Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

Лабораторная работа №3 (вариант №12)

Тема: «Проектирование архитектуры хранилища больших данных»

Дисциплина «Инструменты для хранения и обработки больших данных»

Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
Студентка группы АДЭУ-221
Сабитова А.Б.

Преподаватель:
Босенко Т.М.

Москва
2025

Задача: создать архитектуру хранилища больших данных для компании, занимающейся анализом сетевого трафика и логов для выявления угроз в реальном времени (SEIM), а также поведенческим анализом для обнаружения аномалий (UEBA).

Цель: обеспечить надежное хранение, эффективную обработку и анализ больших объемов данных из различных источников, таких как NetFlow, PCAP, системные логи и прочие.

1. Определение требований для компании в сфере кибербезопасности

1.1. Объем данных

— Ожидаемый объем: 100-150 ТБ.

— Рост: 50-60% ежегодно.

1.2. Скорость получения данных

— NetFlow: обновление каждые 1-5 минут.

— PCAP: захват сетевого трафика идет непрерывно с задержками в пределах нескольких секунд + триггерный захват PCAP при обнаружении подозрительного события, начинается немедленно и длится 10 минут.

— Системные логи: в режиме реального времени, до 30 секунд.

— Данные из Honeypots: передача данных в случае попытки атаки в режиме реального времени, до 1 минуты.

1.3. Типы данных

— Структурированные: системные логи, NetFlow записи (20%).

— Полуструктурированные: логи с различным форматированием, JSON, XML (50%).

— Неструктурированные: PCAP-файлы, данные из honeypots, текстовые логи (30%).

1.4. Требования к обработке

— Анализ сетевого трафика: ежечасно.

— Обработка логов: непрерывно.

— Анализ подозрительных файлов: по требованию.

— Анализ угроз и оценка рисков: еженедельно.

— Анализ эффективности защитных систем: ежемесячно.

— Отчетность для руководства: еженедельно.

1.5. Доступность данных

— Время обнаружения угрозы: 1-5 минут.

— Доступность системы: 99,9% (время простоя в год: не более 8 часов 45 минут).

1.6. Безопасность данных

— Шифрование данных в состоянии покоя, при передаче и хранении.

— Контроль доступа с многоуровневой аутентификацией и аудитом действий пользователей.

— Обеспечение целостности данных и предотвращение несанкционированного изменения.

— Соответствие корпоративным требованиям по защите информации и персональных данных и 152-ФЗ "О персональных данных".

2. Архитектура хранилища больших данных.

2.1. Источники данных

— NetFlow

— PCAP

— Системные логи

— Данные из honeypots.

2.2. Слой сбора данных

— Apache Kafka для потоковых данных. Выбран вместо других брокеров (например, RabbitMQ), так как он лучше всего справляется с обработкой больших потоков данных в реальном времени, обеспечивает масштабируемость и отказоустойчивость,

а также отлично интегрируется с инструментами аналитики и машинного обучения.

— GoFlow2 для сбора NetFlow. Необходим для того, чтобы принимать NetFlow-потоки, конвертировать их в JSON и отправлять в Kafka. Выбран, так как обладает наиболее высокой скоростью обработки и поддерживает прямую интеграцию с Kafka.

— Vector для сбора логов и данных из honeypots. Выбран Vector вместо Logstash, так как обладает более высокой производительностью, обеспечивает минимальную потерю данных и использует меньше ресурсов и памяти, что очень важно при работе с большими объемами логов и honeypot-событий.

— Zeek для захвата сетевого траффика. Выбран, потому что в сравнении с аналогами (Suricata, Snort) он не просто захватывает пакеты, а анализирует сетевой траффик и превращает его в структурированные логи, с помощью которых легко искать аномалии и выявлять атаки.

2.3. Слой хранения данных

— MinIO для хранения сырых и архивных данных. Выбран вместо HDFS, потому что он обеспечивает более высокую производительность при работе большими данными, а также способен с высокой скоростью обрабатывать множество мелких файлов (например, логи, PCAP), что важно для компании в сфере кибербезопасности.

— ClickHouse для хранения актуальных данных (30-90 дней) и оперативной аналитики, отчетности, а также для поиска по структурированным логам. Выбран, потому что в отличие от аналогов обеспечивает очень высокую скорость обработки аналитических запросов на больших объемах данных, что важно для оперативного анализа киберугроз.

— PostgreSQL – для оперативного хранения результатов анализа и метаданных. Выбран, потому что он обеспечивает надежное транзакционное хранилище, гибкость работы с большими объемами данных и идеально подходит как промежуточное хранилище для валидации и агрегации результатов перед их загрузкой в ClickHouse, а также для работы с метаданными.

2.4. Слой обработки данных

— Apache Flink для потоковой обработки в реальном времени. Выбран вместо Spark Streaming, так как требуется высокопроизводительная обработка больших потоков данных с миллисекундной задержкой.

— Apache Spark для пакетной обработки. Выбран из-за его высокой скорости обработки больших объемов исторических данных, что позволяет проводить глубокий анализ логов. Кроме того, отлично интегрируется с ClickHouse и Kafka.

2.5. Слой аналитики и машинного обучения

— Jupyter Notebook для анализа данных, а также разработки и тестирования моделей машинного обучения. Выбран, так как имеет множество готовых расширений и библиотек и отлично интегрируется с Spark и Kafka. В отличие от облачных решений Jupyter позволяет организовать работу в полностью изолированной среде, что важно для безопасности.

— Apache Superset для визуализации. Выбран Apache Superset вместо Tableau или Power BI, потому что лучше справляется с подключением к нестандартным источникам данных (ClickHouse, Kafka, Spark).

2.6. Слой управления данными

— Apache Atlas для управления метаданными. Выбран, потому что в отличие от аналогов (DataHub, Collibra) интегрируется с многочисленными системами обработки данных (Hadoop, Kafka,

Spark и др.), позволяя автоматически собирать и отслеживать метаданные из разных источников.

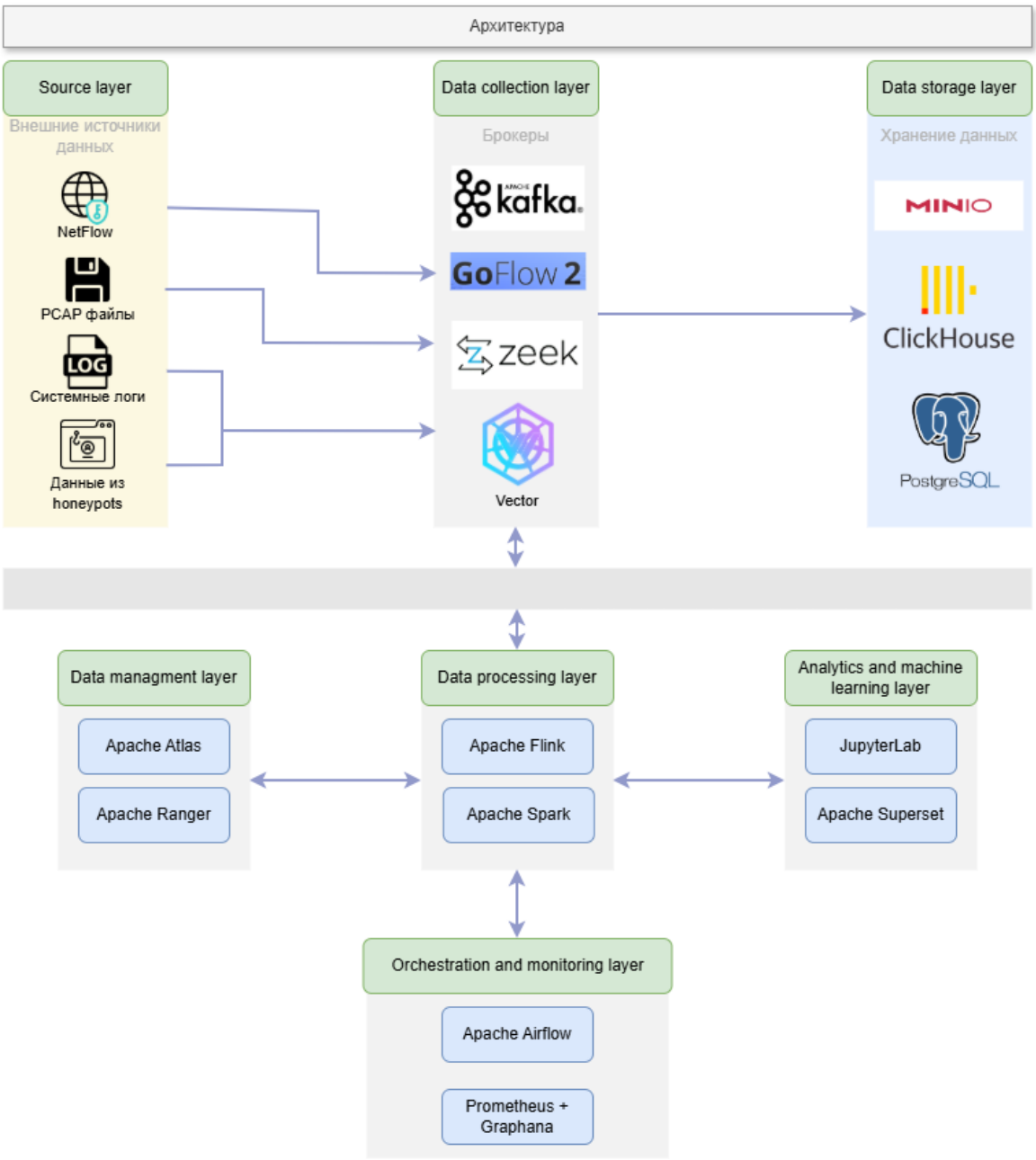
— Apache Ranger для управления доступом и безопасностью. Выбран, потому что это единственная система, которая обеспечивает централизованное управление доступом ко всем компонентам платформы безопасности через единые политики. Для сравнения, в Apache Sentry можно управлять доступом только в рамках экосистемы Hadoop.

2.7. Слой оркестрации и мониторинга

— Apache Airflow для оркестрации рабочих процессов. Выбран, так как обладает наиболее развитой системой мониторинга, логирования и оповещений, что необходимо для контроля и аудита потоков данных и результатов выполнения в безопасности.

— Prometheus и Grafana. Выбраны вместо Zabbix, так как Prometheus более эффективен для сбора и мониторинга метрик контейнеризированных систем (Kubernets, Kafka, Flink), а Grafana предлагает более современные, гибкие и настраиваемые дашборды с поддержкой множества источников данных.

3. Схема архитектуры



4. Процесс обработки данных

1. Данные собираются из различных источников через слой сбора данных и поступают в Apache Kafka.
2. Сырые данные сохраняются в MinIO для долгосрочного хранения, где они предварительно сжимаются для минимизации объема занимаемого пространства и избежания быстрого заполнения хранилищ.

3. Потоковые данные обрабатываются в реальном времени с помощью Apache Flink для оперативного выявления аномалий и быстрого реагирования на инциденты безопасности.
4. Пакетные задачи, такие как анализ исторических атак, корреляция событий безопасности и расширенная аналитика, выполняются с помощью Apache Spark по расписанию.
5. Результаты анализа и обработанные актуальные данные сначала сохраняются в PostgreSQL для проверки и валидации, а затем поступают в ClickHouse для быстрого доступа, оперативной аналитики, построения отчетов и поиска по логам.
6. Аналитики используют JupyterLab и Apache Superset для исследования данных, визуализации и построения интерактивных отчетов.
7. Модели машинного обучения разрабатываются и тестируются в Jupyter, развертываются в реальном времени через потоковую обработку или пакетно с помощью Apache Airflow.
8. Управление метаданными и безопасностью обеспечивается инструментами Apache Atlas и Apache Ranger соответственно.
9. Оркестрация процессов выполняется через Apache Airflow, а мониторинг и визуализация метрик – через Prometheus и Grafana.

5. Масштабирование и отказоустойчивость

— Использование кластерной архитектуры Kafka, MinIO, ClickHouse, Spark для горизонтального масштабирования и обработки больших объемов данных.
— Репликация данных в MinIO для обеспечения отказоустойчивости и надежного хранения.
— Потоковая обработка в реальном времени через кластер Apache Flink с поддержкой восстановления после сбоев.
— Применение Prometheus и Grafana для мониторинга состояния кластеров и оперативного реагирования на сбои.

6. Безопасность

— Реализация шифрования объектов на стороне сервера (SSE) ля защиты данных в MinIO, что обеспечивает автоматическое шифрование данных при записи и расшифрование при чтении, предотвращая утечку при хранении и снижая риски компрометации данных.
— Шифрование всех межсервисных соединений (MinIO, Kafka, ClickHouse, Spark) через TLS для предотвращения перехвата трафика
— Применение Apache Ranger для детального контроля доступа, политики безопасности и разграничения прав пользователей.
— Использование Kerberos для аутентификации пользователей
— Регулярное резервное копирование данных из MinIO, а также настройка сценариев аварийного восстановления.

7. Потенциальные проблемы и пути их решения

1. Сложность управления потоками данных и нагрузкой на Kafka и Flink

Описание:

- источники генерируют данные с разной скоростью и объемом, что ведет к неравномерному поступлению событий в Kafka. Пиковые нагрузки или всплески активности могут перегружать брокеры Kafka и вызывать задержки или потерю сообщений.
- Apache Flink выполняет потоковую обработку, требующую своевременного получения и обработки данных из Kafka. При перегрузках задержки на входе и внутри Flink могут приводить к накапливанию очередей и снижению качества аналитики.
- Kafka и Flink имеют ограниченный объем вычислительных ресурсов, поэтому необходимо эффективное управление нагрузкой для избежания сбоев.

Пути решения:

- Разделение потоков данных по типам, приоритетам или категориям в отдельные топики Kafka.
- Постоянное отслеживание ключевых метрик Kafka и Flink (задержки, пропускная способность и т.п.) с помощью Prometheus

и Grafana для своевременного выявления узких мест и внесения корректировок.

- Добавление новых брокеров в Kafka и новых вычислительных узлов в Flink при увеличении нагрузки.

2. Сложность обеспечения качества данных

Описание:

- Данные из источников приходят в разных форматах и с разной скоростью, отсутствует валидация и контроль качества на входе, это может привести к пропускам, дублированию, временной несогласованности, и соответственно, ошибкам в аналитике и при обучении моделей машинного обучения

Пути решения:

- Реализация промежуточного слоя – Apache Flink + Great Expectations, который будет проверять форматы, диапазоны, пропуски и дубли перед записью в MinIO
- Использование Kafka Schema Registry для контроля форматов и совместимости данных
- Отслеживание метрик качества (процент валидных записей, дубли, null-поля) в Prometheus и Grafana