИЕ Д Пример графиков суммарного объёма трафика узлов сети 73](#_Toc135823510)

[ПРИЛОЖЕНИЕ Е Пример графиков числа уникальных взаимодействующих хостов узлов сети 75](#_Toc135823511)

# **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

|  |  |
| --- | --- |
| БД | – база данных |
| ГИС | – гибридная интеллектуальная система |
| ИИ | – искусственный интеллект |
| ИНС | – искусственная нейронная сеть |
| ОС | – операционная система |
| ПО | – программное обеспечение |
| СОА | – система обнаружения аномалий |
| СТГ | – статистическая группа |
| ТКО | – телекоммуникационное оборудование |
| ACK | – acknowledgment flag |
| API | – application programming interface |
| CWR | – congestion window reduced flag |
| DDoS | – distributed denial-of-service |
| DNS | – domain name system |
| DoS | – denial-of-service |
| ECE | – explicit congestion notification echo flag |
| FIN | – finish flag |
| HTTP | – hypertext transfer protocol |
| ICMP | – Internet control message protocol |
| ID | – identifier |
| IP | – Internet protocol |
| PCAP | – packet capture |
| PSH | – push flag |
| RST | – reset flag |
| SMOTE | – synthetic minority oversampling technique |
| SQL | – structured query language |
| SVN | – subversion |
| SYN | – synchronization flag |
| TCP | – transmission control protocol |
| UDP | – user datagram protocol |
| URG | – urgent flag |
| UUID | – universal unique identifier |

# **ВВЕДЕНИЕ**

Недавние громкие инциденты в таких компаниях, как SolarWinds, Microsoft, T-Mobile, Facebook, Colonial Pipeline, JBS и Acer, свидетельствуют о значительном ущербе, наносимом передовыми кибератаками. Такие типы атак могут успешно обходить конечные точки и область защиты, оставаться скрытыми и активными в сетях в течение длительного времени и способны нанести значительный ущерб.

Системы обнаружения аномалий сетевого трафика разрабатываются специально для реагирования на современные киберугрозы. Главное преимущество таких систем – способность выявлять неизвестные или скрытые угрозы путём анализа сетевой активности на предмет отклонений в реальном времени.

Различные пакеты сетевых данных имеют уникальные сигнатуры, такие как заголовки, всевозможные форматы представления, кодирования данных и т. д. Используя системы обнаружения аномалий сетевого трафика, предприятия могут автоматически составить карту приемлемого поведения сетевых пакетов, чтобы выделить любые необычные события. Данные этих необычных событий далее анализируются, чтобы отсеять шум и ложные срабатывания. Полученные предупреждения отправляются в группу управления сетью или группу информационной безопасности для оперативного решения проблемы.

Системы обнаружения аномалий сетевого трафика значительно упрощают процесс мониторинга сети, предоставляя предварительно созданные аналитические модели, интеграции и меры по снижению ложных срабатываний. Они также могут выявлять аномалии, которые ускользают от традиционных систем безопасности конечных точек.

73% специалистов по кибербезопасности согласны с тем, что постоянный мониторинг сети имеет решающее значение для получения полной картины аномалий безопасности и защиты конечных точек, согласно отчёту от ENEA Qosmos [1]. Также стоит отметить, что системы обнаружения аномалий сейчас продолжают активно внедряться, и 55% респондентов в том же опросе заявили, что они уже внедрили или планируют внедрить это решение в отдельный продукт или в качестве более комплексной системы.

Можно выделить следующие ключевые особенности систем обнаружения аномалий сетевого трафика:

1. Непрерывный мониторинг сети – обнаружение аномалий сетевого трафика является ключевым элементом информационной безопасности. Это постоянно включённая деятельность, непрерывный мониторинг поведения сети для поиска потенциальных угроз. Ключевые параметры поведения сравниваются с приемлемыми стандартами и историческими образцами, поэтому инструмент выявляет любое событие, которое выделяется.
2. Детальная осведомлённость о поведении сети, поскольку недостаточно просто выделить необычное сетевое событие без какой-либо справочной информации. Инструмент должен обеспечивать детальную осведомлённость и контекстуализированную информацию для принятия соответствующих мер.
3. Оповещения в реальном времени – это основная функция инструмента обнаружения аномалий сетевого трафика. Оповещения в реальном времени позволяют команде управления сетью получать информацию о потенциальной угрозе сразу после её обнаружения, не дожидаясь запланированного отчёта или проверки приборной панели.
4. Встроенные или подключённые системы реагирования, т. к. обнаружение аномалий поведения сети является частью более широкой функции обнаружения и реагирования сети. Подобно другим системам безопасности, таким как брандмауэры или программное обеспечение для предотвращения вторжений, отправляющим предупреждения, данные о сетевых аномалиях также должны обрабатываться для быстрого реагирования. Некоторые инструменты имеют встроенную функцию, чтобы пользователи могли исследовать аномалии и реагировать на них в рамках одного рабочего процесса. Другие системы могут быть подключены к стороннему программному обеспечению cо схожими возможностями.

Обеспечение информационной безопасности с помощью систем обнаружения аномалий сетевого трафика является жизненно важным для достижения целостной защиты.

Таким образом, исследования в этой области являются довольно актуальными с практической точки зрения.

Целью работы является разработка интеллектуальной системы для выявления аномалий сетевого трафика в виде модели искусственного интеллекта. Сама система должна быть проста в использовании и настройке, иметь возможность быстрого развёртывания в любых условиях.

Для достижения поставленной цели в работе были решены следующие задачи:

* проведён анализ существующих методов обнаружения аномалий;
* структурирован и формализован подход в классификации сетевых аномалий;
* построена и реализована архитектура интеллектуальной системы;
* сформирован набор простых и производных входных данных о сетевом трафике для выявления аномалий;
* проверена применимость наиболее популярных методов искусственного интеллекта в рамках решении задачи обнаружения аномалий: случайного леса, бустинга на деревьях, многослойной ИНС с прямой связью.

Программирование системы и реализация моделей искусственного интеллекта осуществлялось на языке Python3 c использованием таких библиотек как:

* Pandas – для обработки и анализа структурированных данных;
* Keras, Tensorflow – для реализации моделей глубокого обучения;
* Matplotlib, Plotly – для визуализации данных двумерной и трёхмерной графикой;
* Numpy – для линейной алгебры и математических вычисления над матрицами;
* Imblearn – для проведения балансировки исходного набора данных;
* XGBoost, CatBoost – для использования моделей градиентного бустинга.

Также в ходе разработки интеллектуальной системы были задействованы следующие программы и утилиты:

* Clickhouse – для представления данных о сетевом трафике в виде БД;
* Nfdump – для обработки сетевого трафика в формате Netflow;
* Tcpdump & libpcap – для захвата сетевого трафика и последующего хранения в виде файлов формата PCAP.

Основными результатами, полученными в работе, являются:

* успешное применение разработанной интеллектуальной системы в ходе выявления аномалий сетевого трафика реального предприятия;
* достижение высокой точности обнаружения аномалий для моделей бустинга на деревьях и многослойной ИНС с прямой связью.

Результаты работы предназначены для дальнейшей интеграции в программный комплекс Стетоскоп компании ЦСС-Безопасность – анализатор для мониторинга сетевых потоков между компонентами информационных систем организации и сетью Интернет [2].

Использование разработки позволяет вывести информационную безопасность предприятия на новый уровень, обнаруживать сложные сетевые атаки в реальном времени.

# **1** **ОПИСАНИЕ СЕТЕВЫХ АНОМАЛИЙ И ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ИХ ОБНАРУЖЕНИЯ**

## 1.1 Важность обнаружения аномалий сетевого трафика

Обнаружение аномалий сетевого трафика заключается в процессе мониторинга корпоративных сетей с целью выявления аномального поведения и является неотъемлемой частью современных систем кибербезопасности. Оно обеспечивает подробный обзор сетевой активности и позволяет командам безопасности обнаруживать и реагировать на современные сетевые угрозы.

Можно выделить ряд ключевых моментов, подчёркивающих важность обнаружения аномалий сетевого поведения для различных отраслей.

### 1.1.1 Укрепление безопасности предприятия

В процессе обнаружения сетевых аномалий используется статистика, собранная с маршрутизаторов, коммутаторов и сетевых зондов для выявления действий недобросовестных субъектов. В сочетании с дополнительными решениями по кибербезопасности система обнаружения аномалий позволяет выявлять бот-сети, направленные атаки, неизвестные вредоносные программы, утечки данных, внутренние угрозы и другие уязвимости. Кроме того, решение может повысить эффективность работы сети за счёт автоматического обнаружения отклонений и операционных проблем.

### 1.1.2 Улучшение производительности приложения

Пандемия подтолкнула почти каждый вид бизнеса к созданию цифрового присутствия. Поскольку конкуренция в онлайн-пространстве достигает новых высот, производительность приложений часто является наиболее значимым решающим фактором в получении дохода. Внутренние приложения могут определять производительность труда сотрудников, а приложения, ориентированные на клиентов, должны быть быстрыми и безопасными, чтобы привлекать и удерживать клиентов.

Традиционно команды разработчиков применяли «реактивный подход» к мониторингу производительности приложений и реагировали на проблемы безопасности только после их возникновения. В результате предприятие может пострадать от последствий целенаправленной атаки *–* иногда даже до того, как будет обнаружено наличие проблемы и предприняты меры по исправлению ситуации.

### 1.1.3 Оптимизация пользовательского опыта

Динамическое реагирование на потребности пользователей является жизненно важным в настоящее время. Если организация не сможет соответствовать запросам пользователей, она, скорее всего, уступит конкурентам. Поэтому команды разработчиков должны постоянно стремиться к тому, чтобы новые функции внедрялись как можно быстрее.

Однако не каждая версия приложения может пройти тщательное тестирование перед выпуском. Ошибки и недочёты в системе безопасности могут периодически проскальзывать, что в лучшем случае приведёт к негативному опыту пользователей, а в худшем *–* к потере конфиденциальных данных субъектами угроз.

Система обнаружения аномалий сетевого трафика отслеживает производительность различных продуктов в сети с момента их развёртывания. Эти данные повышают ценность отчётов о каждом A/B-тестировании, модификации воронки покупок или других изменениях процесса.

Поведенческие аномалии *–* будь то признаки серьёзных угроз или мелких ошибок *–* могут быть устранены более эффективно с помощью системы обнаружения аномалий. Это решение может помочь обеспечить своевременную реакцию на DDoS-атаки, непредвиденные изменения и другие сбои, вызванные выпуском ошибочных версий.

Обнаружение аномалий сетевого поведения легко сопоставляет соответствующие сетевые метрики с существующими поведенческими данными, предоставляя разработчикам оперативные отчёты в случае нарушения производительности приложения. Этот инструмент использует данные, собранные в течение длительного времени, чтобы выявить потенциальные проблемы с производительностью приложения до того, как пользователи могут пострадать.

### 1.1.4 Упорядочивание процессов для службы безопасности

Поскольку в настоящее время недостатки безопасности используются быстрее, чем когда-либо прежде, любому онлайн-бизнесу необходимо сохранять бдительность и обеспечивать бесперебойную работу. Для предотвращения серьёзных нарушений жизненно важно реагировать на угрозы безопасности в режиме реального времени. Обнаружение аномалий в поведении сети может помочь смягчить все *–* от ошибок API и простоя сервера до сбоев при загрузке и т. д. *–* благодаря мгновенному предоставлению отчётов. Всесторонний охват сети и минимизация времени отклика в бизнес-сетях *–* вот основные принципы обнаружения аномалий сетевого трафика.

### 1.1.5 Применение в различных отраслях

Решения по обнаружению аномалий сетевого поведения играют ключевую роль в повышении уровня безопасности организаций во многих отраслях:

1. Электронная коммерция *–* именно здесь обнаружение аномалий сетевого поведения может помочь в выявлении таких проблем, как скачки напряжения в сети, простои и даже сбои в ценообразовании.
2. Телекоммуникации *–* являясь одним из крупнейших производителей данных временных рядов, телекоммуникационные компании должны бороться с аномалиями в своих сложных сетях или рисковать простоем. Обнаружение аномалий в поведении сети может помочь контролировать такие параметры, как качество звонков, задержка и джиттер, чтобы активно решать проблемы производительности в режиме реального времени.
3. Интернет-реклама – в этой индустрии ежедневно обрабатываются миллиарды транзакций, а расчёты в реальном времени происходят в течение миллисекунд. Здесь можно использовать обнаружение аномалий в поведении сети для непрерывного мониторинга транзакций и поддержания *status-quo*.
4. Видеоигры – игровые компании могут использовать обнаружение аномалий сетевого поведения для мониторинга операционных систем, сегментов пользователей, статистики, используемых устройств и многого другого, чтобы быстро устранять сбои и ошибки, которые могут привести к негативному пользовательскому опыту.
5. Финансы – это отрасль, где безопасность имеет первостепенное значение. Здесь обнаружение аномалий сетевого поведения может использоваться как часть своих мер безопасности для безопасной обработки транзакций и противодействия продвинутым атакам до того, как они нанесут масштабный ущерб.

## 1.2 Интернет-трафик

Интернет-трафик, или сетевой трафик – это количество данных, передаваемых через сеть Интернет за определённый период времени [3]. Трафик является ключевым аспектом работы Интернета, и его объём постоянно растёт с развитием технологий и увеличением числа пользователей.

Сетевой трафик может быть измерен в числе переданных и полученных байт, а также пакетах. Обычно он возникает, когда пользователь посещает сайты, загружает файлы, отправляет или получает электронные письма, смотрит видео, играет в онлайн-игры и т. д. Каждое действие пользователя в Интернете приводит к передаче данных через сеть.

Существует различные типы сетевого трафика. Входящий трафик – это данные, которые пользователь получает из сети, например, при загрузке страницы. Исходящий трафик – это данные, которые пользователь отправляет в сеть, например, при отправке электронной почты. Кроме того, сетевой трафик может быть как локальным, то есть передаваться внутри одной сети, так и глобальным – передаваться через Интернет.

Ключевой составляющей любого сетевого трафика являются набор его пакетов.Структура сетевых пакетов включает в себя заголовок и полезную нагрузку. Заголовок содержит информацию о том, каким образом пакет должен быть обработан при передаче по сети, а также сведения о его источнике и назначении. Полезная нагрузка представляет собой данные, которые нужно передать. Также в структуру сетевых пакетов могут входить дополнительные поля, которые содержат информацию о контроле целостности данных, управлении потоком и других параметрах.

Сетевые пакеты передаются между устройствами по определённым протоколам, которые определяют правила передачи данных и обмена информацией. Одни из наиболее распространённых протоколов являются TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ICMP (Internet Control Message Protocol) и HTTP (Hypertext Transfer Protocol). Каждый протокол имеет свои особенности и используется для различных целей, таких как передача данных, управление сетевым трафиком и обмен сообщениями.

Сетевые аномалии могут быть обнаружены именно на основе анализа содержимого заголовков пакетов, например адреса отправителя и получателя, порта назначения и источника и многого другого. Также в случае возможности дешифровки полезной нагрузки возможна и его обработка на предмет аномалий.

## 1.3 Описание сетевых аномалий

Любая корпоративная сетевая инфраструктура включает в себя множество составляющих, функционирующих независимо или взаимодействующих между собой, и каждый из этих компонентов является потенциальным источником сетевых аномалий. И проблемы в функционировании компьютерных сетей прежде всего обнаруживаются именно с помощью вызванных ими аномалиями трафика.

В общем случае «аномалия» – отклонение от нормы, неправильность [4]. Примером аномалии может являться повреждение коммутатора, который привёл к созданию непредвиденного трафика, или появление неопознанных кодов ошибок при отключении одной из службы.

Для повышения точности определения неправомерных действий внутри защищённого периметра сети необходимо учитывать множество факторов: источники и причины возникновения сетевых аномалий, количество аномалий и их потенциальную связь между собой, процесс преобразования зарегистрированных аномалий сетевого трафика в событие информационной безопасности.

### 1.3.1 Классификация аномалий

Существуют различные подходы в классификации аномалий сетевого трафика. В качестве основных классификационных признаков используются [5]:

* тип источника,
* причина возникновения,
* область (место) возникновения,
* способ проявления,
* характер изменений.

Классификация аномалий по типу источника включает в себя такие группы как:

* ошибки, не связанные с человеческим фактором – отказ оборудования или отсутствие связи;
* человеческая ошибка – неправильная настройка оборудования;
* злонамеренная человеческая деятельность – внутренняя атака, когда сотрудник компании целенаправленно повреждает систему, или внешняя атака, когда злоумышленник пытается нанести ущерб сети.

Помимо категоризации основных типов источников, важным аспектом изучения проблемы аномалий в сетевом трафике является исследование причин их появления в компьютерной сети. Можно выделить следующие две категории причин [6]:

* аномалии, связанные с производительностью – причинами их появления могут являться:

1. ошибки проектирования аппаратных и программных составляющих систем;
2. износ физических компонентов;
3. некорректные внедрение, конфигурация, использование и администрирование составляющих компьютерной сети;
4. особенности работы телекоммуникационных систем, которые не могут быть исправлены программными средствами;

* аномалии, связанные с безопасностью – к данной категории можно отнести:

1. использование уязвимостей аппаратного или программного обеспечения;
2. недостаточная настройка средств защиты и мониторинга сети;
3. ошибки в политике безопасности организации – что позволяет пользователям и злоумышленникам совершать неправомерные или непредусмотренные политикой действия во внутреннем защищённом периметре сети.

Описание наиболее часто возникающих причин сетевых аномалий представлено в таблице 1.1.

Таблица 1.1 – Примеры распространённых причин аномалий

| Причина сетевой аномалии | Описание | Влияние на изменение трафика |
| --- | --- | --- |
| DoS-, DDoS-атака | Отправление большого количества запросов на сервер или устройство, что может привести к отказу в обслуживании | Критическое повышение объёма трафика от множества удалённых источников к одному адресу назначения |
| Деятельность червей | Вредоносная программа, которая самостоятельно распространяется по сети и использует уязвимости ОС | Заметный выброс в трафике одного или нескольких портов назначения без доминирующего адреса |
| Отключения | Неполадки в сети, вызывающие падение трафика между парой источник-назначение | Падение объёма трафика обычно до нуля. Возможна долговременная неактивность потоков источник-назначение от или к одному маршрутизатору |
| Переключения потока | Изменение пути трафика с одного входящего маршрутизатора на другой | Падение в объёме трафика по одному потоку и повышение по другому. Может затрагивать сразу несколько потоков трафика |
| Точка-мультиточка | Распространение контента от одного сервера многим пользователям | Выброс в объёме трафика от одного источника к нескольким назначениям по наиболее известному порту |
| Альфа-аномалия | Необычно высокий уровень трафика типа точка-точка | Критическое повышение объёма трафика с доминирующим потоком источник-назначение непродолжительное время (до 10 минут) |

Продолжение таблицы 1.1

|  |  |  |
| --- | --- | --- |
| Причина сетевой аномалии | Описание | Влияние на изменение трафика |
| Сканирование сети/портов | Сканирование одного хоста по всем портам или сканирование сети по определённым открытым портам с целью поиска уязвимостей | Скачек в трафике по числу потоков в секунду, c небольшим числом пакетов в каждом от одного источника |
| Перегрузка | Непредвиденно повышенный спрос на сервис или сетевой ресурс | Непродолжительный скачек по числу потоков в секунду к источнику по одному единому порту |
| Анализаторы трафика (сниферы) | Программное или аппаратное обеспечение, которое позволяет перехватывать и анализировать сетевой трафик, проходящий через определённый участок сети | При работе в активном режиме – перенаправление трафика на другой узел или изменение содержимого пакетов |
| Сетевые модификаторы | Программы или устройства, которые изменяют сетевой трафик, проходящий через них | Изменение объёма трафика, его направления, скорости передачи и содержания |

Область возникновения аномалий также является важным критерием для классификации, поскольку позволяет локализовать проблемную область для последующего устранения. Аномалии могут возникать в следующих узлах сети [5]:

* внутренние, локальные сервера,
* удалённые сервера,
* маршрутизаторы и межсетевые экраны,
* рабочие станции.

Способ проявления – ещё один подход в классификации аномалий, его группа включает в себя [5]:

* изменение трафика,
* перезагрузка сетевого оборудования и сервисов,
* отключение и отказ сетевого оборудования.

Последний тип классификации – характер изменений трафика, является самым важным, поскольку именно его составные группы система обнаружения аномалий должна определять. Данная категория включает в себя следующие отклонения:

* аномалии на основе объёма трафика – увеличение или уменьшение общего объёма трафика в сети;
* аномалии на основе времени – изменение паттернов трафика в определённое время;
* аномалии на основе поведения – изменение поведения пользователей в сети, например, большое количество неудачных попыток аутентификации;
* аномалии на основе протоколов – использование необычных или запрещённых протоколов в сети;
* аномалии на основе маршрутов – изменение маршрутов трафика в сети.

Таким образом, классификация аномалий сетевого трафика довольно разнообразна. Каждый тип атак имеет свои особенности и может привести к различным последствиям для безопасности сети и конфиденциальности данных. Поэтому важно иметь системы защиты, которые могут обнаруживать и предотвращать различные типы аномалий сетевого трафика.

### 1.3.2 Проблема обнаружения аномалий

В реальности определить аномалию не так просто, как это может показаться на первый взгляд. При мониторинге сети могут возникнуть проблемы, которые способны существенно ограничить возможности корректного обнаружения аномалий.

#### 1.3.2.1 Ложноположительные срабатывания

Ввиду изменяющихся условий и строения сети то, что вчера было обычным явлением, спустя время может стать аномалией. Это связано с тем, что передаваемые данные меняются независимо от того, есть аномалия или нет. Именно поэтому процесс обнаружения аномалий скорее основан на оценке вероятности произошедшего события, и если она превышает заранее заданный порог, то тогда его можно расценивать как аномалию.

Пример ложноположительного события – в результате рекламы интернет-портала на него стало заходить большее число посетителей, чем это было в среднем. Таким образом, нельзя гарантировать, что удастся правильно и своевременно обнаружить все аномалии, а также исключить факт ложных срабатываний, из-за которых во время автоматической обработки событий легитимный трафик или служба могут быть идентифицированы как проблемные, что приведёт к снижению их активности. При этом обработка и анализ подобных аномалий вручную тоже довольно затруднителен, поскольку требует огромных затрат времени и сил.

#### 1.3.2.2 Зашифрованность трафика

По соображениям безопасности и конфиденциальности алгоритмы шифрования в компьютерных сетях постоянно расширяются и совершенствуются. Зашифрованный трафик сильно ограничивает возможность обнаружения аномалий, поскольку уменьшает объем данных, который может быть подвержен мониторингу и анализу. Например, в случае мониторинга зашифрованной электронной почты email-адреса становятся недоступными. Однако в рамках сетевых пакетов шифрованию подвергается только часть данных на уровне приложения, что не отменяет возможность статистического анализа числа соединений, используемых служб, портов и т. д. К сожалению, злоумышленники осведомлены об ограниченности работы систем выявления аномалий, и скрывают свою деятельность в зашифрованном сообщении, чтобы предотвратить любую возможность обнаружения.

## 1.4 Анализ существующих подходов в обнаружении аномалий

### 1.4.1 Сигнатурные и адаптивные (интеллектуальные) методы

Методы обнаружения сетевых атак по заранее созданным шаблонам (сигнатурам) являются традиционными и основаны на сличении текущих событий сетевого трафика с базой известных правил на их основе [7]. Сигнатура точно описывает, какой тип данных ищет система. Примером сигнатуры может быть поиск пакета, который имеет одинаковые IP-адреса отправителя и получателя, или поиск определённого содержимого в пакете. Чтобы сигнатуры были точными и выявляли известные аномалии в сети, их необходимо создавать вручную, используя знания о каждой проблеме или атаке.

Использование сигнатурного анализа даёт максимальную уверенность при детектировании ранее известной записанной атаки, однако плохо раскрывает атаки «нулевого дня» или необычные атаки [8]. Также стоит принять во внимание тот факт, что злоумышленник при должном знании используемой базы правил на предприятии способен организовать свою атаки в обход обнаружению.

В качестве примеров систем с таким подходом можно выделить Snort и Suricata как наиболее распространённые и с открытым исходным кодом [9].

Также существуют адаптивные методы сигнатурного анализа, которые способны модифицировать имеющуюся базу правил при необходимости [7]. Такой подход сейчас стремительно набирает популярность, но ему всё же присуща вероятностная природа. При обучении адаптивной модели можно установить точность распознавания, и она едва ли окажется равной 100%. Нивелирование этого недостатка представляется неотъемлемой частью поиска наилучшего метода распознавания атак для возможности выявлять те угрозы, которые будут пропущены сигнатурной системой обнаружения вторжений.

### 1.4.2 Методы обнаружения аномалий и злоупотреблений

В компьютерных сетях можно выделить модель «нормального сетевого трафика», которая фиксирует основные взаимодействия между компонентами сети. И тогда если в процессе работы показатели отклонения текущего трафика относительно обычного профиля системы превышают установленные пороговые значения, то такой трафик будет расценён как аномальный. Именно на этом подходе основаны методы обнаружения аномалий и злоупотреблений [8].

В состав системы обнаружения аномалий обычно входит модуль моделирования, отвечающий за построение нормального профиля сети, и модуль обнаружения, выявляющих аномалии сетевого трафика, который проектируется адаптивным с целью независимости – отсутствия привязанности к конкретному шаблону при сопоставлении аномалий.

Суть злоупотреблений заключается в том, что они являются такой цепочкой событий, которые заведомо ведут к эксплуатированию уязвимостей и другому вреду информационной системы. С позиции обнаружения аномалий или злоупотреблений используются общие методы интеллектуального анализа данных, такие как нейронные сети, генетические алгоритмы, деревья решений и др. Отличие лишь связано с тем, что рассматривается по умолчанию как главное, а что – как цель для выявления [10].

#### 1.4.2.1 Методы статистической группы

Группа методов статистического анализа является основой систем обнаружения аномалий. К ней можно отнести: анализ распределений интенсивности сетевого трафика, метод х-квадрат, цепи Маркова и т. д. При использовании таких методов прежде всего необходимо подобрать подходящий исходный набор данных, являющийся заведомо нормальным и повседневным, с большим охватом по времени, иначе полученная модель описания сетевого трафика окажется слишком привязанной к конкретному промежутку или чрезвычайно обобщённой. В свою очередь это может вызвать большое число ложных срабатываний и неспособность к реальному обнаружению вторжений.

К преимуществам таких методов стоит отнести ясность полученных результатов и принципов работы, возможность в адаптивности при возникающих изменениях в поведении пользователей локальной сети, а также обнаружение особых модифицированных атак, поскольку сами атаки не берутся во внимание, а только их механизм воздействия.

Вейвлет-анализ позволяет обнаружить значимые области в большом наборе данных и при этом сгладить незначимые шумы. Его суть заключается в использовании специальных базисных функций, которые применяются для разложения исходного сигнала по соответствующим коэффициентам. В качестве сигнала можно рассматривать интенсивность сетевого трафика в единицу времени [11].

По результатам разложения вейвлет-анализ выявляет самую весомую информацию на основании наиболее высокой амплитуды соответствующих колебаний сигнала, при этом колебания сигнала с меньшей амплитудой вовсе игнорируются. Тогда выделение наиболее значимых сигналов и их сравнение с эталонными сигналами исходной базовой выборке позволяет обнаружить вторжение в локальную сеть [12].

Фрактальный анализ использует принцип самоподобных интервалов данных на отрезках времени с разным масштабом, который удовлетворяет свойству самоподобия. При этом было установлено, что результирующее значение показателя фрактального анализа, так называемый показатель Херста, будет разным у аномального и нормального трафика, что в свою очередь значительно упрощает принятие решения о наличии вторжения при выявлении аномального трафика [13].

К статистическим методам также относится кластерный анализ сетевого трафика. Его суть заключается в выделении таких характеристик, на основе которых можно произвести однозначное разделение на группы нормального и аномального поведения [14]. Однако данные подход заметно отличается от классического метода классификации, поскольку набор классов формируется только в процессе самого анализа. Сигналом об обнаружении вторжения здесь выступает совпадение текущего состояния с одной из аномальных групп или несовпадение характеристик ни с одной из нормальных групп.

Если аномальные состояния сетевого трафика можно удобно отделить от нормальных состояний линейной гиперплоскостью, то в таком случае применим метод опорных векторов. Задача его заключается в том, чтобы обнаружить гиперплоскость, от которой ближайшие к ней объекты классов будут находиться максимально далеко, что позволяет заметно снизить или же полностью устранить возникающую ошибочную классификацию. Однако далеко не всегда такое отделение возможно, поэтому на практике исходные данные сетевого трафика обычно сперва как-то обрабатывают, – например, с помощью вейвлет-анализа [15].

#### 1.4.2.2 Методы искусственного интеллекта

Методы искусственного интеллекта называются таковыми, поскольку в их устройстве заложена имитация различных сторон человеческого мышления. К этой группе методов можно отнести: генетические алгоритмы, иммунные системы, нейронные сети, деревья решений, байесовские сети, экспертные системы.

Одними из наиболее перспективных методов решения задачи обнаружения аномалий являются всевозможные модели искусственных нейронных сетей. Они представляет из себя набор искусственных нейронов, которые имеют связи между собой – синапсы, имеющие различные числовые веса, изменяющиеся в ходе обучения нейронной сети. Выходом нейрона является значение его функции активации, которая считается на основании суммы произведений входных сигналов x и весов связанных синапсов, как показано в формуле (1.1):

где – значения входных сигналов, поступивших на вход нейрона;

– веса синапсов соответствующих сигналов, изменяющихся в ходе обучения;

– смещение с весом .

Нейронные сети могут состоять как из одного слоя нейронов, так и из множества слоёв – такая сеть уже называется многослойной. В многослойных нейронных сетях исходные данные поступают на первые слой, далее в качестве сигналов для следующего слоя принимаются значения выходных сигналов предыдущего слоя [16].

В результате обучения веса нейронной сети набор весов её синапсов приобретает такое значение, что на выходе модели с некоторой точностью начинает предсказываться желаемая характеристика – например, классификация возникшей сетевой атаки. Возможно также применение сразу нескольких нейронных сетей, каждая из которых обучается для детектирования конкретного вида атаки [17]. В таком случае они будут являться надстройкой над другой нейросетью, которая только обнаруживает вторжение.

Стоит отметить, что у искусственных нейронных сетей есть существенный недостаток – длительность обучения. Также не всегда представляется возможным объяснить, какие закономерности выявляет модель в процессе работы.

Метод анализа данных дерево решений также применим в рамках решения задачи обнаружения аномалий [18]. Он состоит из совокупности элементов следующих видов: узлы – атрибуты, по значениям которых происходит переход по одному из рёбер, исходящих из этого узла; рёбра – элементы, соединяющие узлы в дереве; листья – итоговые решения метода, хранят метки классификации данных.

Строится дерево решений с помощью различных итеративных алгоритмов, а в ходе работы происходит последовательный просмотр поступающих атрибутов в узлах и, таким образом, проход по дереву до конечного листа. При этом в процессе выбора пути обхода никогда не возникает неоднозначности.

В основе методов «байесовская сеть» лежит установление вероятностных зависимостей между анализируемыми данными. Тогда при возникновении отклонений данных от установленных вероятностей можно распознать аномалию сетевого трафика [19].

Наивный байесовский классификатор является частным случаем модели байесовской сети, с предположением об отсутствии смысловой зависимости входных переменных полностью основывается на статистической вероятности присутствия определённого типа данных в обучающей выборке. Данный метод на достаточно больших интервалах считается довольно эффективным, и при описанном упрощении довольно близок к группе статистических методов.

#### 1.4.2.3 Комбинация методов статистической группы и искусственного интеллекта

Прямая обработка методами ИИ потока сетевого трафика потребует значительных объёмов дополнительной памяти и длительного периода обучения. В свою очередь методы СТГ способны быстрее обрабатывать большие массивы данных, их применение на предварительном этапе обработки сетевого трафика снизит размеры необходимой памяти информационной системы, а также число ложных срабатываний ввиду возможного наличия шумов [20].

Учитывая преимущества и недостатки двух групп методов, целесообразно использовать статистические методы как основу обнаружения отклонений, которые затем будут обрабатываться методами искусственного интеллекта. Это позволит значительно повысить скорость работы итоговой интеллектуальной системы, которую называют гибридной (ГИС). Современные исследования в этой области связаны с определением наиболее эффективной комбинации методов СТГ и ИИ. При этом довольно перспективным направлением является также включение в полученную систему механизма внешнего указания ошибок со стороны эксперта-учителя, чтобы модель могла внести поправки в свой метод принятия решения [7].

### 1.4.3 Гибридные методы

Некоторые методы обнаружения аномалий основаны на использовании обеих вышеописанных подходов, и это довольно оправдано, поскольку не всегда возникающее отклонение можно посчитать представляющим угрозу. Для уточнения этого целесообразно задействовать второй этап проверки, т. е. применить сигнатурный анализ к подозрительному трафику. Таким образом повышается вероятность в принятии правильного решения [21].

# **2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ ОБНАРУЖЕНИЯ АНОМАЛИЙ**

## 2.1 Постановка задачи

На основе данных сетевого трафика предприятия решить задачу классификации сетевых узлов. Полученная интеллектуальная система должна уметь классифицировать узлы с точностью выше 90%, в результате чего будет составлен портрет поведения устройств в сети, и при реальной работе, в случае ошибочной классификации, поведение узлов будет расцениваться как аномальное.

Итоговая система должна уметь детектировать аномалии в реальном времени и работать непрерывно.

В ходе работы необходимо составить архитектуру системы обнаружения аномалий и её программно реализовать. Также потребуется формализовать требования к набору входных данных, провести тщательный разведочный анализ имеющегося сетевого трафика, оценить наличие выбросов, сбалансированности присутствия различных сетевых узлов и т. д.

Для решения задачи предлагается использовать методы искусственного интеллекта как наиболее перспективные – необходимо провести проверку подходящих из них и выбрать наилучший.

## 2.2 Архитектура системы

В данном разделе будет рассмотрено построение базовой архитектуры интеллектуальной системы обнаружения аномалий. Схема данной архитектуры показана на рисунке 2.1.

Рассмотрим каждый из компонентов архитектуры в отдельности.

Модуль захвата трафика – это программное обеспечение, которое используется для перехвата и записи сетевого трафика в файловом представлении. Данный модуль работает путём установки на устройство, которое находится на пути трафика – на маршрутизаторе или коммутаторе. Наиболее удобным форматом представления захваченного трафика является PCAP, хранящий данные о сетевых пакетах, поскольку он имеет поддержку различных протоколов и предусматривает сжатие файлов.

Рисунок 2.1 – Базовая архитектура СОА

Изображение выглядит как диаграмма, текст, План, зарисовка

Автоматически созданное описание

Модулем преобразования трафика является набор программного обеспечения, производящего конвертацию данных о соединениях из файла PCAP в формат таблицы БД. Цель модуля преобразования трафика – облегчить анализ и мониторинг сетевого трафика для последующего модуля визуализации трафика и модуля подготовки данных.

В случае представления данных сетевого трафика в формате БД появляется ряд преимуществ. Во-первых, БД обеспечивает структурированное хранение данных, что позволяет легко и быстро находить нужную информацию. Во-вторых, в большинство БД встроен удобный механизм SQL-запросов, который упрощает агрегацию данных. В-третьих, БД может быть использована для хранения больших объёмов данных, что позволяет производить анализ трафика за длительный период времени.

Поскольку основным объектом обработки ядра системы обнаружения аномалий являются сетевые потоки, набор захваченных пакетов из PCAP требует агрегации, что вызывает необходимость представления в ещё одном, промежуточном формате, с информацией о всех входящих в сетевой трафик потоках. Наиболее удобным видом представления для этого является формат Netflow, который включается в себя следующую информацию [22]:

* IP-адрес отправителя и получателя,
* номер порта отправителя и получателя,
* используемый протокол (TCP, UDP, ICMP и т. д.),
* объем переданных данных: число пакетов и байт,
* скорость передачи данных: число пакетов в секунду и байт в секунду,
* время начала и окончания передачи данных,
* продолжительность передачи данных,
* флаги TCP,
* прочая малозначимая информация: MAC-адреса отправителя и получателя, метки MPLS и т. д.

В ходе преобразования в формат Netflow также стоит задача проверки данных в PCAP файлах на целостность, поскольку возможные ошибки сетевого оборудования могут привести к записи далеко не всей нужной информации. Для этого также применяется фильтрация, которая отбрасывает записи о потоках с пустыми полями.

Последним шагом работы модуля преобразования трафика является добавление данных о соединениях формата Netflow в используемую БД.

Модуль визуализации трафика позволяет пользователю увидеть на примере различных диаграмм и графиков те данные, которые в данный момент подлежат обработке в ядре СОА. При составлении первоначального портрета поведения сетевых узлов этот модуль помогает провести разведочный анализ данных для оценки корректности исходного набора данных трафика. Составляются графики следующего вида:

1. график встречаемости каждого узла в сети;
2. график числа записей о соединениях за каждый день;
3. статистические графики поведения узлов – интенсивность входящего и выходящего трафика, суммарный объём трафика, число взаимодействующих узлов и т. д.

Модуль визуализации трафика может помочь лучше понять, как используется сеть, и выявить такие проблемы как узкие места в пропускной способности или неожиданный трафик.

После заключения данных о сетевом трафике в табличную форму встаёт задача представить эту информацию в таком виде, который будет подаваться на вход моделям искусственного интеллекта, поскольку в реальности набор данных о сетевом трафике обладает рядом проблем:

* Несбалансированность присутствия соединений сетевых узлов, что негативно сказывается на качестве обучения моделей.
* Довольно малый контекст информации в случае обработки только одной записи о соединении конкретного сетевого узла, что делает неспособным выявление сложных аномалий.
* Наличие большого числа узлов с виртуальными, подменяющимися IP адресами, которые имеют непродолжительное время жизни и довольно хаотичное поведение, что также оказывает влияние на результате обучения моделей.

Всё это требует учёта и принятия соответствующих мер. Для формирования сбалансированного набора в модуле подготовки данных может применяться такой подход как отбор строго фиксированного числа записей по каждому сетевому узлу, а также генерация синтетических данных в случае недобора до установленного количества записей. В свою очередь проблема малого контекста информации решается путём проведения агрегации по соединениям в рамках фиксированного временного интервала т.е. вычисления таких параметров как суммы (по числу соединений, объёма переданных и полученных байт), среднего (скорость передачи данных, размер пакета) и числа уникальных значений (удалённых хостов, протоколов, входных и выходных портов и т. д.). И, наконец, узлы с виртуальными IP адресами могут быть выделены в отдельную группу неизвестных устройств, что также поможет решить ещё одну проблему – появление неизвестного IP адреса, который ранее мог не присутствовать в обучающей выборке.

Ещё одна ключевая задача модуля подготовки данных – вычисление производных параметров, поскольку имеющейся информации Netflow зачастую недостаточно для успешного разграничения поведения сетевых узлов. При анализе потоков можно выделить большое количество таких параметров, например:

* Количество запросов к серверам с различными сервисами: DNS, HTTP/HTTPS, FTP, SMTP, POP3 и т. д. Это поможет более точно представлять поведения узлов.
* Время суток, когда на узле наблюдается наибольшая активность – это может помочь определить, в какое время суток узел наиболее подвержен атакам или неправомерной активности.
* Количество соединений, завершённых с ошибкой – это может помочь определить, какие проблемы сетевой стабильности могут возникать на данном узле.
* Количество соединений, завершённых по истечении времени – это может помочь определить, какие проблемы сетевой стабильности могут возникать на данном узле.
* Среднее время между отправкой пакетов от узла.

Это только малая часть возможных производных параметров. Все они должны подбираться в зависимости от того трафика, с которым предстоит работа. Например, в случае параметров запросов к серверам можно выделить в уже имеющемся трафике наиболее популярные типы сервисов и использовать только их.

Следующим структурным элементом выступает ядро системы обнаружения аномалий, которое состоит из модулей составления портрета и оценки.

Модуль составления портрета отвечает за первоначальное обучение используемой модели искусственного интеллекта. Важнейшим начальным шагом его работы является оценка значимости сформированного набора параметров. Это требуется с целью отбора наиболее важных признаков и удаления несущественных, что заметно ускорит итоговое время работы системы. Данный процесс подразумевает многократное обучение модели искусственного интеллекта с разными подмножествами рассматриваемого набора параметров, что весьма трудозатратно. Проще всего проводить такую оценку путём работы с легковесными моделями, общее время обучения которых непродолжительно.

После отбора значимых признаков модуль составления портрета выполняет обучение целевой модели. С учётом вариативности параметризации здесь потребуется задействование оптимизаторов, которые смогут автоматизировать этот процесс.

Финальный шаг работы модуля составления портрета – оценка качества выученного поведения сетевых узлов полученной целевой моделью на примере рассмотрения некоторой тестовой выборки. В случае неприемлемого уровня точности распознавания узлов потребуется проведение перенастройки модуля подготовки данных для формирования иного набора производных параметров.

Центральный и конечный элемент системы обнаружения аномалий – модуль оценки, который применяет итоговую обученную модель искусственного интеллекта на реальном трафике для выявления аномалий. Суть его работы заключается в попытке предсказания сетевого узла, имея только набор параметров его поведения. Если предсказание окажется неверным, то его действия расцениваются как аномальное – и система зафиксирует это событие.

Полученная итоговая архитектура интеллектуальной системы обнаружения аномалий обладает рядом преимуществ, а именно:

1. Возможность реорганизации работы каждого из модулей в отдельности, без внесения существенных правок в другие модули. Например, в ядре СОА при желании могут быть использованы любые модели искусственного интеллекта.
2. Структурные элементы системы являются исчерпывающими и покрывают любые возникающие потребности при развёртывании решения в разных условиях.
3. Система позволяет автоматизировать процесс проверки любых возникших гипотез – как качества моделей, так и значимости используемого набора параметров.

## 

## 2.3 Описание моделей искусственного интеллекта

Для решения задачи обнаружения аномалий в сетевом трафике были выбраны следующие модели искусственного интеллекта как наиболее перспективные: случайный лес, бустинг на решающих деревьях, многослойная ИНС с прямой связью.

### 2.3.1 Случайный лес

Случайный лес – алгоритм машинного обучения с учителем, который построен на основе алгоритмов дерева решений. Он использует ансамблевое обучение, которое представляет собой метод, объединяющий множество классификаторов для решения сложных проблем.

Более формально, случайный лес – это классификатор, состоящий из набора древовидных классификаторов , где – независимые идентично распределённые случайные векторы, и каждое дерево отдаёт единичный голос за наиболее популярный класс при поступлении на вход некоторого вектора [23].

Алгоритм случайного леса предсказывает результат на основе предсказаний составных деревьев решений. Результирующее значение определяется как среднее или среднее арифметическое выводов различных деревьев. Увеличение числа деревьев повышает точность результата.

Можно выделить следующие преимущества алгоритма случайного леса:

* Высокая точность и надёжность за счёт использования большого количества деревьев решений для составления прогнозов.
* Возможность обработки недостающих значений, которые могут быть вычислены двумя способами. Первый – использование медианных значений для замены непрерывных переменных, второй – подсчёт среднего арифметического недостающих значений.
* Возможность отбора признаков по степени важности для предсказания результата.

В свою очередь недостатками алгоритма случайного леса являются:

* Вычислительная сложность, поскольку для предсказания случайного леса необходимо провести голосование среди всех входящих в него деревьев решений.
* Трудности в интерпретировании результата в отличие от одного дерева решений.

**2.3.1.1 Дерево решений**

Дерево решений – это классификатор, выраженный в виде рекурсивного разбиения пространства экземпляров [24]. Дерево решений состоит из узлов, которые образуют корневое дерево, то есть это направленное дерево с центральным узлом – «корнем», который не имеет входящих рёбер. Все остальные узлы имеют ровно одно входящее ребро. Узел с исходящими рёбрами называется внутренним или тестовым узлом. Все остальные узлы называются листьями (также известными как терминальные узлы или узлы принятия решений).

В дереве решений каждый внутренний узел разбивает пространство экземпляров на два или более подпространств в соответствии с определённой дискретной функцией значений входных атрибутов. В простейшем и наиболее часто встречающемся случае каждый тест рассматривает один атрибут, так что пространство экземпляров разбивается в соответствии со значением атрибута. В случае числовых атрибутов условие относится к диапазону.

Каждый лист приписывается к одному классу, представляющему наиболее подходящее целевое значение. В качестве альтернативы лист может содержать вектор вероятности, указывающий на вероятность того, что целевой атрибут имеет определённое значение. Экземпляры классифицируются путём перемещения от корня дерева вниз к листу в соответствии с результатами тестов на этом пути.

В случае числовых атрибутов деревья решений можно геометрически интерпретировать как набор гиперплоскостей, каждая из которых ортогональна одной из осей. Сложность дерева оказывает решающее влияние на его точность, и она явно контролируется используемыми критериями остановки и методом обрезки. Обычно сложность дерева измеряется одной из следующих метрик: общее количество узлов, общее количество листьев, глубина дерева и количество используемых атрибутов. Индукция дерева решений тесно связана с индукцией правил. Каждый путь от корня дерева решений к одному из его листьев может быть преобразован в правило простым объединением тестов вдоль пути для формирования антецедентной части, а в качестве значения класса берётся предсказание класса листа.

**2.3.1.2 Построение случайного леса**

Название леса случайным связано с тем, что:

* *случайно* выбирается обучающий набор данных для построения каждого дерева леса;
* генерируется *случайное* подмножество признаков при расчёте вывода для разделения узлов.

При построении случайного леса используется подход бутстрэп, предложенный в 1977 году Брэдли Эфроном. Идея бутстрэпа заключается в генерации множества выборок из исходных данных путём случайного выбора элементов с повторением. Каждая выборка имеет ту же длину, что и исходные данные, но может содержать повторяющиеся элементы [25].

Пусть имеется некоторая выборка с числом элементов в ней равным , из которой равномерно и равновероятно выбирается элементов с возможными повторениями. Повторяя процедуру раз, получаем набор выборок . Полученные выборки используются далее для обучения каждого из классификаторов в случайном лесе. Такой метод называется бэггингом (bagging = bootstrap aggreagation), и был предложен Лео Бреманом в 1996 году [26].

Используя подходы бутстрэпа и бэггинга, алгоритм построения модели случайного леса в общем случае работает следующим образом:

1. Из общего набора данных выбирается случайная выборка методом бутстрэпа для построения решающего дерева.
2. Случайным образом выбирается признаков из общего числа признаков, где . Опытным путём были получены рекомендации использовать в задачах классификации значение
3. Среди признаков вычисляется узел , используя наилучшую точку разбиения.
4. Узел разделяется на дочерние узлы, используя наилучшее разбиение.
5. Шаги с 2 по 4 повторяются, пока не будет достигнуто количества узлов.
6. Шаги с 1 по 5 повторяются раз для создания деревьев. Каждое из деревьев создаётся на своей выборке , что представляет собой бэггинг.

### 2.3.2 Бустинг на решающих деревьях

Бустинг на решающих деревьях является аддитивной моделью машинного обучения и делает прогнозы путём объединения решений из последовательности базовых моделей, которыми являются деревья решений. Формально можно записать эту модель как:

где – итоговая модель бустинга,

– один из базовых классификаторов, т. е. дерево решений.

Преимущества бустинга на решающих деревьях:

* Поддержка обработки категориальных признаков без дополнительного кодирования, а также отсутствующих значений.
* Высокая точность в сравнении с другими методами машинного обучения.
* В сравнении с другими моделями – более быстрое обучение на больших наборах данных.

К недостаткам модели можно отнести:

* Склонность к переобучению – решается путём применения регуляризации, т. е. добавления штрафов на сложность модели в целевую функцию обучения.
* Высокая затрата вычислительных ресурсов при обучении.
* Продолжительность обучения.
* Сложность в интерпретации.

**2.3.2.1 Построение бустинга на решающих деревьях**

В отличие от случайного леса, который строит все базовые классификаторы независимо друг от друга, каждый из которых использует свою выборку данных, бустинг на решающих деревьях использует особую технику построения модели, называемую градиентным бустингом.

Название градиентный бустинг происходит от его связи с подходом градиентного спуска в методах оптимизации. Предположим, что необходимо оптимизировать некоторую дифференцируемую функцию – найти значение аргумента , при котором функция принимает минимальное значение. Тогда градиентный спуск работает путём итеративного нахождения:

где – размер шага.

Аналогично, если считать, что – классификатор, обученный на итерации , и – эмпирическая функция потерь, тогда на каждой итерации функция будет перемещаться в направлении отрицательного градиента со значением шага. Следовательно, следующая составная функция выбирается как:

где – размер обучающей выборки,

– признаки экземпляра обучающей выборки,

– класс экземпляра обучающей выборки.

После определения функции классификатор обновляется:

**2.3.2.3 CatBoost**

CatBoost или Categorical Boosting – одна из практических реализаций алгоритма градиентного бустинга [27]. CatBoost вносит следующие уточнения в общую технику градиентного бустинга:

* Возможность обработки категориальных признаков.
* Упорядоченный бустинг, решающий проблему переобучения модели.

Одним из наиболее распространённых методов обработки категориальных данных является одномоментное кодирование, но при большом количестве признаков оно становится невыполнимым. Для решения этой проблемы признаки группируются в категории по целевой статистике (оценка целевого значения для каждой категории) методом упорядочивания образцов, т. е. алгоритм будет полагаться на обучающие примеры, обработанные ранее, тем самым избегается утечка целевой оценки переменной.

Пусть имеется набор данных с образцами. Каждый образец имеет наборов признаков в векторе , а также реальное значение цели . Математически целевую оценку некоторой категориальной переменной элемента под номером из набора данных можно представить как:

где – индикаторная функция, принимающая значение 1 при условии, что -ая компонента вектора равна -ой компоненте вектора ,

– упорядоченный случайным образом набор данных, рассмотренный ранее,

функция, возвращающая порядок рассмотрения -го элемента набора данных ,

и – параметры, предохраняющие арифметическое недополнение.

Эта техника также гарантирует использование всех доступных прошлых данных для каждого примера для вычисления его целевой статистики и, таким образом, кодирования категориальных переменных.

При использовании упорядоченной целевой статистики для кодирования категориальных переменных градиенты функции потерь по отношению к функции классификатора являются случайными, поскольку используется случайная перестановка для выбора элементов . Таким образом, распределение градиентов может быть смещено в зависимости от кодировки . Этот условный сдвиг приводит к смещению в оценке функции простого классификатора , и значений метрик на данных, которые не использовались во время обучения. Такая проблема называется сдвигом предсказания.

CatBoost вводит упорядоченный бустинг, чтобы избежать этой проблемы. При упорядоченном бустинге выполняется случайная перестановка обучающих примеров и поддерживается различных вспомогательных моделей, -ая модель обучается только на первых выборках в перестановке), и на каждом шаге остаток или ошибка получается путём использования остатков предыдущей модели.

Однако на практике упорядоченный бустинг в описанном выше понимании неосуществим, поскольку данные ограничены, а потребность в памяти для поддержания различных моделей слишком велика. Поэтому в CatBoost данный подход используется с некоторыми упрощением – одна структура дерева (последовательность признаков разбиения) является общей для всех моделей. То есть CatBoost использует тот же , который определил упорядоченную целевую статистику, как данные для определения структуры или подгонки дерева решений , и использует полный набор данных как данные для оценки того, является ли тем деревом решений, которое минимизирует ожидаемые потери. Он использует множество перестановок для вычисления ряда наборов остаточных значений, которые он может задействовать для нахождения , получения и сохранения гарантии того, что ни одно из значений не используется для вычисления значений градиентов. Тем самым уменьшается дисперсия в оценках градиентов (скорость изменения функции потерь) и предотвращается смещение предсказания.

#### 2.3.2.2 XGBoost

XGBoost (eXtreme Gradient Boosting) – это мощный и широко распространённый алгоритм градиентного бустинга, который используется для решения множества различных типов задач машинного обучения. Эта реализация градиентного бустинга специально разработана для обеспечения эффективности и масштабируемости, что делает его популярным выбором для работы с большими наборами данных.

XGBoost реализует общий подход градиентного бустинга, описанный ранее, но с добавлением следующих специфических компонентов [28]:

* Функция потерь аппроксимируется с помощью ряда Тейлора.
* Листья дерева решений содержат веса, которые могут быть регуляризированы.

Веса листьев дерева рассчитываются следующим образом:

где – производная первого порядка от по ,

– производная второго порядка от по ,

– параметр регуляризации,

– вес листа ,

– все экземпляры на этом листе.

### 2.3.3 Многослойная ИНС с прямой связью

Многослойная ИНС с прямой связью — это структура, которая отображает набор входных данных на набор соответствующих выходов. Многослойная ИНС с прямой связью состоит из нескольких слоёв узлов в направленном графе, причём каждый слой полностью соединён со следующим. За исключением входных узлов, каждый узел представляет собой нейрон (или элемент обработки) с нелинейной функцией активации. Модель с прямой связью использует для обучения сети метод контролируемого обучения, называемый обратным распространением, и обладает способностью классифицировать данные, которые не являются линейно разделимыми [16].

Для формального описания нейронов можно использовать так называемую функцию отображения , которая назначает для каждого нейрона подмножество , состоящее из всех потомков данного нейрона, где – множество всех нейронов многослойной ИНС [29]. Тогда подмножество состоит из всех предшественников данного нейрона , поскольку каждый нейрон в определённом слое связан со всеми нейронами в следующем слое. Связь между -м и -м нейроном характеризуется весовым коэффициентом , к выходному значению -го нейрона добавляется также пороговый коэффициентом . Весовой коэффициент отражает степень важности данной связи в нейронной сети. Итоговое выходное значение (активность) -го нейрона определяется уравнениями (2.7) и (2.8):

где – активность -го нейрона,

– передаточная функция.

В ходе проведения обучения происходит изменение пороговых коэффициентов и весовых коэффициентов так, чтобы минимизировать функцию ошибки , устанавливающую связь между вычисленными и требуемыми выходными значениями [29].

Можно выделить следующие достоинства многослойной ИНС с прямой связью [29]:

* Изменяемость – модель способна адаптироваться к переменности данных без участия человека.
* Нелинейность, поскольку нейрон – это нелинейное устройство. Это очень важное свойство, особенно если связь между входом и выходом по своей природе нелинейна.
* Надёжность – модель обладает устойчивостью к шумам во входных данных.

К недостаткам модели стоит отнести:

* Медленная сходимость – отражение того факта, что при построении аппроксимирующего отображения параметров на результат не используется физическое понимание.
* Сложность в объяснении предсказаний и интерпретации процесса, происходящего при обучении.
* Продолжительность обучения из-за большого количества настраиваемых весов, что решается использованием ресурсов GPU.

#### 2.3.3.2 Алгоритм обучения нейронной сети

Алгоритм обучения заключается в повторном вычислении функции ошибки для каждого входа и обратном распространении ошибки от одного слоя к предыдущему. Веса для конкретного узла корректируются прямо пропорционально ошибке в узлах, с которыми он связан.

В основе обратного распространения лежит метод крутой минимизации. Для настройки весовых и пороговых коэффициентов имеет место следующее [29]:

где – скорость обучения (.

Ключевой задачей здесь является вычисление частных производных и . Их расчёт проходит через следующие этапы:

Из уравнений (2.11) и (2.12) вытекает следующее важное соотношение:

Для последующих вычислений достаточно вычислить только .

Если нейрон относится к выходному слою, то значение рассчитывается напрямую. Для нейрона , принадлежащего к внутреннему слою:

поскольку в соответствие с (2.12):

На основе данного подхода можно рекуррентно вычислять производные целевой функции ошибки для выходного слоя, а затем для всех скрытых слоёв. Этот алгоритм называется обратным распространением, поскольку ошибка на выходе распространяется от выходного слоя через скрытые слои к входному слою.

Режим обучения начинается с произвольных значений весов – это могут быть случайные числа – и продолжается итерационно. Каждая итерация полного обучающего набора называется эпохой. В каждую эпоху сеть корректирует веса в направлении, которое уменьшает ошибку по алгоритму обратного распространения ошибки. По мере продолжения итерационного процесса с постепенной подстройкой весов происходит сходимость к локально оптимальному набору значений. Описанный процесс продолжается достаточное число эпох, прежде чем обучение будет завершено.

#### 2.3.3.1 Архитектура нейронной сети

Схема архитектуры используемой многослойной ИНС с прямой связью показана на рисунке 2.2.

Слои Dense представляют структуру связи нейронов и применяют линейное преобразование к вектору , являющемуся выходом предыдущего слоя или входными данными нейронной сети:

где – матрица весов нейронов,

– bias, т. е. значение смещения,

– выход слоя, т. е. вектор с размерностью равной числу нейронов указанного слоя.

Рисунок 2.2 – Архитектура нейронной сети

Изображение выглядит как текст

Автоматически созданное описание

Между всеми слоями Dense в нейронной сети размещаются нелинейные функции активации ReLU (rectified linear unit) для того, чтобы усложнить итоговую представляемую зависимость от входных данных. Значение ReLU вычисляется по следующей формуле:

где – поступивший выход от предыдущего слоя.

Использование ReLU по сравнению с другими функциями активации позволяет быстрее и эффективнее обучать нейросеть на больших и сложных наборах данных.

Поскольку данные на вход нейронной сети подаются группами, называемыми батчами, слои Batch Normalization отвечают за их нормализацию – центрирование и повторное масштабирование. Это делается с целью более быстрого и стабильного обучения сети.

Пусть имеется батч размером , тогда его среднее и дисперсия рассчитывается как [30]:

где – вектор от предыдущего слоя.

Далее производится нормализация по каждому измерению :

где – произвольно малая константа и добавляется в знаменатель для численной стабильности.

Полученное нормализованное значение имеет нулевое среднее и единичную дисперсию, если не учитывать добавление . Важно также отметить, что в ходе практической работы нейронной сети значения и в слое Batch Normalization считаются как средние от всех, что возникали ранее при обучении, поскольку на этапе вывода зависимость от батча больше не является полезной из-за необходимости детерминированности выхода.

Для восстановления репрезентативности сети следует финальный шаг преобразования Batch Normalization – повторное масштабирование, который выглядит следующим образом:

где – компонента вектора , являющаяся результатом преобразования,

– коэффициент масштабирования, который изменяется в процессе оптимизации,

*–* коэффициент смещения, который также является изменяемым.

В предложенной архитектуре многослойной ИНС с прямой связью также задействуются слои Dropout. Они используют подход регуляризации для уменьшения избыточного переобучения путём случайных занулений выходов нейронов в ходе построение модели [31].

Выход последнего полносвязного слоя является вектором с размерностью равной числу классов , который далее преобразуются функцией активации Softmax для нормализации полученных значений к вероятностям принадлежности к каждому из классов сетевых узлов. Softmax имеет вид [32]:

Полученный вектор с вероятностями принадлежности каждому классу является итоговым выходом модели. Индекс максимального значения в нём можно расценивать как метку того класса, которому соответствуют поступившие входные данные о сетевом трафике.

## 2.4 Стек используемых технологий

Для реализации интеллектуальной системы по обнаружению аномалий сперва необходимо определиться с набором используемых технологий. Это может включать в себя программы для работы с данными, такие как Python, R или MATLAB, библиотеки для машинного обучения, такие как TensorFlow, Keras или Scikit-learn. Здесь важно учитывать специфику решаемой задачи и возможные особенности анализируемых данных. Например, в случае работы с нейронными сетями может потребоваться нормализация числовых значений, а при явной неравномерности встречаемости классов – их балансировка. Выбранный стек технологий должен иметь методы для решения подобных задач.

Также необходимо определиться с программами-утилитами, которые будут готовить данные в удобную для анализа форму, поскольку различные устройства и программы могут использовать разные форматы записи трафика.

### 2.4.1 Python

Python – это высокоуровневый язык программирования, который был разработан в конце 1980-х годов Гвидо ван Россумом [33]. Он был создан для того, чтобы быть простым в использовании и чтении, поэтому синтаксис Python легко читается и понимается.

Python является интерпретируемым языком, что означает, что код выполняется построчно, без необходимости компиляции. Это делает Python очень удобным для быстрой разработки прототипов и экспериментов.

Одной из главных особенностей Python является его читаемость и простота написания кода. Этот язык также является кроссплатформенным языком программирования, что означает, что он может быть запущен на различных операционных системах, таких как Windows, Linux и macOS.

В целом, Python – это мощный и гибкий язык программирования, который может быть использован для создания широкого спектра приложений и проектов.

Современная версия языка Python3 выпущена в 2008 году и является более современной и улучшенной версией языка, которая имеет множество новых функций и возможностей.

Ещё Python имеет множество библиотек, которые делают его очень гибким и мощным языком программирования. Он широко используется в науке о данных, машинном обучении, веб-разработке, автоматизации задач и других областях.

#### 2.4.1.1 Scikit-learn

Scikit-learn – это библиотека машинного обучения для языка программирования Python [34]. Она предоставляет инструменты для решения задач классификации, регрессии, кластеризации и др. с использованием различных алгоритмов машинного обучения.

Scikit-learn имеет множество функций, таких как выборка и предобработка данных, выбор моделей, оценка моделей, настройка гиперпараметров и многое другое. Она также поддерживает множество алгоритмов машинного обучения, включая линейную регрессию, деревья решений, метод опорных векторов, случайный лес, градиентный бустинг и др.

Scikit-learn является одной из самых популярных библиотек машинного обучения в сообществе Python. Она имеет открытый исходный код и широко используется в научных исследованиях, промышленности и академических кругах. Scikit-learn также интегрируется с другими библиотеками Python, такими как NumPy, Pandas и Matplotlib, что делает её ещё более мощной и удобной в использовании.

#### 2.4.1.2 Keras

Keras – это высокоуровневый фреймворк для глубокого обучения, написанный на языке Python [35]. Он позволяет быстро создавать и обучать нейронные сети, включая свёрточные нейронные сети, рекуррентные нейронные сети и комбинации обоих. Keras предоставляет простой и интуитивно понятный интерфейс для создания моделей глубокого обучения, что делает его доступным для широкого круга пользователей, включая начинающих в области машинного обучения.

Keras был разработан с учётом скорости и гибкости, что позволяет легко экспериментировать с различными архитектурами нейронных сетей и параметрами обучения. Он также поддерживает множество библиотек для выполнения вычислений, включая TensorFlow, Theano и Microsoft Cognitive Toolkit.

#### 2.4.1.3 Tensorflow

TensorFlow – это библиотека открытого исходного кода для глубокого обучения, разработанная компанией Google Brain Team [36]. Она позволяет создавать, обучать и развёртывать нейронные сети для решения различных задач машинного обучения, включая классификацию, сегментацию, детекцию объектов и многое другое.

TensorFlow использует графы вычислений для представления моделей глубокого обучения, что позволяет оптимизировать вычисления и ускорить процесс обучения. Библиотека поддерживает множество языков программирования, включая Python, C++, Java и Go.

TensorFlow имеет множество функций для работы с данными, включая загрузку и предобработку данных, а также функции для визуализации результатов обучения. Он также поддерживает различные методы оптимизации, включая градиентный спуск, стохастический градиентный спуск и адаптивные методы оптимизации.

#### 2.4.1.4 Matplotlib и Plotly

Matplotlib и Plotly – это библиотеки визуализации данных для языка программирования Python.

Matplotlib предоставляет широкий спектр инструментов для создания различных видов графиков, диаграмм и других визуализаций данных [37]. Он поддерживает множество форматов вывода, включая PNG, PDF, SVG и другие. Matplotlib имеет простой и понятный синтаксис, что делает его доступным для начинающих пользователей.

Plotly, с другой стороны, является интерактивной библиотекой визуализации данных [38]. Он позволяет создавать интерактивные графики, которые можно изменять и настраивать в реальном времени. Plotly поддерживает множество типов графиков, включая диаграммы рассеяния, гистограммы, круговые диаграммы и другие. Он также имеет возможность создания 3D-графиков и карт.

#### 2.4.1.5 NumPy

NumPy – это библиотека для языка программирования Python, которая предоставляет поддержку многомерных массивов, матриц и высокоуровневых функций для работы с ними [39]. Она также предоставляет множество математических функций для работы с массивами, включая операции линейной алгебры, статистики, случайных чисел и другие.

NumPy является одной из основных библиотек для научных вычислений в Python и широко используется в различных областях. Она обладает высокой производительностью и эффективностью благодаря использованию оптимизированных алгоритмов и возможности интеграции с другими библиотеками для научных вычислений, такими как SciPy, Pandas и Matplotlib.

NumPy также имеет поддержку индексации и срезов массивов, что делает его удобным для работы с большими объёмами данных. Он также предоставляет множество функций для чтения и записи данных из различных форматов файлов, включая CSV, TXT и другие.

В целом, NumPy является важной библиотекой для научных вычислений в Python и позволяет удобно и эффективно работать с многомерными массивами и матрицами.

#### 2.4.1.6 Imblearn

Imbalanced-learn (Imblearn) – это библиотека для языка программирования Python, которая предоставляет поддержку для работы с несбалансированными наборами данных [40]. Она предоставляет множество методов и алгоритмов для балансировки выборки, включая методы для увеличения числа объектов минорного класса, уменьшения числа объектов мажорного класса, а также комбинированные методы.

Imblearn также предоставляет возможности для оценки качества моделей на несбалансированных данных, такие как метрики precision, recall, F1-score и ROC-AUC. Она также поддерживает использование различных алгоритмов классификации и регрессии, включая SVM, Naive Bayes, Decision Trees и другие.

Imblearn является полезной библиотекой для работы с несбалансированными наборами данных в различных областях и позволяет улучшить качество моделей на несбалансированных данных и повысить точность прогнозирования.

**2.4.1.7 XGBoost**

XGBoost (eXtreme Gradient Boosting) – это оптимизированная распределённая библиотека градиентного бустинга, разработанная для обеспечения высокой эффективности, гибкости и переносимости. Она реализует алгоритмы машинного обучения в рамках градиентного бустинга. XGBoost обеспечивает параллельный древовидный бустинг, который быстро и точно решает многие проблемы науки о данных [41].

Особенности XGBoost:

* Распараллеливание – модель реализуется для обучения на нескольких ядрах процессора.
* Регуляризация: XGBoost включает в себя различные регуляризации, чтобы избежать чрезмерной подгонки. Регуляризация со штрафами обеспечивает успешное обучение, поэтому модель может обобщать адекватно.
* Нелинейность: XGBoost может обнаруживать и обучаться на основе нелинейных моделей данных.
* Перекрёстная валидация: Встроенная и поставляется из коробки.
* Масштабируемость: XGBoost может работать распределено благодаря распределённым серверам и кластерам, таким как Hadoop и Spark, что позволяет обрабатывать огромные объёмы данных. Он также доступен для многих языков программирования, таких как C++, JAVA, Python и Julia.

**2.4.1.8 CatBoost**

CatBoost – это библиотека градиентного бустинга на деревьях решений, которая разработана исследователями и инженерами Яндекса и используется для поиска, рекомендательных систем, персональных помощников, самодвижущихся автомобилей, предсказания погоды и многих других задач [42].

Можно выделить следующие достоинства библиотеки CatBoost:

* Обработка категориальных признаков: CatBoost может автоматически обрабатывать категориальные признаки без необходимости их кодирования.
* Регуляризация: CatBoost поддерживает регуляризацию L2 для уменьшения переобучения.
* Автоматический подбор параметров: CatBoost может автоматически настраивать параметры модели с использованием кросс-валидации.
* Параллельное обучение: CatBoost может обучаться параллельно на нескольких ядрах процессора и на нескольких узлах в кластере.
* Поддержка GPU: CatBoost позволяет обучать свои модели на ядрах видеокарт.
* Интерпретируемость: CatBoost позволяет легко интерпретировать важность признаков, используемых в модели.
* Обработка пропущенных значений: CatBoost может обрабатывать данные с пропущенными значениями без необходимости заполнения или удаления строк.

### 2.4.2 Tcpdump и Libpcap

Tcpdump и Libpcap – это две связанные библиотеки, которые используются для анализа сетевых пакетов в операционных системах [43].

Libpcap является библиотекой, которая предоставляет низкоуровневый доступ к сетевым интерфейсам и позволяет захватывать пакеты, передаваемые по сети. Она обеспечивает возможность захвата пакетов с помощью различных фильтров, которые могут быть заданы пользователем, а также обеспечивает поддержку различных форматов файлов для хранения захваченных пакетов.

Tcpdump – это утилита командной строки, которая использует библиотеку Libpcap для захвата и анализа сетевых пакетов. Она позволяет задавать различные фильтры для захвата только определённых типов пакетов, а также может выводить информацию о захваченных пакетах в различных форматах.

Tcpdump и Libpcap широко используются в различных областях, включая сетевую безопасность, мониторинг сетевого трафика, отладку сетевых приложений и другие. Они предоставляют мощные инструменты для анализа сетевых пакетов и позволяют получать ценную информацию о работе сети и приложений, работающих в ней.

### 2.4.3 Nfdump

Nfdump – это утилита командной строки, которая используется для анализа и обработки данных о сетевом трафике, полученных из файлов формата NetFlow [44]. Она позволяет анализировать данные о трафике на основе различных критериев, таких как IP-адрес источника и назначения, порт, протокол и т. д.

Nfdump обеспечивает поддержку фильтров, которые позволяют выбирать только нужные данные о трафике, а также поддерживает вывод информации в различных форматах, включая CSV, XML и JSON. Кроме того, Nfdump может работать в режиме реального времени, что позволяет мониторить сетевой трафик в режиме онлайн.

Nfdump широко используется в области сетевого мониторинга и безопасности, а также для оптимизации сетевых приложений и управления сетевыми ресурсами. Она предоставляет мощный инструмент для анализа данных о сетевом трафике и позволяет получать ценную информацию о работе сети и приложений.

### 2.4.4 Clickhouse

Clickhouse – это колоночная база данных с открытым исходным кодом, которая разработана для обработки больших объёмов данных в режиме реального времени [45]. Она использует специальную архитектуру для обеспечения высокой производительности и эффективности при работе с большими объёмами данных.

Одной из ключевых особенностей Clickhouse является поддержка SQL-запросов, что делает её удобной для использования в различных приложениях и системах. Благодаря своей скорости и производительности, Clickhouse может обрабатывать миллиарды строк данных за считанные секунды.

Clickhouse также обладает рядом других функциональных возможностей, таких как репликация данных, шифрование, масштабирование и мониторинг. Эти возможности позволяют использовать Clickhouse для различных задач, включая аналитику данных, бизнес-анализ, мониторинг и машинное обучение.

В целом, Clickhouse представляет собой мощный инструмент для работы с большими объёмами данных, который может использоваться в различных областях, где требуется быстрый и эффективный анализ больших объёмов информации.

## 2.5 Описание программной разработки

Весь исходный код описанной программной разработки интеллектуальной системы представлен в репозитории проекта на Github, ссылка на который указана в приложении А. Решение программировалось в среде Jupyter Notebook – каждый из блоков кода в ноутбуке отвечает за свою задачу в рамках описанной ранее модульной структуры, и вывод результата работы любого блока представлялся непосредственно в ноутбуке в виде графического (гистограммы распределения узлов, важности признаков и т. д.) или же текстового представления (предсказания модели, оценки качества). Важно отметить, что решение Jupyter Notebook позволяет делать отложенный вызов блоков – пока не освободится выполнение процессором актуального блока, что автоматизирует работу системы.

### 2.5.1 Захват и преобразование трафика

Первоначальный захват трафика осуществляется с помощью утилиты Tcpdump. Используя встроенную библиотеку Libpcap, она конвертирует проходящий через неё трафик в файловый формат PCAP, который затем хранится на рабочем сервере.

На следующем шаге работы системы модуль подготовки данных производит преобразование имеющихся файлов PCAP в представление Netflow. Для этого вызывается специальная подпрограмма, в основе которой лежит утилита Nfpcapd, являющаяся частью более общего решения – Nfdump.

В ходе её работы производится отбор нужных для преобразования файлов PCAP путём сравнения их временных меток модификации, после чего выполняется преобразование в формат Netflow методом Nfpcapd – c уточнённым набором параметров запуска, в результате чего сетевой трафик нарезается на минутные интервалы и раскидывается по соответствующим файлам. Далее вызывается утилита Nfdump, которая представляет данные файлов Netflow в удобном для дальнейшего чтения виде – формате CSV. Итоговые файлы CSV используются для формирования результирующей таблицы БД Clickhouse.

Описание набора параметров сетевых потоков Netflow в представлении таблицы БД Clickhouse показано в таблице 2.1.

Таблица 2.1 – Состав таблицы Clickhouse с характеристиками сетевого трафика

| Название параметра | Тип данных | Описание |
| --- | --- | --- |
| id | UUID | Идентификатор записи |
| tsstart | UInt32 | Время регистрации первого пакета потока в секундах (Unix timestamp) |
| tsend | UInt32 | Время регистрации последнего пакета потока в секундах (Unix timestamp) |
| duration | Float64 | Длительность потока в секундах и миллисекундах |
| prot | UInt8 | Протокол (TCP, UDP, ...) |
| sip | String | IP-адрес источника |
| sport | UInt32 | Порт источника |
| dip | String | IP-адрес приёмника |
| dport | UInt32 | Порт приёмника |
| flags | String | TCP-флаги (CEUAPRSF – CWR, ECE, URG, ACK, PSH, RST, SYN, FIN) |
| pkts | UInt64 | Количество пакетов в потоке |
| bytes | UInt64 | Количество байт в потоке |
| bps | UInt64 | Среднее количество байт/сек в потоке |
| pps | UInt64 | Среднее количество пакетов/сек в потоке |
| bpp | UInt64 | Среднее количество байт/пакет в потоке |

### 2.5.2 Визуализация трафика

После заключения данных о сетевом трафике в табличном представлении его можно визуализировать посредством методов модуля визуализации трафика.

Для построения графиков с числом записей и встречаемости каждого узла в сети достаточно сделать запросы в таблицу Clickhouse на выдачу количества записей по каждому IP и полученную информацию использовать для отрисовки соответствующих гистограмм методами библиотеки Matplotlib языка Python.

Визуализация поведения сетевых узлов в виде графика параллельных координат выполняется с использованием библиотеки Plotly. При построении графиков из таблицы Clickhouse извлекаются данные о сетевых соединениях конкретного узла в рамках одного дня, которые затем фильтруются по протоколу сетевого уровня и агрегируются в пятиминутные промежутки – с целью уменьшения расходов на вычислительные ресурсы. Стоит добавить, что полученные графики параллельных координат обладают возможностью интерактивной настройки – изменения масштаба и расстояния между координатными столбцами.

### 2.5.3 Подготовка данных для ядра СОА

На следующем этапе работы системы модуль подготовки данных извлекает параметры сетевого трафика из таблицы БД Clickhouse путём запросов на выборку по временному промежутку и требуемому IP. При получении очередной порции данных проверяется отсутствие дубликатов, пропусков, а также производится бинаризация флагов TCP из строкового представления. Финальным шагом является агрегация по установленному временному окну.

Алгоритм агрегации для формирования записи устроен следующим образом:

1. Задаётся частота запуска расчёта – например, 1 минута.
2. Задаётся исторический интервал, в рамках которого формируется итоговая запись – например, 10 минут.
3. Задаётся количество промежутков исторического интервала.
4. Исторический интервал делится на промежутки по установленному количеству. Например, в случае настройки поделить 10 минутный интервал на 5 частей будут получены промежутки c минутами 1-2, 3-4, 5-6, 7-8, 9-10 соответственно.
5. Записи о соединениях размещаются по соответствующим временным промежуткам исторического интервала.
6. В рамках каждого промежутка происходит агрегация параметров – вычисление средних величин для скорости передачи пакетов и байт; cуммы для числа переданных или полученных байт и пакетов; числа уникальных портов, протоколов, или удалённых хостов.
7. Итоговый набор просчитанных параметров является единой записью для последующей обработки моделями.

Сформированная по представленному алгоритму запись охватывает больший временной промежуток, чем обычное соединение, тем самым заметно обобщая поведение сетевого узла.

**2.5.3.1 Вычисление производных параметров**

Описанная ранее запись с историческим окном имеет время конца, которое представляется в другом, отличном от Unix timestamp формате – циферблатном времени, вычисляемом по формуле (2.23):

где – число часов от начала текущего дня;

– число минут от начала текущего часа.

Циферблатное время указывает на косинус и синус углов часовой и минутной стрелок. Использование обоих тригонометрических функций необходимо с целью однозначного определения времени.Представление в циферблатном формате призвано сделать временные значения более правильными – периодическими, чтобы модели могли лучше запоминать и распознавать повторяющиеся с течением времени события.

Также к циферблатному времени добавляется другие временные параметры – день недели и время суток, которое может принимать следующие значения:

1. утро – от 6 до 12 часов;
2. день – от 12 до 16 часов;
3. вечер – от 16 до 24 часов;
4. ночь – от 24 до 6 часов.

На основе параметров Netflow происходит создание более сложных по смыслу агрегационных параметров – отбираются наиболее активные сервисы и все протоколы сетевого уровня, которые встречались, после чего по каждому из них может вычисляться:

1. среднее по продолжительности передачи данных, числу байтов в секунду, пакетов в секунду, байтов в пакете;
2. сумма по числу соединений, переданных и полученных байтов, пакетов;
3. количество различных портов, узлов приёма или получения данных.

Выбор этих параметров варьируется от результатов отбора по значимости, в случае же первоначального развёртывания системы потребуется просчёт всех из них. Также стоит отметить, что агрегационные параметры определяются в рамках своего промежутка исторического окна, а также с учётом входящих и выходящих соединений.

**2.5.3.2 Формирование выборки**

В зависимости от целей использования данных могут быть сформированы различные выборки – для первоначального обучения СОА стоит задача в создании сбалансированного набора данных за установленный период для передачи в модуль составления портрета, тогда как при реальной работе поступившие данные сразу могут быть обработаны модулем оценки, который выдаст заключение на обнаружение аномалии.

При создании обучающего набора данных сперва задаётся ограничение на максимально возможное число записей по каждому сетевому узлу в выборке – это первый шаг корректировки несбалансированности. На втором шаге осуществляется целевая выборка тех классов-IP, которые существенно преобладают над остальными. Происходит это по принципу квантиля – задаётся процентный порог от всего объёма данных, после чего выполняется последовательный отбор классов-IP со всеми соответствующими им записями, начиная с наиболее встречаемых классов, до того момента, когда доля набранных записей от общего объёма превосходит процентный порог. В рамках решения задачи значение квантиля составляло 95%. Оставшиеся классы-IP, которые не вошли в целевую выборку, приравниваются к классу неизвестный сетевой узел.

Последним шагом корректировки несбалансированности является использование технологии SMOTE. Synthetic Minority Over-sampling Technique основана на создании синтетических примеров для класса, который имеет меньшее количество образцов, чтобы тем самым уравновесить количество образцов в каждом классе. Алгоритм SMOTE работает следующим образом [46]:

1. выбирается случайный элемент из класса, который имеет меньшее количество образцов;
2. выбирается один из его ближайших соседей из того же класса;
3. создаётся новый элемент путём выбора случайной точки между выбранным элементом и его ближайшим соседом;
4. созданный новый элемент добавляется в набор данных.

Алгоритм SMOTE является эффективным способом решения проблемы несбалансированных классов в машинном обучении, но он также может привести к переобучению модели, если количество синтетических примеров слишком велико. Именно поэтому для использования данной технологии в модуле подготовки данных устанавливается критерий на минимальное число записей, при невыполнении которого класс становится кандидатом на обработку SMOTE.

Стоит добавить, что сформированный для обучения итоговый набор данных может быть представлен в формате Pickle для дальнейшего многократного использования. Pickle – это модуль в Python, который позволяет сериализовать, т. e. преобразовать объекты Python в байты и десериализовать – восстановить объект из байтов [47]. Формат Pickle используется для сохранения объектов Python в файлы или передачи их между процессами. Преимуществом хранения данных в Pickle является довольно быстрая скорость загрузки, а недостатком – большой размер файлов в сравнении с другими форматами представления, такими как CSV или JSON.

### 2.5.4 Ядро СОА

Как было отмечено ранее, ядро СОА состоит из модуля составления портрета и модуля оценки. Перед началом работы модуля оценки модуль составления портрета должен оценить сформированный набор параметров сетевого трафика на значимость. Для этого берётся легковесная модель случайного леса, которая подвергается многократному обучению на разных подмножествах признаков с задействованием технологии оптимизации RFECV. Recursive Feature Elimination with Cross-Validation – это метод автоматического отбора признаков, который позволяет определить наиболее значимые признаки для построения модели. Устройство работы RFECV следующее [48]:

1. На первом шаге все доступные признаки используются для построения модели.
2. Затем на каждой итерации наименее значимый признак удаляется, и модель перестраивается.
3. Для каждой модели вычисляется оценка качества с помощью кросс-валидации. Кросс-валидация используется для оценки качества модели на разных подмножествах данных. Обычно данные разбиваются на несколько частей – фолдов, и каждый фолд используется как тестовый набор данных для оценки качества модели, построенной на остальных фолдах. Это позволяет получить более объективную оценку качества модели и избежать переобучения.
4. На последнем шаге выбирается наилучшее подмножество признаков, которые дают наилучший результат.

RFECV позволяет уменьшить количество признаков и улучшить качество модели. Он также позволяет избежать переобучения, когда модель слишком сильно адаптируется к обучающим данным и не может обобщить полученные знания на новые данные.

После получения набора наиболее значимых признаков производится обучение итоговых моделей искусственного интеллекта. Реализация модели случайного леса взята из библиотеки Scikit-learn языка Python, модели градиентного бустинга – из библиотек XGBoost и CatBoost Программная реализация архитектуры нейронной сети с использованием методов Tensorflow представлена в репозитории проекта на Github.

При необходимости в ходе обучения моделей могут быть задействованы различные методы оптимизации по гиперпараметрам. Одним из таких методов является RandomizedSearchCV, который работает следующим образом: вместо перебора всех возможных комбинаций гиперпараметров, выбираются случайные значения из заданных диапазонов гиперпараметров. Затем производится обучение модели на этих случайных значениях и оценивается её производительность. Этот процесс повторяется несколько раз с разными случайными значениями гиперпараметров, и лучшая комбинация гиперпараметров выбирается как оптимальная.

Итоговая обученная наилучшая модель далее используется в модуле оценки для дальнейшей работы системы.

# **3 РЕЗУЛЬТАТЫ И ОЦЕНКА РАБОТЫ СИСТЕМЫ**

## 3.1 Разведочный анализ данных

Прежде чем приступать к обучению моделей и развёртыванию интеллектуальной системы, необходимо оценить данные, с которыми предстоит работать – попытаться увидеть в них общие закономерности, построить распределения составляющих параметров и т. д.

В качестве данных взят сетевой трафик предприятия продолжительностью 898 дней, период – с 24 июня 2020 года по 9 декабря 2022 года. Фрагмент используемого набора данных в формате Netflow представлен в приложении Б.

Распределение числа записей о соединениях по дням показано на рисунке 3.1. На нём видно, что в данных присутствуют незначительные пропуски, имеющие среднюю продолжительность около недели, причём некоторые из них приходятся на выходные и праздничные дни, когда никто из сотрудников не работал и сетевое оборудования было неактивно. Но в то же время существуют и такие пропуски, которые приходятся на рабочие дни, что может быть связано с периодическими ошибками и остановками работы модуля захвата трафика. Также в один из дней наблюдался значительный выброс по числу соединений.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 3.1 – График распределения числа записей по дням

Распределение числа записей о соединениях за всё время по каждому IP адресу показано на рисунке 3.2. Коэффициент встречаемости здесь определяется как логарифм по основанию 10 от суммарного числа соединений конкретного IP.

На данном графике можно увидеть довольно сильную несбалансированность различных классов сетевых узлов – только около ста из них присутствуют в количестве миллиона соединений и более, причём число соединений самого активного узла здесь многократно превышает остальные и составляет более ста миллионов.

Общее число сетевых устройств составило 1455, при этом больше половины из этих сетевых узлов имеет суммарное число соединений за всё время меньше ста, что довольно мало. Это объясняется многократным непродолжительным присутствием так называемых виртуальных машин, общая активность каждой из которых составляет не более дня.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 3.2 – График распределения записей по каждому IP

### 3.1.1 Анализ активности и поведения сетевых узлов

Для построения статистических графиков активности был взят трафик со следующих характерных узлов сети:

* Телекоммуникационное оборудование – сетевые устройства для объединения сегментов сети и пересылки сигналов.
* DNS-сервер – хранит и кэширует записи с информацией о IP-адресах сайтов.
* Веб-сервер – принимает запросы от клиентов, обычно веб-браузеров, и выдаёт ответы с содержанием веб-страниц, изображений, файлов, медиа-потоков и других данных.
* Почтовый сервер – принимает входящие электронные письма от клиентов и посылает исходящие.
* Сервер SVN – централизованная система управления версиями ПО, выдаёт исходный код программ и осуществляет его изменение при запросах.
* Рабочая станция – компьютер, за которым работает сотрудник.

В качестве визуализации поведения этих узлов использовалось несколько типов графиков.

График параллельных координат отражает особенность поведения конкретного узла сети в рамках одного дня – в какое время инициировались соединения с другими локальными и удалёнными хостами, к каким портам (сервисам) происходили обращения, сколько потоков было и т. д. Всё это отлично показывает, какие именно сервисы и службы текущий объект компьютерной сети использует наиболее интенсивно. При построении графиков параллельных координат по дням также можно увидеть, похоже ли в целом поведение одного узла в течение одной рабочей недели и наблюдается ли явное различие с другими узлами в сети.

Более формально, график параллельных координат состоит из столбцов, каждый из которых имеет свою систему величин, и построение связи между соседними столбцами происходит в случае, если регистрировались такие соединения, что их статистика соответствует данным значениям. В нашем случае рассматриваются следующие величины:

* время установки соединения узла с другими хостами,
* порт источника соединения,
* порт приёмника соединения,
* количество подключений источника на конкретный порт приёмника,
* количество уникальных хостов, к которым подключался текущий узел по конкретному порту.

Примеры графиков параллельных координат представлены в приложении В для двух рабочих станций, активных в одни и те же дни. На них видно, что первая станция ведёт себя весьма схожим образом – период активности длится целый день, при этом число потоков по каждой службе может составлять от 1 до 4 с таким же числом уникальных узлов в каждом. Рабочая станция №2 тоже ведёт себя почти одинаково, имеются лишь незначительные различия в числе потоков по каждой службе. При сравнении активности этих сетевых устройств можно заключить, что у них наблюдается разная интенсивность трафика по потокам, но довольно схожее время работы и используемые сервисы.

В ходе анализа всех графиков параллельных координат был сделан вывод, что каждый из сетевых узлов ведёт себя примерно одинаково с течением времени, и в рамках одного класса устройств можно увидеть некоторые различия во времени активности, используемым портам, числу потоков в службах и взаимодействующих хостов. Стоит также добавить, что разные классы устройств по поведению различаются довольно заметно. Например, у серверов SVN наблюдалась строго фиксированная по времени установка одного соединения в день по протоколу UDP, в то время как DNS сервера активны почти всегда и выполняют все выходные соединения по единому порту 53 независимо от протокола транспортного уровня.

Следующие типы графиков, которые были построены в ходе разведочного анализа, демонстрируют интенсивность трафика по каждому узлу сети – суммарный объём переданных и полученных данных за фиксированный промежуток времени. Это ещё одна важная характеристика, которая позволит выявить сходства и различия в поведении устройств.

Примеры графиков продемонстрированы в приложении Г. На них можно увидеть, как для некоторых рабочих станций интенсивность их трафика имеет почти одинаковые периоды скачков, но разные числовые показатели для общих дней недели. В свою очередь интенсивность трафика серверов, например, с сервисом DNS, довольно детерминирована и почти не зависит от дня недели. Также есть и устройства по типу ТКО, интенсивность трафика которых весьма произвольна.

Общий объём выходящего и входящего трафика за двухнедельный период по дням представлен на графиках в приложении Д. Здесь можно увидеть, что объём трафика далеко не всегда держится на одном уровне, особенно это заметно у некоторых рабочих станций и SVN-сервера.

Последний тип графиков, отражающий число уникальных хостов, к которым сетевой узел обращался за две недели, показан в приложении Е. На нём прослеживается явная детерминированность значения для ТКО, относительно небольшой разброс для DNS-сервера и совершенно разные показатели для рабочих станций, что довольно ожидаемо.

В репозитории проекта на Github представлен полный набор построенных статистических графиков.

### 3.1.2 Заключения разведочного анализа

В ходе разведочного анализа было обнаружено, что в используемом наборе данных о сетевом трафике существуют незначительные пропуски, а также явный выброс в один из дней. Это нужно иметь в виду при создании обучающей выборки и не брать эти дни в расчёты.

Стоит также добавить, что число сетевых устройств в трафике заметно велико, а количество их соединений неравномерно распределено, что требует учёта и принятия мер по балансировке при дальнейшем создании тренировочного набора данных.

Анализ статистических графиков поведения устройств показал, что большинство классов сетевых узлов с течением времени ведут себя одинаково по определённым параметрам и подлежат возможному разделению между собой.

## 3.2 Обучение моделей

Для обучения моделей искусственного интеллекта была сформирована выборка, которая имела верхнее ограничение на число записей по каждому сетевому узлу равное ста тысячам за промежуток 23.06.2023 – 23.09.2023. График распределения полученного числа записей на каждый класс после применения подхода отбора по квантилю представлен на рисунке 3.3.

Изображение выглядит как снимок экрана, Прямоугольник, шаблон, текст

Автоматически созданное описание

Рисунок 3.3 – График распределения записей по каждому IP после отбора   
по квантилю

Таким образом, имеется 135 классов наиболее активных устройств, для которых был проведён отбор значимых признаков и последующая балансировка методом SMOTE с дальнейшим обучением моделей искусственного интеллекта. Полученные результаты точности классификации сетевых узлов на тестовой выборке представлены в таблице 3.1.

Таблица 3.1 – Точности распознавания поведения сетевых узлов на тестовой выборке

| Название модели | Первоначальная точность | Точность после оптимизации по гиперпараметрам |
| --- | --- | --- |
| Случайный лес | 45% | 57% |
| CatBoost | 68% | 84% |
| XGBoost | 77% | 90% |
| Многослойная ИНС с прямой связью | 91% | 91% |

В качестве метрики оценки качества здесь используется сбалансированная точность, которая вычисляется как среднее арифметическое между точностью и полнотой для каждого класса. Точность – это доля правильно классифицированных объектов положительного класса относительно всех объектов, которые были отнесены к этому классу. Полнота – это доля правильно классифицированных объектов положительного класса относительно всех объектов, которые принадлежат к этому классу.

В случае несбалансированных классов, когда один класс имеет гораздо большую выборку, чем другой, точность и полнота могут быть искажены. Сбалансированная точность позволяет учесть эту неравномерность и даёт более объективную оценку качества классификации. Значение сбалансированной точности может находиться в диапазоне от 0 до 100 %, где 100 % означает идеальную классификацию, а 0 % – случайное угадывание. Чем ближе значение к максимуму, тем лучше качество классификации.

Можно сделать вывод, что модель градиентного бустинга XGBoost и многослойной ИНС с прямой связью являются наиболее эффективными для решения задачи распознавания сетевых узлов, в то время как модель случайного леса и градиентного бустинга CatBoost показали себя далеко не лучшим образом, и их точность распознавания ниже удовлетворимого уровня.

## 3.3 Оценка качества моделей

В качестве оценки эффективности выбранного подхода формирования агрегационных записей о сетевых соединениях было проведено обучение моделей на стандартных полях данных представления Netflow. Результаты проделанного эксперимента показаны в таблице 3.2.

Таблица 3.2 – Точности распознавания поведения сетевых узлов на стандартных полях данных представления Netflow

| Название модели | Первоначальная точность | Точность после оптимизации по гиперпараметрам |
| --- | --- | --- |
| Случайный лес | 14% | 38% |
| CatBoost | 33% | 37% |
| XGBoost | 37% | 50% |
| Многослойная ИНС с прямой связью | 41% | 52% |

Таким образом, предложенный метод группировки записей является довольно важным для точности предсказаний.

Для того, чтобы показать значимость качественного результата предсказания выбранных моделей ИИ, было проведено сравнение со следующими простыми способами решения данной задачи:

* Случайный бейзлайн – хаотичное угадывание результата.
* Простые эвристики, т. е. использование всевозможных простых правил. Например, запоминание всех задействованных узлом протоколов, портов и его средней интенсивности приёма передачи трафика, и выполнение предсказаний посредством проверки попадания поступивших значений в сформированные промежутки.
* Выбор класса по частоте встречаемости – вероятность предсказания сетевого узла зависит от процентной доли его записей относительно всей выборки.

Полученные точности предсказаний с использованием вышеуказанных подходов представлены в таблице 3.3, тут можно заключить, что классификационная задача весьма сложная и не поддаётся тривиальному решению.

Таблица 3.3 – Точности распознавания поведения сетевых узлов с использованием элементарных подходов

| Название модели | Точность распознавания |
| --- | --- |
| Случайный бейзлайн | 5% |
| Простые эвристики | 8% |
| Выбор класса по частоте встречаемости | 7% |

Далее, было проведено сравнение с наиболее распространёнными методами машинного обучения, способными решить данную задачу, в их стандартном представлении без специфической настройки по параметрам. Заключение оценок этих алгоритмов представлено в таблице 3.4. Полученные точности распознавания указывают на явную неэффективность этих моделей решать поставленную задачу.

Таблица 3.4 – Точности распознавания поведения сетевых узлов распространёнными классификационными моделями

| Название модели | Точность распознавания |
| --- | --- |
| Логистическая регрессия | 35% |
| Решающее дерево | 19% |
| Метод ближайшего соседа | 22% |
| Метод опорных векторов | 54% |
| Байесовский классификатор | 39% |

Для оценки качества обнаружения аномалий в имеющийся трафик были намеренно добавлены различные сетевые угрозы: DoS-, DDoS-атаки, деятельность червей, отключения и другие. Результаты классификации полученного трафика на аномальный и неаномальный представлены в таблице 3.5.

Таблица 3.5 – Точности детектирования аномалий при различных сетевых атаках

| Название модели | Случайный лес | CatBoost | XGBoost | Многослойная ИНС с прямой связью |
| --- | --- | --- | --- | --- |
| Тестовая выборка | 21816 | | | |
| Число обнаруженных атак (всего 14527) | 7389 | 10754 | 13684 | 13932 |
| Количество записей, оценённых как неаномальный (всего 7289) | 4325 | 5252 | 6438 | 6987 |

Таким образом, модели градиентного бустинга XGBoost и многослойной ИНС с также показывают хороший результат в разграничении аномального и неаномального сетевого трафика.

## 3.4 Предложения по дальнейшим улучшениям

В ходе проверки качества работы моделей градиентного бустинга не были использованы другие реализации алгоритма, такие как AdaBoost, LightGBM и другие. Также возможно усложнение текущей используемой архитектуры многослойной ИНС с прямой связью.

Добавление большего набора производных признаков – ещё одно предложение по дальнейшим улучшениям интеллектуальной системы, поскольку ещё не было оценено качество предсказания с такими важными параметрами как:

* категории доменов, к которым обращался сетевой узел;
* статистика запросов по каждому протоколу прикладного уровня;
* полнота использования рабочего времени **–** значение сумматора, в котором считается количество секунд, в которых наблюдалась активность за день.

Стоит отметить, что интеллектуальная система обнаруживает отклонения в сетевом трафике независимо от реальных угроз, она не застрахована от ложноположительных срабатываний. Например, если несколько пользователей локальной сети поменяются компьютерами и продолжат вести себя как обычно, но под другими IP адресами, СОА может расценить поведения каждого из них как аномальное.

В связи с этим возникает необходимость осуществления проверок на сетевые атаки после факта обнаружения аномалий. Для этого предлагается разработать в ядре СОА модели второго порядка, каждая из которых в случае обнаружения аномалии отвечала бы за детектирования конкретного типа сетевых угроз, например червей, DoS- и DDoS-атак и. т. д. Предполагается, что они будут работать заметно медленнее модели обнаружения аномалий из-за обработки более специфического набора параметров, и поэтому использовать такие модели постоянно на сетевом трафике нецелесообразно – только при факте обнаружения отклонения.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения выпускной квалификационной работы бакалавра были получены следующие результаты:

1. Рассмотрены и классифицированы существующие методы обнаружения аномалий сетевого трафика, описаны преимущества и недостатки каждой группы подходов.
2. Проведены анализ и категоризация наиболее распространённых сетевых аномалий.
3. Более подробно рассмотрены методы обнаружения аномалий сетевого трафика на основе моделей искусственного интеллекта.
4. Построена и реализована архитектура интеллектуальной системы по обнаружению аномалий.
5. Определён набор входных параметров для моделей, предложен новый подход в формировании записей о сетевых соединениях.
6. Произведено успешное тестирование работы системы на выучивании поведения сетевых узлов, а также выполнена проверка на обнаружение аномального трафика с различными типами атак.

Результаты проделанной работы показывают, что разработанная архитектура интеллектуальной системы является довольно эффективной и покрывает возникающие потребности эксплуатации. В ходе тестирования моделей искусственного интеллекта было выяснено, что бустинг на деревьях XGBoost и многослойная ИНС могут эффективнее всего обнаруживать аномалии сетевого трафика.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Qosmos ENEA. Network Detection and Response Report [Доклад] // Cybersecurity INSIDERS. – 2021. – P. 25-28.

2. ЦСС-Безопасность. Программный комплекс ретроспективного анализа сетевого трафика «Стетоскоп» версии 3.0 / ЦСС-Безопасность. – 2022. – C. 2-3.

3. Lucas, M.W. Network flow analysis / M.W. Lucas. – No Starch Press, 2010. – P. 9-11.

4. Евгеньева, А.П. Словарь русского языка в четырёх томах / А.П. Евгеньева // РАН, Институт лингвистических исследований. – Изд. 4-е, стереотип. – М.: Рус. яз.; Полиграфресурсы, 1999.

5. Микова, С.Ю. Подход к классификации аномалий сетевого трафика / С.Ю. Микова, В.С. Оладько, М.А. Нестеренко // Международный научный журнал «Инновационная наука». – 2015. – Вып. 11. – C. 78-80.

6. Monowar, H.B. Network Traffic Anomaly Detection and Prevention. Concepts, Techniques, and Tools / Monovar H. Bhuyan, Dhruba. K. Bhattacharyya, Jugal K. Kalita // Springer International Publishing, 2017. – P. 39-42.

7. Добкач, Л.Я. Анализ методов распознавания компьютерных атак / Л.Я. Добкач // Правовая информатика. – 2020. – Вып. 1. – C. 67-74.

8. Методы обнаружения аномалий и вторжений / E.В. Ананьин , И.C. Кожевникова, А.В. Лысенко, А.В. Никишова // Проблемы современной науки и образования. –2016. – Вып. 34 (76). – С. 48-50.

9. Добкач, Л.Я. Анализ эффективности систем обнаружения вторжений // Труды Всеросс. студ. конф. «Студенческая научная весна», посвящённая 165-летию со дня рождения В. Г. Шухова» / Росмолодежь, МГТУ им. Н.Э. Баумана, СНТО им. Н.Е. Жуковского. – М. : Изд. дом «Научная библиотека», 2018. – С. 314-315.

10. Браницкий, А.А. Анализ и классификация методов обнаружения сетевых атак / А.А. Браницкий, И.В. Котенко // Труды СПИИРАН. – 2016. – № 2 (45). – С. 207-244.

11. Микова, С.Ю. Результат исследования алгоритмов выявления сетевых аномалий / C.Ю. Микова, В.C. Оладько // Вопросы кибербезопасности. – 2015. – Вып. 4(12). – С. 38-41.

12. Амосов, О.С. Вейвлет-алгоритмы оценивания нестационарных процессов с фрактальной структурой, имеющих неоднородности и нарушения /   
О.C. Амосов, C.Г. Баена // Информатика и системы управления. – 2017. – №2 (52). – С. 85-99.

13. Басараб, М.А. Обнаружение аномалий в информационных процессах на основе мультифрактального анализа / М.А. Басараб, И.C. Строганов // Вопросы кибербезопасности. – 2014. – № 4 (7). – С. 30-40.

14. Конин, А.В. Кластеризация статистических характеристик сетевого трафика для обнаружения аномалий. Информационные технологии в науке, промышленности и образовании / А.В. Конин, А.В. Коробейников // Сборник трудов региональной научно-технической очно-заочной конференции. – Ижевск: Изд-во Ижевского гос. техн. ун-та им. М.Т. Калашникова, 2016. – С. 32-39.

15. Zhang, R., Zhang, S., Muthuraman, S., Jiang, J. One Class Support Vector Machine for Anomaly Detection in the Communication Network Performance Data // 5th WSEAS Int. Conference on Applied Electromagnetics, Wireless and Optical Communications, Tenerife. – 14-16 Dec., 2007.

16. Flow-Based Anomaly Intrusion Detection System Using Two Neural Network Stages / Y. Abuadlla, G. Kvascev, S. Gajin, Z. Jovanovic // Computer Science and Information Systems. – 2014. – № 11 (2). – P. 601-622.

17. Vrushali, D.M. Anomaly based IDS using Backpropagation Neural Network / D.M. Vrushali, S.N. Pawar // International Journal of Computer Applications. – 2016. -–136 (10).

18. Барсегян, А.А. Анализ данных и процессов: учеб. Пособие / А.А. Барсегян, М.С. Куприянов, И.И. Холод, М.Д. Тесс, С.И. Елизаров. – СПб.: БХВ-Петербург, 2009. – 512 с.

19. Шелухин, О.И. Обнаружение аномальных состояний компьютерных систем средствами интеллектуального анализа данных системных журналов / О.И. Шелухин, В.С. Рябинин, М.А Фармаковский // Вопросы кибербезопасности. – 2018. – № 2 (26). – С. 33-43.

20. Батурин, Д.С. Анализ методов обнаружения атак в информационных сетях / Д.С. Батурин // Вестник АмГУ. – 2019. – Вып. 87. – С. 54-59.

21. LAN Abnormalities Threat Detection: An Outlook and Applicability Analysis / A.M. Modorskiy, A.S. Minzov, O.R. Baronov, A.Y. Nevskiy // Вопросы кибербезопасности. – 2018. – № 1(25). – С. 11-18.

22. NetFlow Export Datagram Format // Cisco [Электронный ресурс]. – URL: <https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html\> (дата обращения 19.03.2023).

23. Breiman, L. Random Forests / L. Breiman // Machine Learning. – 2001. – № 45. – P. 5-32.

24. Breiman, L., Friedman, J., Olshen R., Stone, C. Classification and Regression Trees. – Wadsworth, Belmont, CA. – 1984.

25. Efron, B. Bootstrap Methods: Another Look at the Jackknife. – Institute of Mathematical Statistics. – 1979.

26. Breiman, L. Bagging predictors / L. Breiman // Machine Learning. – 1995. – №24. – P. 123-140.

27. Prokhorenkova L., Gusev G., Vorobev A., Dorogush V., Gulin A. CatBoost: unbiased boosting with categorical features. – 2017.

28. Tianqi Chen, Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. – 2016.

29. D. Svozil, V. Kvasnicka, Jiri Pospíchal. Introduction to multi-layer feed-forward neural networks. – Chemometrics and Intelligent Laboratory Systems. – 1997. – P. 44-49.

30. Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. – 2015.

31. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov // Journal of Machine Learning Research. – 2015. – № 15. – P. 1929-1958.

32. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron. Deep Learning. – MIT Press, 2016. – P. 180-184.

33. History and License – Python 3.11.3 documentation [Электронный ресурс]. – URL: <https://docs.python.org/3/license.html> (дата обращения 13.03.2023).

34. Scikit-learn: machine learning in Python – scikit-learn 1.2.2 documentation [Электронный ресурс]. – URL: <https://scikit-learn.org/stable/> (дата обращения 12.03.2023).

35. Keras: Deep Learning for humans [Электронный ресурс]. – URL: <https://keras.io/> (дата обращения 12.03.2023).

36. TensorFlow [Электронный ресурс]. – URL: <https://www.tensorflow.org/> (дата обращения 12.03.2023).

37. Matplotlib – Visualization with Python [Электронный ресурс]. – URL: <https://matplotlib.org/> (дата обращения 12.03.2023).

38. Plotly Python Graphing Library [Электронный ресурс]. – URL: <https://plotly.com/python/> (дата обращения 12.03.2023).

39. NumPy [Электронный ресурс]. – URL: <https://numpy.org/> (дата обращения 12.03.2023).

40. Imbalanced-learn documentation – Version 0.10.1 [Электронный ресурс]. – URL: <https://imbalanced-learn.org/stable/> (дата обращения 12.03.2023).

41. XGBoost Documentation – xgboost 1.7.5 documentation [Электронный ресурс]. – URL: <https://xgboost.readthedocs.io/en/stable/> (дата обращения 12.03.2023).

42. CatBoost – open-source gradient boosting library [Электронный ресурс]. – URL: <https://catboost.ai/> (дата обращения 12.03.2023).

43. Tcpdump & libpcap [Электронный ресурс]. – URL: <https://www.tcpdump.org/> (дата обращения 12.03.2023).

44. Nfdump documentation // Nfdump [Электронный ресурс]. – URL: <https://nfdump.sourceforge.net/> (дата обращения 12.03.2023).

45. Fast Open-Source OLAP DBMS – ClickHouse [Электронный ресурс]. – URL: <https://clickhouse.com/> (дата обращения 12.03.2023).

46. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P. SMOTE: synthetic minority over-sampling technique // Journal of artificial intelligence research. – 2002. – №16. – P. 321-357.

47. Pickle – Python object serialization – Python 3.11.3 documentation [Электронный ресурс]. – URL: <https://docs.python.org/3/library/pickle.html> (дата обращения 16.03.2023).

48. Gene selection for cancer classification using support vector machines / I. Guyon, J. Weston, S. Barnhill, V. Vapnik // Machine. Learning. – 2002. – 46(1-3). – P. 389-422.

# **ПРИЛОЖЕНИЕ А QR-код ссылки на репозиторий проекта в Github**

Изображение выглядит как шаблон, Симметрия, прямоугольный, искусство

Автоматически созданное описание

Рисунок А.1 – QR-код ссылки на репозиторий проекта в Github

# **ПРИЛОЖЕНИЕ Б QR-код ссылки на фрагмент используемого набора данных**

Изображение выглядит как шаблон, прямоугольный, искусство, Симметрия

Автоматически созданное описание

Рисунок Б.1 – QR-код ссылки на фрагмент используемого набора данных

# **ПРИЛОЖЕНИЕ В Пример графиков параллельных координат активности узлов сети**

Изображение выглядит как линия, диаграмма, текст, Параллельный

Автоматически созданное описание

Рисунок В.1 – График активности по выходящим соединениям рабочей станции №1 за 2022-04-19

Изображение выглядит как линия, диаграмма, График, Параллельный

Автоматически созданное описание

Рисунок В.2 – График активности по выходящим соединениям рабочей станции №1 за 2022-04-20

Изображение выглядит как линия, диаграмма, снимок экрана, Параллельный

Автоматически созданное описание

Рисунок В.3 – График активности по выходящим соединениям рабочей станции №2 за 2022-04-19

Изображение выглядит как линия, диаграмма, текст, Параллельный

Автоматически созданное описание

Рисунок В.4 – График активности по выходящим соединениям рабочей станции №2 за 2022-04-20

# **ПРИЛОЖЕНИЕ Г Пример графиков интенсивности трафика узлов сети**

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок Г.1 – График интенсивности трафика рабочей станции по пятницам за двухнедельный период

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок Г.2 – График интенсивности трафика DNS-сервера по понедельникам за двухнедельный период

Рисунок Г.3 – График интенсивности трафика DNS-сервера по пятницам за двухнедельный период

Изображение выглядит как диаграмма

Автоматически созданное описание

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок Г.4 – График интенсивности трафика ТКО по понедельникам за двухнедельный период

# **ПРИЛОЖЕНИЕ Д Пример графиков суммарного объёма трафика узлов сети**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок Д.1 – График суммарного входного объёма трафика DNS-сервера за промежуток 2020/09/01-2020/09/14

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Рисунок Д.2 – График суммарного входного объёма трафика SVN-сервера за промежуток 2020/09/01-2020/09/14

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Рисунок Д.3 – График суммарного выходного объёма трафика рабочей станции №1 за промежуток 2020/09/01-2020/09/14

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок Д.4 – График суммарного выходного объёма трафика рабочей станции №2 за промежуток 2020/09/01-2020/09/14

# **ПРИЛОЖЕНИЕ Е Пример графиков числа уникальных взаимодействующих хостов узлов сети**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок Е.1 – График числа уникальных взаимодействующих хостов ТКО за промежуток 2020/09/01-2020/09/14

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок Е.2 – График числа уникальных взаимодействующих хостов DNS-сервера за промежуток 2020/09/01-2020/09/14

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Рисунок Е.3 – График числа уникальных взаимодействующих хостов рабочей станции №1 за промежуток 2020/09/01-2020/09/14

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок Е.4 – График числа уникальных взаимодействующих хостов рабочей станции №2 за промежуток 2020/09/01-2020/09/14