Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Научно-образовательный центр «Высшая ИТ школа»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

АВТОМАТИЗАЦИЯ ПРОЦЕССА ЛОГ-МЕНЕДЖМЕНТА В ПРОЕКТЕ SOC.

Шамов Егор Сергеевич

Направление подготовки 09.03.04 Программная инженерия

Направленность (профиль) «Программная инженерия»

Научный руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ким Константин Станиславович, доцент, учебный офис

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ г.

Научный консультант

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зоркин Александр Сергеевич, ведущий инженер, ООО "ТЦР"

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ г.

Автор работы

студент группы № 972103

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. С. Шамов

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ г.

Оглавление

[1 Введение в предметную область 7](#_Toc195898864)

[1.1 Инфраструктура SIEM 7](#_Toc195898865)

[1.1.1 Источники логов 7](#_Toc195898866)

[1.1.2 Коллекторы 7](#_Toc195898867)

[1.1.3 Apache Kafka 8](#_Toc195898868)

[1.1.4 Парсеры (нормализаторы) 8](#_Toc195898869)

[1.1.5 Система хранения логов 9](#_Toc195898870)

[1.1.6 Коррелятор 9](#_Toc195898871)

[1.2 Настройка приема логов из источника 10](#_Toc195898872)

[1.2.1 (Поместить в конец) Времязатраты на создание коллектора 11](#_Toc195898873)

[2 Анализ существующих процессов 12](#_Toc195898874)

[2.1 Процесс настройки сбора событий 12](#_Toc195898875)

[2.1.1 Создание коллектора 12](#_Toc195898876)

[2.1.2 Прямая интеграция с Kafka 12](#_Toc195898877)

[2.2 Процесс настройки нормализации событий 12](#_Toc195898878)

[2.2.1 Создание парсера 12](#_Toc195898879)

[2.2.2 Написание правил нормализации 12](#_Toc195898880)

[3 Автоматизация процесса 13](#_Toc195898881)

[3.1 Избавление от людей в процессе 13](#_Toc195898882)

[3.1.1 Настройка параметров топика 13](#_Toc195898883)

[3.1.2 Конфигурация балансировщика 13](#_Toc195898884)

[3.1.3 Заключение 14](#_Toc195898885)

[3.2 Настройка автоматизации 14](#_Toc195898886)

[3.2.1 Предлагаемая структура проекта для инвентаризации источников. 14](#_Toc195898887)

[3.2.2 Структура файла «main.yml» 14](#_Toc195898888)

[3.2.3 Описание процесса развертывания 16](#_Toc195898889)

[3.2.4 Автоматизация поиска свободного порта 16](#_Toc195898890)

[3.2.5 Автоматизация создания топика 16](#_Toc195898891)

[3.2.6 Создание топика в собственном кластере 16](#_Toc195898892)

[3.2.7 Создание топика в кластере KaaS решения 16](#_Toc195898893)

[3.3 Автоматизация создания коллектора 17](#_Toc195898894)

[3.3.1 Требуемые параметры для запуска 17](#_Toc195898895)

[3.3.2 Генерация конфигурационного файла 17](#_Toc195898896)

[3.3.3 Развертывание сервиса 17](#_Toc195898897)

[3.3.4 Предоставление информации о развернутом экземпляре пользователю 17](#_Toc195898898)

[3.4 Настройка CI/CD пайплайнов в репозитории 17](#_Toc195898899)

[3.4.1 Проверка корректности конфигурации 17](#_Toc195898900)

[4 Автоматизация создания парсера 18](#_Toc195898901)

[4.1.1 Генерация конфигурационных файлов из шаблона 18](#_Toc195898902)

[4.1.2 Механизм развертывания парсера 18](#_Toc195898903)

Глоссарий

**Инцидент информационной безопасности** – это подтверждённое событие или группа событий, указывающие на нарушение политики безопасности, компрометацию системы или другую угрозу, требующую расследования и реагирования.

**Аудит безопасности** — это процесс систематической (часто автоматизированной) проверки и оценки состояния информационной безопасности системы или ее компонента с целью выявления уязвимостей, несоответствий требованиям и рисков, а также выработки рекомендаций по их устранению.

Введение

Security Operations Center (SOC) – это специализированное подразделение, которое осуществляет мониторинг и реагирование на инциденты информационной безопасности в режиме реального времени.

Вся работа SOC построена на анализе данных с различных компонентов наблюдаемых систем. Внутри SOC любое событие из какой-либо наблюдаемой системы мы называем логом. **Лог (или событие)** — это небольшая запись, которую система или приложение делает при наступлении каких-то значимых ситуаций. Например: вход пользователя в систему, ошибка в приложении, инициация нового сетевого соединения и т.д. В контексте информационной безопасности (далее ИБ) такие логи помогают понять, что происходило в системе и выявить подозрительную активность или инциденты.

Для сбора, хранения и управления логами в SOC используется система управления информацией и событиями безопасности (англ. Security Information and Event Management System, **SIEM**). Помимо перечисленного, в задачи SIEM входит обогащение логов дополнительной информацией из других источников, а также последующий корреляционный анализ, который позволяет выявлять сложные цепочки взаимосвязанных событий, указывающих на потенциальные инциденты ИБ.

Поскольку SIEM-система обрабатывает десятки, а иногда и сотни тысяч событий в секунду, в её архитектуре нередко используется промежуточный слой — шина или буфер обмена данными, например Apache Kafka. Такая шина позволяет эффективно распределять и балансировать потоки логов, обеспечивая равномерную доставку событий между компонентами системы. Также Kafka играет ключевую роль в обеспечении отказоустойчивости, масштабируемости и надежности обработки данных: она временно хранит события, предотвращая их потерю при высоких нагрузках или сбоях отдельных компонентов SIEM, и позволяет обрабатывать события асинхронно, но почти в режиме реального времени.

Не все источники событий могут напрямую отправлять данные в Kafka - особенно это касается устаревших систем или решений, не поддерживающих современные протоколы интеграции. Для таких случаев используется промежуточный компонент - коллектор. Коллектор принимает логи от источников по различным протоколам и передаёт далее в Kafka, никак не видоизменяя исходное сообщение. Такой подход обеспечивает обратную совместимость с существующей инфраструктурой SIEM.

В последнее время из-за быстрого роста компании каждый месяц появляется один или два новых источника, уже сейчас количество источников исчисляется десятками. В связи с этим замечены две проблемы.

1. Создание коллектора в текущий момент занимает до двух суток времени, и на протяжении этого времени инженер, ответственный за создание коллектора, должен держать контекст в голове.
2. Инвентаризация источников логов ведется вручную параллельно процессу настройки приема событий из источника.

Указанные проблемы обусловлены недостаточным уровнем автоматизации существующих процессов. Цель данной работы – разработать и внедрить автоматизированный процесс настройки приема логов из различных источников, включающий в себя развёртывание и конфигурирование коллекторов логов и других вспомогательных элементов.

Для достижения этой цели в работе необходимо решить следующие задачи.

1. Проанализировать существующий процесс развертывания коллекторов, выявить операции, подлежащие автоматизации.
2. Спроектировать и разработать инструменты автоматизации, которые позволят исключить или минимизировать ручные операции
3. Внедрить разработанное решение в инфраструктуру SOC и провести оценку его эффективности.

# Анализ существующего процесса

Прежде чем перейти к анализу процесса, необходимо погрузиться в предметную область, чему посвящена глава 1.1.

## Введение в предметную область

Вся работа SOC построена вокруг двух систем – SIEM и SOAR. Первая система (SIEM) анализирует логи и создает **алерты** - уведомления о потенциальных инцидентах безопасности, требующих проверки или реагирования. Вторая (SOAR) используется для работы с инцидентами.

**SOAR** (англ. Security Orchestration, Automation, and Response) – система оркестрации, автоматизации и реагирования на инциденты ИБ. В SOAR работает команда реагирования на инциденты ИБ (англ. CSIRT, Computer Security Incident Response Team). Члены этой команды занимаются анализом алертов, их эскалацией в инциденты, расследованием инцидентов и написанием отчетов об инцидентах. Все эти действия выполняются частично или полностью через SOAR. Этой информации о SOAR достаточно для понимания контекста.

Работу SIEM разберем подробнее далее. На рисунке 1 отражены основные сущности SIEM и их взаимодействие с SOAR.

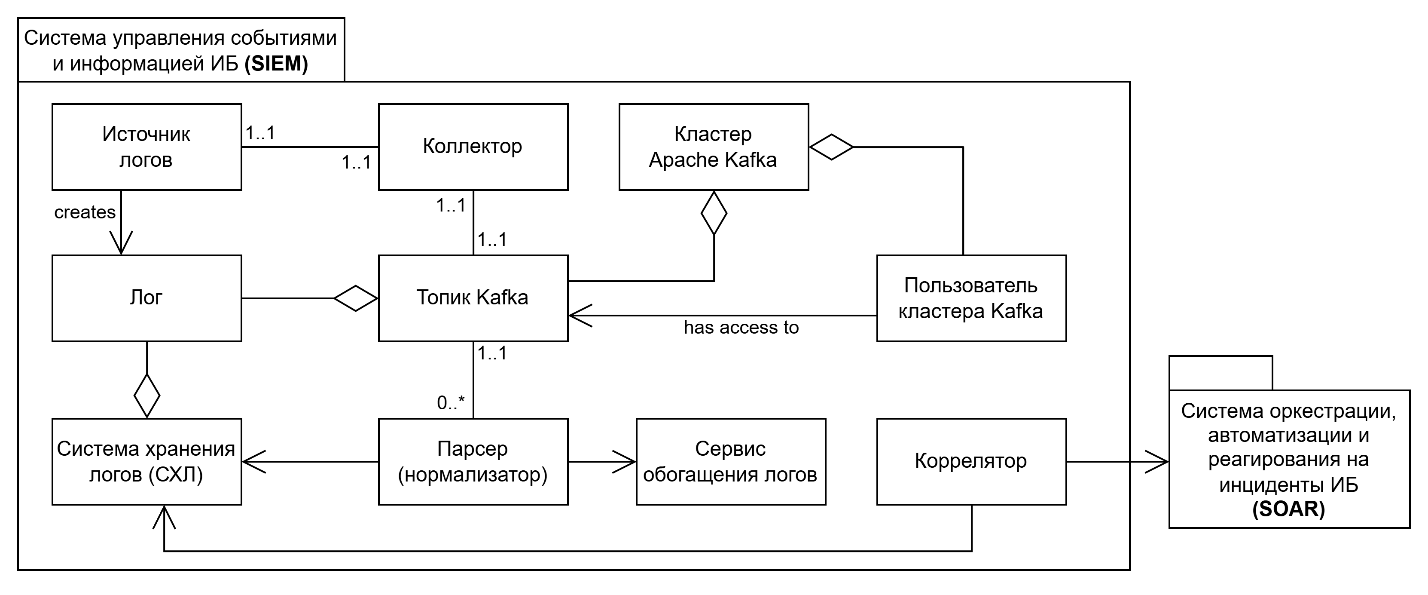


Рисунок 1 – Предметная область SIEM

### Источники логов

К источникам логов могут относиться разные системы, но чаще это сетевое оборудование, антивирусное ПО, приложения для аудита безопасности операционных систем, системы управления учетными данными (англ. Identity and Access Management System, IdM/IAM) и другие сервисы, которыми пользуется большое количество сотрудников (хоть и не всегда явно).

Как было сказано ранее, не всегда источники логов умеют передавать данные через современные протоколы взаимодействия. Например, большое количество серверного оборудования предлагает экспорт логов только через UDP (англ. User Datagram Protocol, сетевой протокол транспортного уровня). А некоторые приложения (например, система управления проектами Jira) сохраняют данные аудита только в базу данных (БД), в таком случае, единственный путь получения данных – читать их прямо из БД. Для таких случаев необходим какой-то сервис-адаптер, преобразующий один формат данных в другой. Таким сервисом выступает коллектор.

### Коллекторы

**Коллектор** — это сервис, который принимает события от различных источников по разным протоколам и интерфейсам, и передаёт их в стандартизированном виде в систему Apache Kafka. Основная задача коллектора заключается в обеспечении совместимости устаревших или ограниченных в плане интеграции систем с современной инфраструктурой SIEM. При этом коллектор не изменяет смысл или содержимое события, а только адаптирует транспортный и форматный уровень передачи данных, что позволяет централизованно и унифицированно обрабатывать логи в SIEM-системе.

Коллектор должен уметь пропускать через себя большой поток данных в единицу времени, оставаясь при этом легковесным и нересурсозатратным. Среди инструментов с открытым исходным кодом можно выделить следующие инструменты, подходящие на роль коллектора:

Filebeat;

FluentD;

Vector.dev.

Сюда что-то стоит дописать?

### Apache Kafka

**Apache Kafka** — это распределённая платформа потоковой передачи данных, предназначенная для обработки и передачи больших объёмов сообщений в режиме реального времени. В контексте (SIEM) Kafka играет ключевую роль в обеспечении надёжной и масштабируемой доставки логов от коллекторов к следующим компонентам SIEM – парсерам.

Ключевой сущностью Kafka является топик – логический канал, в который публикуются сообщения. Например, можно создать отдельные топики для различных типов логов: сетевых событий, логов какого-то приложения, событий на серверах операционной системы Linux. Каждый топик может быть разделён на несколько партиций – физических разделов топика - для обеспечения параллельной обработки и масштабируемости.

Пользователи в Apache Kafka представляют собой сущности, которые взаимодействуют с кластером, выполняя операции чтения, записи и администрирования. Обычно, пользователи в Kafka разграничены по ролям, и пользователей, которые читают данные из топика (или топиков) называют «консьюмер» (англ. consumer), а пользователей, которые пишут данные в топик (или топики) называют «продьюсер» (англ. producer). Но существование пользователей с правами, как на чтение, так и на запись в один или множество топиков также допустимо.

### Парсеры (нормализаторы)

**Парсер, или нормализатор логов,** — это компонент, который получает сырые события (логи) из Kafka, анализирует их структуру и формат, и преобразует в унифицированный вид, понятный и пригодный для обработки системой SIEM. Поскольку источники логов могут существенно различаться по структуре, набору полей и формату, парсер выполняет роль преобразователя, устраняющего эти различия.

Например, в логах сетевого оборудования может содержаться информация о портах и IP-адресах в одном формате, тогда как в логах приложений — о действиях пользователей и кодах ошибок в совершенно другом виде. Парсер приводит такие события к общему формату, добавляя необходимые поля, стандартизируя метки времени, уровни критичности и другие характеристики события.

Дополнительно парсер может:

выполнять фильтрацию событий (например, исключать неинтересные типы логов);

обогащать события предварительной технической информацией (время обработки, геолокация по IP и т.д.);

маркировать события тегами или классификацией для упрощения анализа.

Таким образом, парсер обеспечивает стандартизацию входящих логов и играет ключевую роль в качестве "точки входа" в аналитическую часть SIEM-системы.

### Система хранения логов

**Система хранения логов (СХЛ)** — это компонент SIEM, предназначенный для долговременного хранения нормализованных, обогащённых и проанализированных логов. Такие системы позволяют безопасно и эффективно сохранять большие объёмы данных с возможностью последующего поиска, фильтрации и аналитики.

В качестве СХЛ могут использоваться как коммерческие решения, так и open-source инструменты. Одним из наиболее популярных решений с открытым исходным кодом является Elasticsearch. Он обеспечивает полнотекстовый поиск, гибкую фильтрацию и масштабируемое хранение данных.

Логи в СХЛ сохраняются в обогащённом и исчерпывающем формате, что позволяет специалистам по информационной безопасности — в том числе и сотрудникам команд реагирования на инциденты (SOC/CSIRT) — эффективно использовать их для расследования, анализа и последующего принятия решений. Удобные интерфейсы доступа и возможность построения поисковых запросов делают такие системы неотъемлемым инструментом работы аналитиков и инженеров ИБ.

### Коррелятор

**Коррелятор** — это компонент системы SIEM, выполняющий анализ нормализованных и обогащённых событий с целью выявления цепочек взаимосвязанных событий, которые в совокупности могут указывать на инцидент информационной безопасности.

В отличие от парсера, который работает с отдельными логами, коррелятор обрабатывает события в совокупности, используя заданные правила или сценарии. Эти правила могут описывать, например:

последовательность событий от одного пользователя за короткий промежуток времени (вход, изменение прав, попытка подключения по SSH);

множественные ошибки входа с разных IP-адресов, указывающие на подбор пароля;

аномальные действия, выходящие за пределы типичной активности для конкретного пользователя или системы.

Корреляторы могут быть основаны как на простых правилах, так и на более сложных алгоритмах машинного обучения или поведенческого анализа. Результатом работы коррелятора алерты, которые поступают в SOAR-систему для последующего автоматического реагирования или передачи специалистам для анализа.

Таким образом, коррелятор играет ключевую роль в выявлении инцидентов, трансформируя поток отдельных логов в осмысленные и значимые сценарии потенциальных угроз.

## Описание существующего процесса настройки приема логов

Для того, чтобы настроить прием логов из источника, необходимо корректно выстроить цепь от источника логов до СХЛ. Это значит, что необходимо получить данные, нормализовать их и отправить в СХЛ.

### Описание наиболее распространенного бизнес-процесса

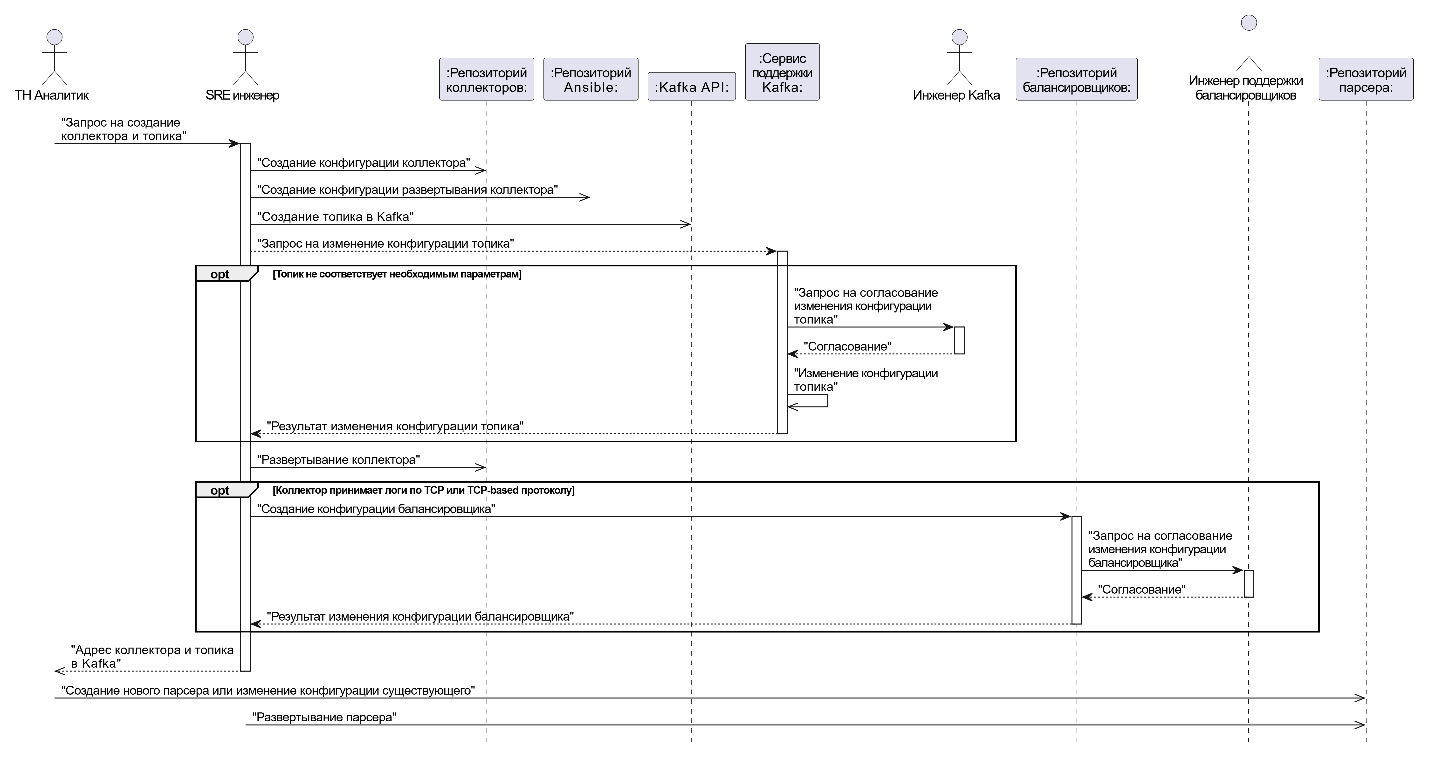


Рисунок 2 – Процесс настройки сбора логов с источника с использованием сетевого коллектора

На рисунке 2 представлен наиболее распространенный процесс настройки сбора логов. Под «наиболее распространенным» я подразумеваю, что он не содержит краевых случаев, например, когда источник умеет писать напрямую в Kafka, когда коллектор сам вычитывает логи из базы данных источника SQL-запросами или когда на один из запросов согласования приходит отказ[[1]](#footnote-1)

Для настройки сбора логов в таком процессе необходимо настроить коллектор, топик, в который будет писать данные коллектор, и парсер, который будет перекладывать события из топика в систему хранения логов.

#### Создание коллектора

Создание коллектора подразумевает создание конфигурационного файла самого коллектора, который включает в себя поля «входа» и «выхода», определяющие, откуда коллектор получает данные и куда он их передает.

Коллектор, получающий данные по сети, работает как сервер с открытым TCP или UDP портом. Все сетевые коллекторы размещены на виртуальных машинах в разных дата-центрах так, что каждый коллектор представлен на каждой виртуальной машине. Такой подход упрощает управление, так как все коллекторы находятся в одном месте и управляются из одного места, а также обеспечивает повышенную отказоустойчивость, так как трафик до коллекторов, особенно по TCP и TCP-based протоколам, поддается балансировке.

Из-за того, что на одной машине работает сразу несколько коллекторов, работающих как сервер, необходимо выделить уникальный порт под каждый коллектор. Определение, какой порт будет занимать тот или иной коллектор определяется вручную на этапе написания конфигурации. SRE инженер, сверяясь с репозиторием с конфигурациями коллекторов, ищет минимальный свободный порт и прописывает его. Конечно, порт можно выбирать и случайным образом, но мы придерживаемся того, что последовательное резервирование портов выглядит более «аккуратно».

На «выходе» коллектор всегда передает данные в Kafka. Для отправки данных в топик Kafka необходима учетная запись (далее УЗ), способная писать в этот топик. Эту УЗ необходимо прописать в «выходе» в конфигурации коллектора, как и название топика и адреса брокеров Kafka кластера. Сейчас во всех коллекторах используется одна и та же УЗ с доступом на запись в любой топик. Это более простой подход, но не самый безопасный. Вернемся к нему позднее в данной работе. Так как название топика обычно определяет TH аналитик, оно заранее известно, поэтому топик можно указать еще до его создания.

Как уже сказано ранее, трафик с источника до коллектора можно балансировать. В случае с коллекторами, принимающими логи по TCP и TCP-based протоколам, это делается наиболее просто и быстро – используются балансировщики, которые предоставляются как платформенное решение внутри компании. Это значит, что достаточно выбрать доменное имя и в определенном репозитории описать конфигурацию балансировщика в определенном формате. Все обслуживание на себя берет команда обслуживания балансировщиков. Мы используем один и тот же адрес балансировщика для всех источников – меняется только порт. Соответственно, при создании нового коллектора необходимо создать новую запись в конфигурационном файле балансировщика с указанием, какой порт должен прослушивать балансировщик и куда он должен переадресовывать трафик.

TODO: листинг с примером файла балансировщика

По правилам использования балансировщика, любые изменения в конфигурациях проходят через согласование командой обслуживания балансировщиков. Время ответа коллег может достигать суток, но чаще это несколько часов.

#### Создание топика

Kafka-кластера в компании так же предоставляются в формате Infrastructure as a Service (IaaS). То есть, в кластере SOC команда SRE имеет учетные записи, которые способны писать и читать в любые топики, а также имеет право на создание топиков. Все остальные действия выполняются через команду обслуживания Kafka. Создание топика происходит через веб-сервис, разработанный командой обслуживания Kafka. Для создания топика необходимо авторизоваться, а затем заполнить форму с указанием имени, описания и других параметров топика. Топик создается автоматически после отправки формы. При создании топика некоторые параметры в форме (например, *retention.ms* – время жизни сообщения в топике) ограничены по максимальному принимаемому значению. Например, максимальное время жизни сообщения в топике – 24 часа. В случае, если необходимо создать топик с параметрами выше максимальных в форме – необходимо сначала создать топик, а затем через сервис поддержки запросить увеличение параметров. В сервисе поддержки запрос увеличения требует согласования командой обслуживания Kafka. Среднее время обработки запроса на увеличение более одного рабочего дня (8 часов), что можно приравнять к суткам в контексте выполнения задачи.

#### Создание парсера

Код парсеров в SOC сгруппирован в репозитории по типу обрабатываемых логов. Например, парсеры сетевых логов, парсеры логов Windows, парсеры логов VPN и т.д. В каждом репозитории может быть код нескольких пайплайнов. В контексте парсеров пайплайн – это сценарий, в котором определено, откуда брать лог, как его обрабатывать (например, какие поля нужно переименовать, а какие удалить или добавить) и куда его отправлять.

Создание парсера под определенный источник значит или создание нового пайплайна в существующем репозитории парсеров, или создание нового репозитория с новым пайплайном в нем. Обычно, этим занимается команда Thread Hunting аналитиков, так как они в контексте того, как нужно обрабатывать логи для коррелятора. Команда SRE занимается только развертыванием парсеров, выделением ресурсов под них и другими операционными задачами.

### Краевые случаи бизнес-процесса

В главе 1.2.1 описан наиболее распространенный сценарий настройки сбора логов с источника. Но есть источники, к которым не применим данный сценарий.

#### Источники, умеющие писать в Kafka

В случае, если источник умеет писать напрямую в Kafka, это наиболее предпочитаемый вариант. Kafka – распределенная и отказоустойчивая система и добавление любых прослоек перед ней скорее создаст дополнительные точки отказа, чем принесет какую-то пользу от стандартизации подхода. То есть, если источник умеет писать в Kafka, создается топик, под него через поддержку создается УЗ с правами на запись, а учетные данные передаются команде, ответственной за источник. Соответственно, далее под источник создается/настраивается парсер, коллектор не создается.

#### Источники, логи которых можно получить только из базы данных

Существуют источники, которые пишут логи аудита только к себе в базу данных. Например, так делают Kaspersky Security Center – антивирусное ПО и Jira – ПО для управления проектами. Данные с таких источников важны, но получать их через коллектор-сервер не получится, необходим коллектор, который будет читать данные прямо из базы данных источника и перекладывать их в Kafka. Это частные случаи и для них всегда необходима определенная логика работы с БД. Для такого используются коллекторы на Logstash или приложения собственной разработки. Logstash позволяет настроить чтение из реляционных баз данных и записывать в топик Kafka. В случае NoSQL баз данных могут использоваться коллекторы собственной разработки – скрипты на Python и других языках программирования, которые также занимаются перекладыванием логов из базы данных в топик Kafka.

## Описание недостатков в существующем процессе.

### Времязатраты на создание коллектора и топика

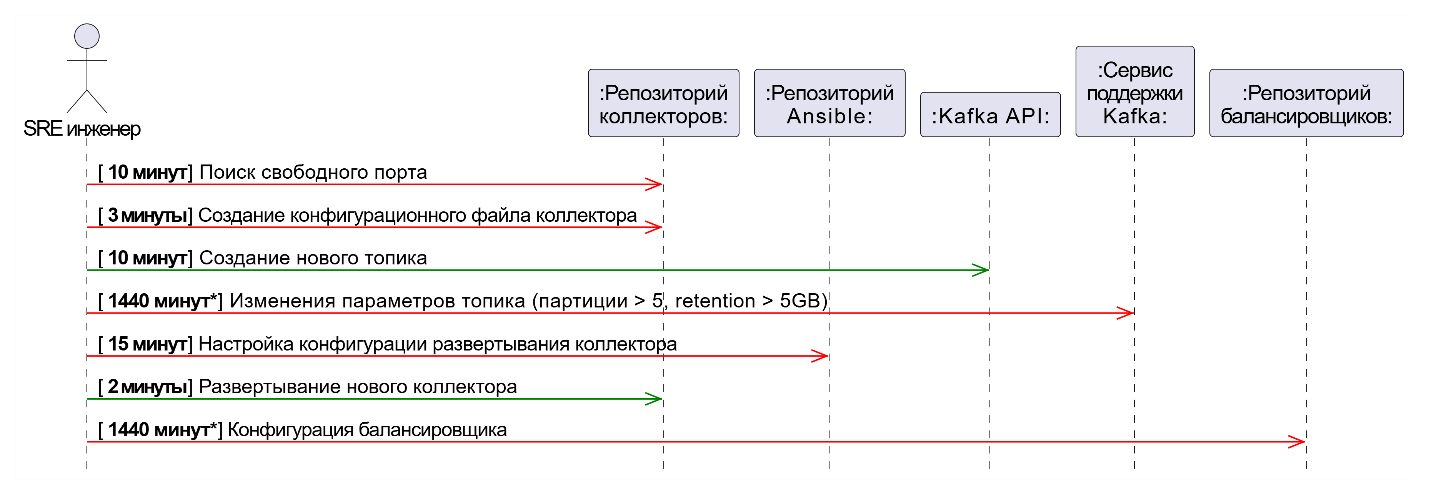


Рисунок 2 – Описание процесса настройки приема данных из источника по TCP или TCP-based протоколу с указанием времязатрат.

На рисунке представлены этапы, необходимые для настройки приема данных по TCP или TCP-based протоколу Красным цветом выделены этапы, выполняемые вручную, зеленым – автоматизированные. 1440 минут занимают этапы, в которых участвуют сторонние команды, среднее время ответа на запрос которых составляет около суток.

TODO

# Проектирование целевого процесса

Так как существующий процесс содержит некоторые недостатки, обновленный процесс должен их устранять. Первостепенная задача – автоматизировать процессы, задействующие ручной труд команды SRE инженеров, чтобы сократить операционную нагрузку на них, тем самым позволив им уделять больше времени задачам, связанным с помощью другим командам. По этой причине изменения в целевом процессе коснутся в первую очередь тех подпроцессов и сущностей, с которыми взаимодействует команда SRE. Сюда входит создание и конфигурация коллекторов, а также создание сущностей в Kafka.

## Требования к решению

Ключевое требование от команды SRE – все изменения должны происходить централизованно через единый источник правды. То есть, при необходимости создать коллектор или поменять параметры существующего достаточно произвести изменения в одном месте. В случае необходимости получить информацию о коллекторе (например, описание, его адрес или конфигурацию) достаточно обратиться в вышеупомянутый единый источник правды.

Также, обновленный процесс настройки сбора логов с источника должен занимать значительно меньше времени и внимания инженера. Все действия, которые можно автоматизировать, должны быть автоматизированы.

## Сокращение количества участников процесса

В существующем процессе есть два внешних участника – команда обслуживания Kafka и команда обслуживания балансировщиков. Для ускорения процесса настройки сбора логов было бы здорово сократить участие данных команд в процессе до минимального. Рассмотрим далее, как это можно сделать.

### Автоматизация настройки параметров топика

Существующий кластер Kafka развернут на инфраструктуре, находящейся в распоряжении отдела SOC, однако его администрированием занимается команда обслуживания Kafka.

Известно, что настройка параметров уже осуществляется автоматикой, но её запуск происходит только после согласования с ответственными. На текущий момент видятся следующие решения этой проблемы.

1. Договориться о получении сервисной УЗ для управления топиками.
2. Мигрировать на более технологичное решение Kafka as a Service.

#### Попытка получить сервисную УЗ для управления топиками

Первый вариант связан с тем, что управление сущностями в кластере Kafka подразумевает наличие учетной записи с определенными правами. По умолчанию команды, использующие кластера, не имеют таких УЗ. Если убедить команду обслуживания Kafka в том, что для нашей команды эта УЗ необходима, появится возможность создавать свои автоматизации по конфигурации топиков, используя данную УЗ.

После обсуждения данной идеи с командой обслуживания Kafka был получен ответ о невозможности предоставления УЗ с требуемыми правами, так как используемый кластер Kafka находится в состоянии EoL (англ. End-of-Life), что значит, что он не обновляется, и любые изменения в его конфигурации или политиках доступа не осуществляются. Помимо этого, была получена рекомендация мигрировать на решение Kafka as a Service.

#### Миграция на обновленный кластер (Kafka as a Service)

**Kafka as a Service (KaaS)** — это управляемое облачное решение на базе Apache Kafka, предоставляемое внутри компании. В модели KaaS клиентам предоставляются квоты на различные сущности — такие как топики, группы потребителей и учетные записи. Принципиальная разница между текущим решением и KaaS заключается в модели управления: сервис остаётся тем же (Kafka), но процессы взаимодействия стандартизированы и унифицированы.

Ключевым преимуществом KaaS является то, что управление топиками, пользователями и их доступами осуществляется не через обращения к команде сопровождения, а напрямую силами клиентов через веб-интерфейс или API. Это позволяет автоматизировать процессы создания и настройки топиков, а также исключить необходимость согласования изменений с командой обслуживания Kafka, что значительно ускоряет работу и повышает гибкость взаимодействия.

При проектировании обновленного процесса как таковой «миграции» не подразумевается, так как подразумевается, что сначала необходимо создать процесс, а затем мигрировать на него уже существующие источники данных.

### Конфигурация балансировщика

Настройка всех балансировщиков происходит через изменения в одном репозитории, CI/CD пайплайн которого автоматически разворачивает балансировщики с необходимыми конфигурациями. За активное состояние балансировщиков, их процесс развертывания и другие автоматизации в этом репозитории ответственна команда обслуживания балансировщиков.

Чтобы создать новый балансировщик или изменить конфигурацию существующего, необходимо внести изменение в репозиторий. Проблема в том, что прямое внесение изменений доступно только команде обслуживания балансировщиков. Все остальные изменения вносятся через механизм форков (англ. fork). По правилам системы контроля версий в компании создание форков возможно только от личной учетной записи, сервисные УЗ не могут иметь личных репозиториев. В таких условиях, данную задачу невозможно автоматизировать полностью. В таком случае можно рассмотреть следующие варианты.

1. Получить сервисную УЗ, которая способна создавать изменения в репозитории конфигураций балансировщиков без создания форка.
2. Заранее создать необходимые конфигурации балансировщиков

Первый вариант не был согласован командой обслуживания балансировщиков, так как они не готовы брать на себя ответственность за обслуживание данной УЗ. Остается только второй вариант.

Второй вариант подразумевает, что вместо автоматической настройки балансировщика при создании или изменении коллектора, несколько балансировщиков будут заранее сконфигурированы как готовые к использованию. Минус от такого подхода лишь в том, что в конфигурации балансировщика не будет известно, трафик какого источника обрабатывает балансировщик, так предполагается создание сущностей с динамически генерируемым именем, например, balancer\_for\_port5678 вместо balancer\_for\_windows\_logs\_collector. Также стоит учесть, что подход подразумевает создание объемного артефакта в ручном или полуавтоматическом режиме, но лишь единожды.

### Итог

Миграция на кластер в KaaS и предварительная подготовка балансировщиков позволят сократить время настройки приема логов с источника примерно на двое суток, что удовлетворяет требованиям, описанным выше.

## Настройка автоматизации

Одно из описанных требований – управление состоянием должно производиться в централизованном месте, которое будет являться единым источником правды. Для организации такой сущности часто применяется Git репозиторий с определенной файловой структурой. Рассмотрим именно этот вариант.

### Предлагаемая структура проекта для управления коллекторами.



Рисунок 3 - файловая структура инвентаря источников

Важно, чтобы созданный инвентарь имел предсказуемую структуру. На рисунке 3 отражена предлагаемая структура файлов и директорий. Идея данного решения в том, чтобы использовать требование уникальности названий директорий в файловых системах как часть механизма идентификации. Название директории будет использоваться как ключ источника – идентификатор, используемый в именах и описаниях коллекторов, топиков Kafka и других сущностях, который позволит определить, к какому источнику относится определенная сущность.

### Структура файла «main.yml»

В директории для каждого коллектора существует файл «main.yml», содержащий описание источника, различную метаинформацию, информацию о развертывании, а также информацию о том, откуда коллектор принимает данные и куда отправляет. На основе этого файла в зависимости от его конфигурации создаются разные сущности.

Листинг 1 - Пример файла с описанием коллектора



### Описание процесса развертывания

Используется Ansible-плейбук, который генерирует конфигурационные файлы, если они не заданы, и кладет их на целевой хост коллектора. Затем запускает контейнеры с монтированием конфигурационных файлов и пробросом указанных переменных окружения.

### Автоматизация поиска свободного порта

Так как адрес коллектора и протокол взаимодействия – часть контракта на поставку данных из источника, изменение адреса крайне нежелательно. Автоматическое назначение порта коллектору в случае сбоя может нарушить соответствие топика и данных, которые в нем лежат, перепутав вообще все, что недопустимо.

Но, чтобы поиск свободного порта во время создания порта не занимал много времени, можно ускорить его, внедрив какие-либо подсказки для инженера. Например, в репозитории коллекторов настроить автоматическое обновление файла README.md, в котором будет указываться актуальный свободный порт для коллектора.

То есть, после внесения изменений в главную ветку проекта, CI/CD пайплайн будет искать свободный порт, а затем отображать его в README.md. Это позволит ускорить время поиска свободного порта при создании коллектора с 10 минут до двух.

### Автоматизация создания топика

Как было сказано ранее, существующий кластер Kafka имеет статус EoL, поэтому в данной работе не будет рассматриваться сценарий автоматизации создания топиков и других сущностей на этом кластере.

В случае с KaaS кластером для создания топика достаточно авторизоваться через сервисную учетную запись с доступом на управление сущностями в кластере KaaS, а затем по API вызвать метод создания топика с определенными параметрами (количество партиций, политики удаления сообщений, описание).

### Автоматизация создания пользователей

В существующем кластере используется одна и та же УЗ для записи в любые топики Kafka. Это нарушает принцип наименьших привилегий, при утечке этой УЗ есть риск того, что во множество топиков могут отправить множество бессмысленных сообщений, создав тем самым искусственную нагрузку на парсеры, а также перетереть сообщения со смыслом, которые еще не были вычитаны парсерами. Как было сказано ранее, потеря событий для SOC – риск того, что какая-либо нелегитимная активность не будет замечена вовремя или вовсе.

KaaS позволяет создавать УЗ в кластере и управлять их правами через API, что позволяет создавать отдельные УЗ под каждый топик. Для этого, как и с топиком, достаточно вызвать метод API с заполнением необходимых параметров.

Настройка прав доступа пользователя в KaaS осуществляется через отдельную сущность – разрешение (англ., permission). Для настройки доступа пользователю к определенному топику необходимо создать permission между топиком и пользователем. Разрешения могут быть типов ConsumePermission или ProducePermission – разрешение на чтение из топика и разрешение на запись в топик соответственно. Создавать разрешения также можно через API, указав имя существующей УЗ в кластере KaaS, название топика и идентификатор разрешения (произвольное число, уникальное в рамках пары «топик-УЗ», так как разрешений может быть несколько).

### Автоматизация создания коллектора

Создание коллектора подразумевает разворачивание активного экземпляра коллектора, который будет работать ровно с той конфигурацией, что указана в едином источнике правды. Данная автоматизация должна учитывать случаи, когда конфигурацию необходимо сначала сгенерировать из шаблона, и когда конфигурация уже лежит в директории единого источника правды.

Для задачи развертывания было бы удобно использовать Ansible[[2]](#footnote-2), так как он позволяет в простом формате описать желаемое состояние, а затем не менее простой командой запустить скрипт, применяющий это состояние. Также Ansible поддерживает генерацию файлов из шаблона и умеет доставлять файлы на удаленные хосты, что может упростить задачу конфигурации коллектора.

### Предоставление информации о развернутом экземпляре заинтересованным лицам

Чаще всего заинтересованным лицам (TH аналитики, команда, отвечающая за источник логов) нужно знать две вещи: «куда отправлять данные?» и «все ли работает?». На первый вопрос ответ дает единый источник правды, в котором должен быть указан адрес, по которому коллектор принимает данные. На второй вопрос должны отвечать системы мониторинга. Ключевые метрики для понимания, работает ли коллектор – скорость приема данных коллектором, скорость записи данных в кластер Kafka и наличие (количество) сообщений в Kafka. Если эти метрики не нулевые при отправке данных в коллектор, значит все работает хорошо. Удобным решением было бы собрать дашборд с указанными метриками, таким образом, заинтересованные лица смогли бы получать ответы на свои вопросы без участия команды SRE.

## Настройка CI/CD пайплайнов в репозитории

CI/CD пайплайн (англ. Continuous Integration / Continuous Delivery & Deployment Pipeline) - определение пайплайна.

В данном репозитории при внесении изменений мы ожидаем, что будет произведено достаточно много действий. Чтобы не усложнять решение за счет ограничений пайплайнов, все изменения будем производить в рамках одного шага – развертывания. На Merge Request будут пайплайны со статическими проверками кода и конфигураций, которые включают в себя проверки отсутствия коллизий занимаемых портов, проверки корректности существующих конфигураций коллекторов.

### Проверка корректности конфигурации

Перед развертыванием измененных конфигураций необходимо удостовериться, точно ли они корректны. Так как схема конфигурационных файлов была определена нами, необходимо написать линтер(ы) конфигураций. Для этой задачи можно воспользоваться готовыми инструментами или написать его вручную.

Если говорить о готовых инструментах, то есть те, что позволяют задать «схему файла» с определением того, что должно содержаться в том или ином поле конфигурационного файла. Это вариант для самой базовой проверки синтаксиса, она позволяет убедиться, что как минимум типы данных используются корректные.

Если необходимо проверять более сложные условия (например, уникальность значений ключа среди множества файлов или что-то сложнее), то без скриптов собственной разработки не обойтись. Чаще такие типы линтеров создаются на Python из-за простоты разработки.

### Применение изменений

Тут говорю про то, как запускается тот комплексный скрипт

# Реализация целевого процесса

## Автоматизация

Тут рассказываю про то, как реализовал логику применения состояния. Про то, что есть основной цикл, читающий директорию, про то, что он запускается внутри CI/CD или вручную, если необходимо, про то, что он авторизуется в KaaS, а также занимается деплоем и остановкой/удалением сущностей.

## Обновленный процесс

Показываю, как фактически теперь выглядит поставка источника на мониторинг.

## Сравнение предыдущего процесса и обновленного (

Тут показываю графики по времени, диаграммы было/стало, возможно еще какую-то табличку. Это должен быть последний раздел этой главы

Заключение

В рамках данной работы …..

Список использованных источников и литературы

1. Site Reliability Engineering: How Google Runs Production Systems // Бетси Бейер, Нейл Ричард Мёрфи, Дэвид Рензин, Кент Кавахара и Стивен Торн, 2016.
2. UML. Основы. Третье издание. Краткое руководство по стандартному языку объектного моделирования // Мартин Фаулер, 2013.
3. Docker Documentation // Docker – [Б.м.]., 2024 – URL: [https://docs.docker.com](https://docs.docker.com/) (дата обращения: 20.06.2024).
4. What is SIEM? // Microsoft – [Б.м.]., 2025 – URL: https://www.microsoft.com/en-us/security/business/security-101/what-is-siem

1. Случай отказа в согласовании в сценариях, указанных на рисунке 2 не будет рассмотрен, так как таких ситуаций еще не возникало и нет предпосылок, чтобы они возникли. Отказ может возникнуть в случае проблем с какими-либо ресурсами, рассмотрение этого сценария не имеет смысла в данной работе. [↑](#footnote-ref-1)
2. Ansible - инструмент автоматизации управления конфигурациями и инфраструктурой. Ansible позволяет описывать процесс развертывания коллектора в виде декларативно описанных плейбуков (файлов, с описанием необходимых действий), что обеспечивает воспроизводимость и минимизирует количество ручных действий. [↑](#footnote-ref-2)