МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени
М. В. ЛОМОНОСОВА
Специализированный учебно-научный центр
(школа-интернат имени А. Н. Колмогорова)

Анализ сетевого трафика для выявления киберугроз с применением методов машинного обучения

Описание проекта

Работу выполнил:
ученик 11В класса
Бабич Егор

Москва, 2025

Постановка задачи

Цель проекта

Разработка системы для обнаружения кибератак на основе анализа сетевого трафика с использованием методов машинного обучения.

Задачи проекта

- Изучить типы кибератак и особенности сетевого трафика;
- Подготовить и обработать датасеты;
- Выбрать, обучить и протестировать модель машинного обучения;
- Реализовать веб-приложение для анализа трафика и визуализации результатов;
- Провести тестирование и анализ эффективности системы;

Обоснование актуальности

При росте количества кибератак (DDoS, фишинг, вредоносное ПО) традиционные системы обнаружения угроз, как правило, основаны на сигнатурных методах и не используют машинное обучение. Это ограничивает их способность выявлять новые или модифицированные типы атак. В отличие от них, предлагаемое веб-приложение использует методы машинного обучения для анализа сетевого трафика и обнаружения аномалий. Такой подход повышает адаптивность и эффективность системы в условиях постоянно изменяющейся структуры угроз, обеспечивая более высокий уровень защиты.

Сбор и анализ информации по проблеме

В качестве источника данных использовался расширенный датасет *KDD Cup 1999* (версии `KDDTrain+` и `KDDTest+`), содержащий 148 517 записей различных типов сетевой активности. Каждая запись представляет собой вектор из 42 признаков, описывающих свойства соединения: числовые, булевы и категориальные признаки, такие как `duration` (продолжительность соединения), `protocol_type` (тип протокола), `src_bytes`, `dst_bytes`, `flag` (статус флага соединения), `count` (количество соединений к тому же хосту за последние 2 секунды) и другие. Также в датасете присутствует целевой признак `attack`, обозначающий тип сетевой активности (нормальной или вредоносной).

Для упрощения и ускорения обработки были отобраны только наиболее информативные признаки: `protocol_type`, `service`, `flag`, `src_bytes`, `dst_bytes`, `count`, `srv_count`, `dst_host_count`, `dst_host_srv_count`.

Категориальные признаки (`protocol_type`, `service`, `flag`) были преобразованы в числовую форму с помощью кодирования метками (`Label Encoding`), что позволило сохранить информацию о взаимосвязях между уровнями признаков без увеличения размерности, как в случае с `One-Hot Encoding`.

Также была произведена бинаризация целевой переменной: все виды атак были сведены к одному классу, обозначающему наличие вторжения, а нормальные соединения — к другому. Итоговая переменная `is_attack` принимала значения 0 (норма) или 1 (атака).

Для визуального анализа и диагностики данных применялись библиотеки `Seaborn` и `Matplotlib`, что позволило выявить особенности распределения признаков и частоты атак. Данные были разделены на обучающую и тестовую выборки в соотношении 80:20 с сохранением пропорции классов (`stratify=y`).

Таким образом, проведённая предварительная обработка позволила адаптировать сложный и многомерный набор данных для обучения моделей машинного обучения, минимизируя потери информации и обеспечивая высокую эффективность анализа.

Разработка идеи и концепции

Разработка системы началась с анализа существующих методов обнаружения киберугроз и выявления их недостатков. В процессе выбора оптимальной архитектуры было принято решение о создании интегрированного решения, которое сочетает в себе три ключевых компонента: модуль захвата и фильтрации трафика, обученную модель машинного обучения, а также удобный веб-интерфейс. Такой подход позволяет не только эффективно анализировать сетевой трафик в реальном времени, но и обеспечивает простоту взаимодействия пользователя с системой через интуитивно понятную визуализацию данных.

Выбор модели машинного обучения для детектирования аномалий в сетевом трафике был основан на эмпирических экспериментах с различными алгоритмами. На начальном этапе были протестированы несколько популярных методов, таких как Decision Tree, KNN, Logistic Regression, SGDClassifier, Naive Bayes и SVM. Все алгоритмы были обучены на реальных данных, и их результаты сравнивались по таким критериям, как точность, скорость обучения и способность к выявлению скрытых закономерностей в данных.

На основе проведённых тестов, лучшими показателями для решения поставленной задачи обладал алгоритм **Random Forest**. Этот алгоритм оказался наиболее устойчивым к переобучению и эффективно справлялся с большими объёмами данных, что делает его идеальным для анализа сложных и многомерных характеристик сетевого трафика. Его способность объединять несколько деревьев решений в ансамбль позволила повысить точность классификации и уменьшить количество ложных срабатываний. Таким образом, выбор Random Forest в качестве модели для системы был логичным и оправданным.

Для обучения модели были заданы следующие ключевые параметры:

- **Размер леса** (`n_estimators=100`) — ансамбль включал 100 деревьев решений, что обеспечило баланс между точностью модели и скоростью её работы;
- **Максимальная глубина дерева** (`max_depth=15`) — ограничение глубины позволило снизить риск переобучения и ускорить обучение;
- **Взвешивание классов** (`class_weight='balanced'`) — использовалось автоматическое балансирование классов, что особенно важно при работе с несбалансированными данными, типичными для задач обнаружения атак;
- **Фиксированное начальное состояние генератора случайных чисел** (`random_state=42`) — обеспечивало воспроизводимость результатов.

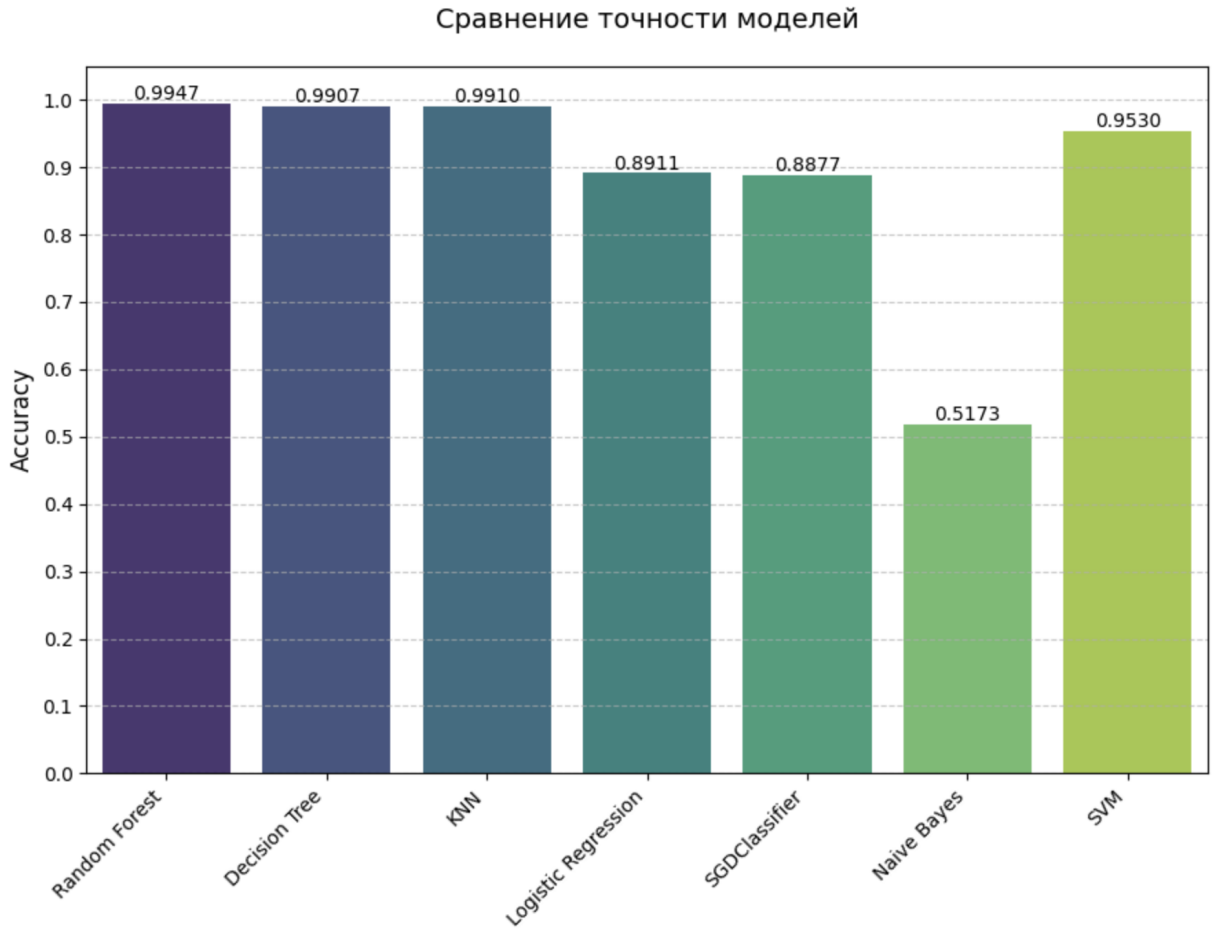

Рис. 1: Сравнение точности различных моделей машинного обучения (Accuracy).

Эти параметры позволили добиться высокой точности при классификации сетевого трафика, сохранив при этом устойчивость модели к переобучению и способность обрабатывать сложные зависимые признаки.

Интерфейс системы был спроектирован с учётом потребностей конечного пользователя. Графики и уведомления, реализованные на веб-странице, позволяют пользователю оперативно отслеживать состояние сети и реагировать на возможные инциденты, что значительно упрощает процесс мониторинга и повышает безопасность информационных систем.

Техническое задание

1. Основное назначение разрабатываемого объекта

Разработка веб-приложения для обнаружения кибератак на основе анализа сетевого трафика с использованием методов машинного обучения. Система должна анализировать входящий сетевой трафик в реальном времени, выявлять аномалии и отображать результаты с помощью визуальных элементов.

2. Технические характеристики и показатели качества

- Система должна обеспечивать анализ сетевого трафика с возможностью обнаружения аномальных паттернов.
- Используемый алгоритм машинного обучения должен демонстрировать точность классификации не менее 80%.
- Веб-интерфейс должен включать визуализацию данных, таких как графики и таблицы для отображения текущего состояния безопасности.
- Система должна быть способна выявлять и оповещать о возможных кибератаках в реальном времени.

3. Технико-экономические требования

- Для запуска системы требуется установка интерпретатора Python версии 3.8 или выше и установка зависимостей, указанных в файле `requirements.txt`.
- Приложение разворачивается локально на устройстве пользователя и запускается через веб-браузер с поддержкой HTML5 и JavaScript.
- Система не требует специализированного оборудования и может быть запущена на персональном компьютере с процессором от 2.0 ГГц, 4 ГБ оперативной памяти и не менее 500 МБ свободного дискового пространства.
- Используются свободно распространяемые и кроссплатформенные технологии, что исключает необходимость приобретения дополнительного программного обеспечения.

Используемые технологии и алгоритмы

Проект реализован с использованием современных инструментов веб-разработки, анализа сетевого трафика и машинного обучения.

Технологии

- **Flask** — микрофреймворк для создания серверной части веб-приложения;
- **SQLite** — встроенная реляционная база данных для хранения информации о трафике;
- **PyShark** (на основе Wireshark) и **tcpdump** — библиотеки для захвата и анализа сетевого трафика;
- **Scikit-learn** — библиотека машинного обучения для обучения и использования классификаторов;
- **Pandas** и **NumPy** — библиотеки для обработки и анализа данных;
- **Chart.js** — JavaScript-библиотека для визуализации данных (графики, диаграммы);
- **Bootstrap 5** — CSS-фреймворк для адаптивной вёрстки интерфейса;
- **Font Awesome** — библиотека иконок для интерфейса.

Алгоритмы и методы

- **Классификация трафика** — используется модель машинного обучения, обученная на основе одного из алгоритмов (Random Forest, SVM, Decision Tree и др.) для определения, является ли трафик вредоносным;
- **Предобработка данных** — кодирование категориальных признаков (Label Encoding), нормализация значений;
- **Анализ трафика** — извлечение признаков из сетевых пакетов (IP, порты, протоколы, флаги, размеры), подсчёт агрегированных характеристик;
- **Визуализация** — отображение статистики с помощью диаграмм (распределение протоколов, сервисов и т.д.).

Особенности реализации

- Многопоточность — параллельный захват трафика и работа веб-интерфейса;
- REST API — взаимодействие между клиентской и серверной частью;
- Асинхронное обновление данных — использование AJAX для отображения информации в реальном времени;
- Обработка в реальном времени — данные анализируются и отображаются с минимальной задержкой.

Руководство пользователя

Руководство программиста

Результаты работы

В рамках проекта реализована полноценная система для анализа и классификации сетевого трафика в реальном времени. Она состоит из трёх ключевых компонентов: инструмента захвата трафика (tcpdump и PyShark), backend-сервиса на Flask для обработки и анализа данных, и frontend-интерфейса для визуализации результатов.

Приложение позволяет:

- производить захват сетевых пакетов с локального интерфейса;
- извлекать ключевые признаки из пакетов (IP, порты, протоколы, флаги и др.);
- применять модель машинного обучения для автоматического определения, является ли трафик аномальным;
- отображать статистику и предупреждения пользователю в удобной форме.

Была реализована система визуализации, включающая круговые и столбчатые диаграммы на основе Chart.js, а также динамическую таблицу с обновлением данных через AJAX. Пользовательский интерфейс адаптивен и прост в использовании, построен с использованием Bootstrap 5.

Для классификации сетевого трафика были протестированы следующие алгоритмы машинного обучения:

- **Random Forest** — ансамблевый метод на основе решающих деревьев, устойчивый к переобучению и шумам в данных
- **Decision Tree** — интерпретируемая модель, строящая правила классификации через последовательное разбиение данных

- **KNN** (k-ближайших соседей) — непараметрический алгоритм, классифицирующий объекты по majority vote среди k ближайших примеров
- **SVM** (Support Vector Machine) — ищет оптимальную разделяющую гиперплоскость с максимальным зазором между классами
- **Logistic Regression** — линейная модель с сигмоидной функцией, предсказывающая вероятности принадлежности к классам
- **SGDClassifier** — стохастический градиентный спуск, эффективный для больших данных
- **Naive Bayes** — вероятностный классификатор, основанный на теореме Байеса с предположением о независимости признаков

Модели сравнивались по метрикам Accuracy, Precision, Recall и F1-Score. Лучший результат показал алгоритм **Random Forest**:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Random Forest	**0.9947**	**0.9961**	**0.9929**	**0.9945**
KNN	0.9910	0.9901	0.9913	0.9907
Decision Tree	0.9907	0.9948	0.9857	0.9903
SVM	0.9530	0.9861	0.9152	0.9493
Logistic Regression	0.8911	0.9212	0.8461	0.8821
SGDClassifier	0.8877	0.9319	0.8271	0.8764
Naive Bayes	0.5173	0.2078	0.0011	0.0022

Таблица 1: Сравнение моделей по основным метрикам

Таким образом, итоговая система сочетает надёжный алгоритм машинного обучения с практическими инструментами сетевого мониторинга и визуального анализа, обеспечивая высокую точность обнаружения угроз при минимальной задержке обработки данных.

Вывод

Реализация проекта продемонстрировала высокую эффективность применения методов машинного обучения в задачах информационной безопасности. В условиях постоянно возрастающего объёма сетевого трафика и усложнения атак веб-приложение, разработанное в рамках проекта, позволяет оперативно выявлять потенциальные угрозы без необходимости ручного анализа.

Использование Random Forest в качестве классификатора обеспечило надёжную детекцию различных типов атак благодаря высокой точности и способности выявлять скрытые закономерности. Интеграция с библиотеками захвата и анализа трафика (tcpdump, PyShark), удобный веб-интерфейс и визуализация данных сделали систему доступной для конечного пользователя и пригодной для использования в реальных условиях.

Проект может быть расширен за счёт добавления новых алгоритмов, увеличения объёма обучающих данных или подключения к распределённым источникам трафика. Таким образом, он представляет собой фундамент для создания более универ-

сальных и масштабируемых систем кибербезопасности на основе интеллектуального анализа данных.

Список используемой литературы

1. Apply machine learning techniques to detect malicious network traffic in cloud computing, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00475-1.

2. Метод опорных векторов (SVM). Подходы, принцип работы и реализация с нуля на Python, https://habr.com/ru/articles/802185/

3. Support Vector Machines (SVM) from scratch, https://www.kaggle.com/code/egazakharenko/support-vector-machines-svm-from-scratch

4. Support Vector Machines, https://scikit-learn.org/stable/modules/svm.html

5. Михайлов А.Н. ОБНАРУЖЕНИЕ АНОМАЛИЙ В СЕТЕВОМ ТРАФИКЕ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ, https://www.вестник-науки.рф/article/19907

6. OWASP Top Ten Project, https://owasp.org/www-project-top-ten/

7. OWASP TOP-10: практический взгляд на безопасность веб-приложений, https://habr.com/ru/companies/simplepay/articles/258499/

8. KDD Cup 1999 Dataset, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.