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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З.А. Полни Меня,

Заполни мои регалии и место работы

*подпись*

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

Научный консультант

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Зоркин,

ведущий инженер, ООО "ТЦР"

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 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

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.

# Введение в предметную область

## Инфраструктура SIEM

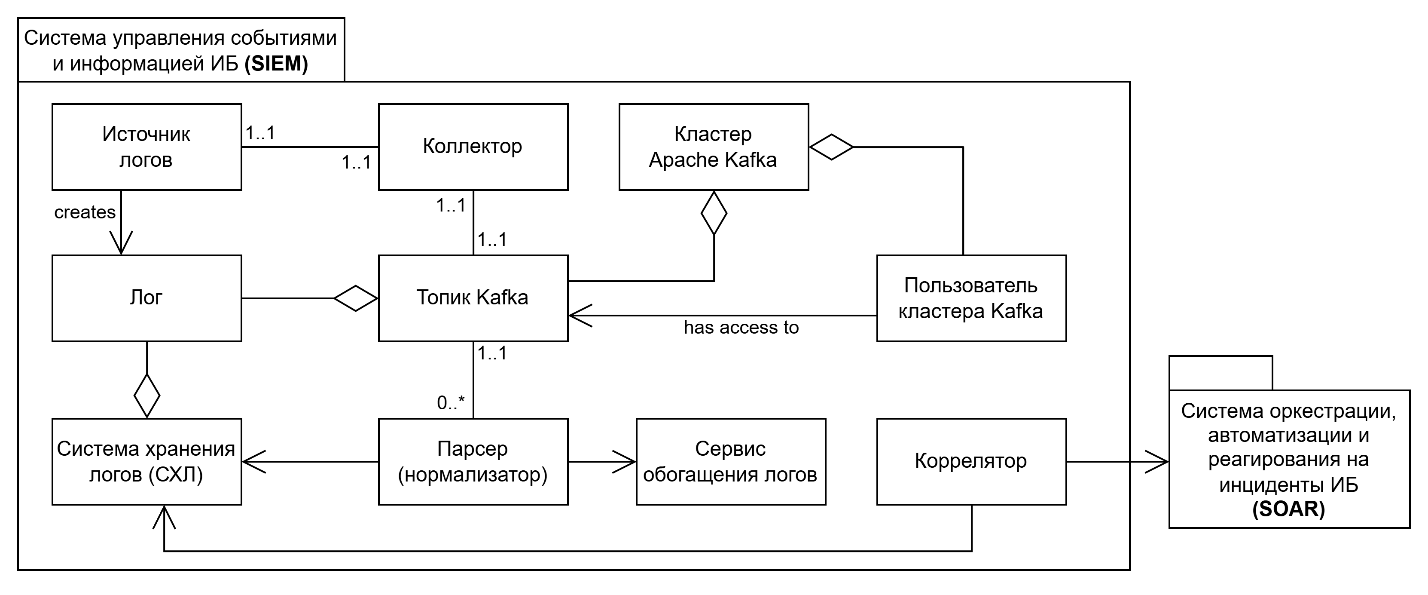


Рисунок 1 – Предметная область SIEM

На рисунке 1 отражены основные сущности системы управления информацией и событиями информационной безопасности. Рассмотрим их подробнее.

### 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 во введении, 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 являются сигналы (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.

## Настройка приема логов из источника

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. На этапе сбора данных шаги могут различаться

TODO Диаграмма с if-else и как делать, если источник умеет писать в Kafka, умеет отправлять данные по сети, или умеет хранить локально и делай, что хочешь.

### (Поместить в конец) Времязатраты на создание коллектора

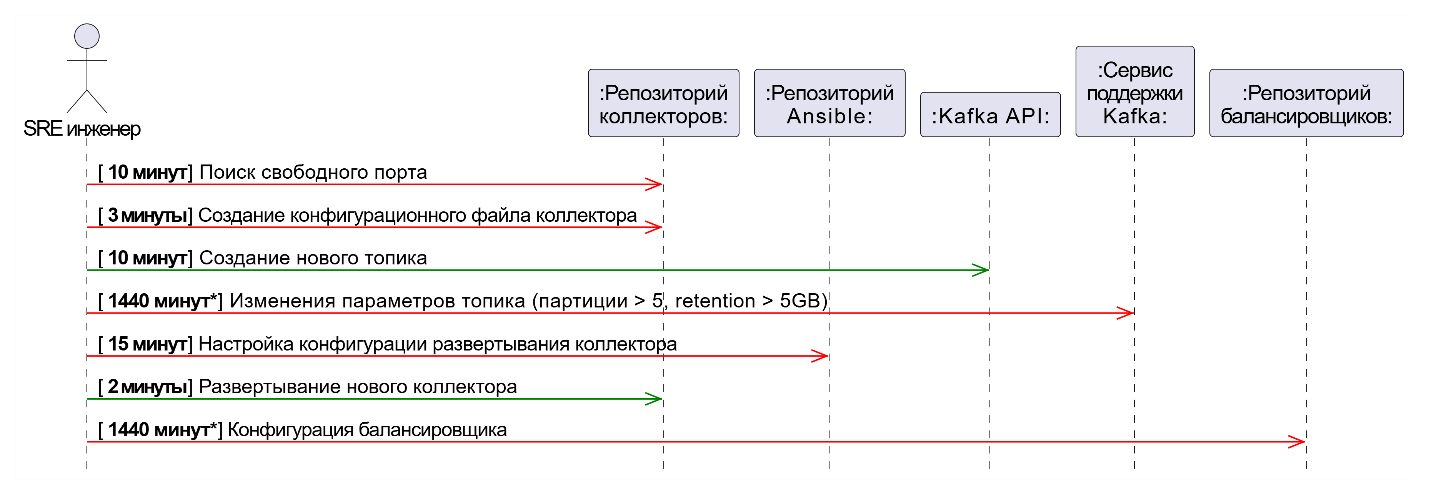


Рисунок 2 – Описание процесса настройки приема данных из источника по TCP или TCP-based протоколу с указанием времязатрат.

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.

# Анализ существующих процессов

## Процесс настройки сбора событий

### Создание коллектора

### Прямая интеграция с Kafka

Блаблаблабла

## Процесс настройки нормализации событий

### Создание парсера

Блаблаблабла

### Написание правил нормализации

#### Обогащение данных

Блаблабла

#### Исключение данных

Блаблабла

# Автоматизация процесса

## Избавление от людей в процессе

Прежде всего, максимальный прирост к скорости создания коллектора будет приобретен в случае избавления от прослоек в виде человека. По рисунку 2 видно, что это необходимо сделать для этапов изменения параметров топика и конфигурации балансировщика.

### Настройка параметров топика

Известно, что настройка уже осуществляется автоматикой, но для ее использования необходимо согласование человеком из операционной поддержки. Есть 2 решения данной ситуации.

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

Первый вариант нам не разрешили, настояли перейти на второй, что дало нам больше свободы в управлении сущностями, но потребует постепенно мигрировать все существующие топики на новое решение.

### Конфигурация балансировщика

Конфигурация происходит через GitOps подход, за реализацию которого ответственна инфраструктурная команда. Изменение файла в их репозитории порождает изменение конфигурации балансировщика.

Соответственно, чтобы создать новый балансировщик, необходимо внести изменение в репозиторий. Проблема в том, что внесение изменений доступно лишь через форк. А по правилам компании создавать форки можно только от личной учетной записи, но никак не от сервисной. Соответственно, это невозможно автоматизировать. В данном случае мы можем рассмотреть следующие варианты.

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

Второй вариант нам не разрешила команда, ответственная за балансировку трафика, остается только первый. Чем мы и воспользуемся. Минус от такого подхода лишь в том, что в конфигурации балансировщика не будет известно, «от чего именно» данный балансировщик, так как конфигурация заранее предполагает создание сущностей с динамически генерируемым именем, например, balancer\_for\_port5678.

### Заключение

Избавившись от людей в двух этих этапах, время настройки приема логов с источника сокращено до минимального. Но этого недостаточно, так как в целях есть полная автоматизация процесса, что и будет описано далее.

## Настройка автоматизации

### Предлагаемая структура проекта для инвентаризации источников.



Рисунок 3 - файловая структура инвентаря источников

Одна из приоритетных задач – настроить единый источник правды о существующих источниках логов. Важно, чтобы созданный инвентарь источников 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 конфиги, если они не заданы. Конфиги кладет на хост. Затем запускает контейнеры с пробросом конфигов как volume 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

### Создание топика в собственном кластере

Блаблаб

### Создание топика в кластере KaaS решения

Блаблабла

## Автоматизация создания коллектора

### Требуемые параметры для запуска

### Генерация конфигурационного файла

### Развертывание сервиса

### Предоставление информации о развернутом экземпляре пользователю

## 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

Существуют схемы Yaml, тут хочу описать то, как будет выглядеть схема Yaml и как именно будет выполняться проверка.

# Автоматизация создания парсера

### Генерация конфигурационных файлов из шаблона

### Механизм развертывания парсера

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