Руководство администратора ПО

«Сервис верификации действий пользователя»

1. ВВЕДЕНИЕ

1.1. Область применения

Настоящий документ предназначен для администраторов, эксплуатирующих программное обеспечение «Сервис верификации действий пользователя» (далее — Система). Руководство содержит сведения о порядке развёртывания, настройки, администрирования и контроля работоспособности Системы, а также о порядке взаимодействия с административным API.

Система предназначена для приёма событий о действиях пользователей, сохранения этих событий в базе данных, применения правил верификации (детекции) и формирования сообщений о выявленных инцидентах с последующей доставкой уведомлений во внешние каналы (Telegram и/или webhook).

1.2. Перечень выполняемых функций администратора/оператора

Администратор обеспечивает развертывание и эксплуатацию Системы в контейнерной среде, выполняет настройку подключений к PostgreSQL и RabbitMQ, задаёт параметры безопасности (ключи доступа), управляет конфигурацией приёма событий и правилами верификации, осуществляет мониторинг состояния компонентов, анализирует журналы (логи), выполняет процедуры резервного копирования и восстановления, а также выполняет обновление и перезапуск сервисов.

1.3. Уровень подготовки администратора/оператора

Администратор должен обладать навыками работы с операционными системами семейства Linux, уметь использовать контейнерную среду Docker и понимать принципы функционирования сетевых сервисов HTTP. Необходимы базовые знания СУБД PostgreSQL, включая создание баз данных, проверку соединений и выполнение резервного копирования.

Для эксплуатации системы достаточно одного администратора, прошедшего подготовку по использованию программного обеспечения и ознакомленного с настоящим руководством.

1.4. Перечень документации

В состав документации, с которой необходимо ознакомиться администратору Системы входят:

- Функциональная спецификация системы
- Техническое задание на разработку системы
- Настоящее руководство администратора

2. УСТАНОВКА СИСТЕМЫ

2.1.1. Требования к вычислительным ресурсам

Система развёртывается в контейнеризированной среде и состоит из нескольких взаимодействующих сервисов. Приведённые требования относятся к одному экземпляру Системы.

Для запуска одного экземпляра Системы требуется не менее 2 ГБ оперативной памяти и 10 ГБ свободного дискового пространства. Указанные ресурсы должны обеспечивать

одновременную работу сервисов приёма событий, обработки, хранения данных и отправки уведомлений.

Система допускает горизонтальное масштабирование. При увеличении объёма входящих событий допускается развёртывание нескольких экземпляров отдельных сервисов (в первую очередь модуля обработки событий) в среде оркестрации контейнеров. В этом случае требования к ресурсам определяются количеством запущенных экземпляров и распределяются между узлами инфраструктуры.

2.1.2. Программные требования

Система предназначена для эксплуатации в контейнеризированной среде (Docker, Docker Compose, Kubernetes или аналогичной). Все компоненты Системы запускаются в отдельных контейнерах и взаимодействуют по сети.

Для функционирования Системы требуется наличие:

- контейнерной платформы Docker версии 20.10 или выше либо совместимой системы оркестрации;
- базы данных PostgreSQL версии 16 или выше;
- брокера сообщений RabbitMQ версии 3.x.

Дополнительно могут использоваться системы мониторинга и сбора метрик. Их наличие не является обязательным для функционирования Системы.

2.2. Порядок установки

1. Смонтируйте диск с дистрибутивом в папку /mnt
2. Скопируйте из дистрибутива исходники из папки /mnt в папку /opt/*sentry*
3. Отредактируйте файл docker-compose.yml в соответствии с разделом 3.2 данного документа
4. Смените текущую папку на /opt/*sentry* и выполните команду: docker compose up -d --build
5. Проверьте работоспособность системы.

3. НАСТРОЙКА СИСТЕМЫ

3.1. Общие сведения

Система поставляется в виде набора сервисов, взаимодействующих между собой посредством брокера сообщений RabbitMQ и общей базы данных PostgreSQL. Взаимодействие по HTTP используется только для внешних клиентов и административного доступа.

3.1.1. Состав сервисов:

- http-in — приём событий пользователя по HTTP, опциональная трансформация входного запроса (JavaScript), публикация события в RabbitMQ.
- detection-worker (модуль api) — обработка событий, сохранение в PostgreSQL, применение JavaScript-правил верификации, публикация инцидентов в RabbitMQ.

- telegram-out — потребление инцидентов и доставка уведомлений (Telegram и/или webhook).
- admin-api — административный JSON-RPC 2.0 API для управления маршрутами http-in и правилами верификации.
- PostgreSQL — хранение событий, правил и маршрутов.
- RabbitMQ — транспорт событий и инцидентов между сервисами.

3.1.2. Основной поток данных:

- http-in публикует входящие события с routing key sentryguard.events.raw.
- detection-worker потребляет события, сохраняет их и применяет правила.
- При выявлении угрозы публикуются инциденты в routing key sentryguard.actions и sentryguard.alerts.
- telegram-out доставляет сообщения во внешние каналы.

3.2. Конфигурируемые параметры

Ниже приведены переменные окружения, реально используемые в коде сервисов.

3.2.1. admin-api (административный JSON-RPC)

Параметры сервиса:

- HOST — адрес и порт HTTP-сервера (по умолчанию :80)
- METRICS_PORT — порт healthz (по умолчанию :3000)
- ACCESS_KEY — ключ доступа к приватным методам (/) (обязателен)

Параметры PostgreSQL:

- DB_HOST — хост БД (по умолчанию db)
- DB_PORT — порт БД (по умолчанию 5432)
- DB_USER — пользователь БД (обязателен)
- DB_PASSWORD — пароль БД (обязателен)
- DB_NAME — имя БД (по умолчанию sentry_db)
- DB_MAX_OPEN_CONNECTIONS — максимум открытых соединений (по умолчанию 30)
- DB_MAX_IDLE_CONNECTIONS — максимум неактивных соединений (по умолчанию 30)
- DB_CONN_MAX_TIME — максимальное время жизни соединения (по умолчанию 3m)

Параметры middleware JSON-RPC (присутствуют в коде):

- RPC_LOGGING_ENABLED — включить логирование вызовов JSON-RPC (по умолчанию false)
- RPC_METRICS_ENABLED — включить метрики вызовов JSON-RPC (по умолчанию false)

3.2.2. http-in (приём событий по HTTP)

Параметры сервиса:

- HOST — адрес и порт HTTP-сервера (по умолчанию :80)

- API_KEY — ключ доступа к входящим HTTP-запросам (проверяется по заголовкам Authorization: Bearer ... и/или X-API-Key)
- METRICS_PORT — порт healthz/metrics (по умолчанию :3000)

Параметры RabbitMQ:

- AMQP_HOST — URI подключения (по умолчанию amqp://localhost:5672/)
- AMQP_EXCHANGE — имя exchange (по умолчанию sentryguard)
- AMQP_COMMAND_TIMEOUT — таймаут AMQP-операций (по умолчанию 10s)

Параметры PostgreSQL для загрузки маршрутов:

- CONFIG_DB_HOST — имя хоста или адрес сервера PostgreSQL, содержащего конфигурационные таблицы маршрутов.
- CONFIG_DB_PORT — порт подключения к PostgreSQL. Если значение не задано, используется стандартный порт 5432.
- CONFIG_DB_USER — имя пользователя базы данных, от имени которого сервис выполняет чтение конфигурации маршрутов.
- CONFIG_DB_PASSWORD — пароль указанного пользователя базы данных.
- CONFIG_DB_NAME — имя базы данных, в которой расположена таблица http_in_routes.
- CONFIG_DB_SSLMODE — режим SSL-подключения к PostgreSQL (например: disable, require, verify-ca, verify-full). При отсутствии значения используется режим disable.

3.2.3. detection-worker (модуль api)

Параметры RabbitMQ:

- AMQP_HOST — URI подключения (по умолчанию amqp://localhost:5672/)
- AMQP_EXCHANGE — имя exchange (по умолчанию sentryguard)
- AMQP_COMMAND_TIMEOUT — таймаут AMQP-операций (по умолчанию 10s)

Параметры PostgreSQL:

- DB_HOST — адрес сервера PostgreSQL, содержащего данные Системы.
- DB_PORT — порт подключения к PostgreSQL. При отсутствии значения используется стандартный порт 5432.
- DB_USER — имя пользователя базы данных, под которым сервис выполняет запись событий и чтение правил.
- DB_PASSWORD — пароль пользователя базы данных.
- DB_NAME — имя базы данных Системы.
- DB_SSLMODE — режим SSL-подключения к PostgreSQL (disable, require, verify-ca, verify-full). Если не задан, используется disable.

Параметры мониторинга:

- METRICS_PORT — адрес и порт HTTP-сервера, на котором сервис публикует служебные эндпоинты (/healthz, /metrics). По умолчанию используется :3000.

3.2.4. telegram-out (доставка уведомлений)

Параметры RabbitMQ:

- AMQP_HOST — URI подключения (по умолчанию amqp://localhost:5672/)
- AMQP_EXCHANGE — имя exchange (по умолчанию sentryguard)

Параметры Telegram:

- TELEGRAM_BOT_TOKEN — токен бота
- TELEGRAM_CHAT_ID — идентификатор чата/канала (обязателен для отправки в Telegram)

Параметры webhook:

- WEBHOOK_URL — общий webhook URL (если задан, может использоваться как единый endpoint)
- WEBHOOK_ACTIONS_URL — webhook для очереди actions
- WEBHOOK_ALERTS_URL — webhook для очереди alerts

Параметры мониторинга:

- METRICS_PORT (по умолчанию :3000)

4. ОПИСАНИЕ API

4.1. Общие сведения

Административный интерфейс Системы реализован по протоколу JSON-RPC 2.0.

Публичный endpoint используется только для проверки доступности сервиса, а все операции управления выполняются через приватный endpoint.

Публичный endpoint: POST /public

Приватный endpoint: POST /

Для вызова приватных методов требуется передать ключ доступа в заголовке:

- Authorization: Bearer <ACCESS_KEY>
- Content-Type: application/json

4.2. Публичные методы

ping

Метод предназначен для проверки доступности административного сервиса.

Запрос:

```
{  "jsonrpc": "2.0",  "method": "ping",  "id": 1 }
```

Ответ:

```
{  "jsonrpc": "2.0",  "result": "pong",  "id": 1 }
```

4.3. Управление правилами верификации

Правила представляют собой JavaScript-функции, применяемые к событиям пользователя.

При срабатывании правила формируется инцидент.

rules.list

Возвращает список всех правил.

Параметры не требуются.

Rules.get

Возвращает правило по идентификатору.

Параметры:

- id — идентификатор правила.

rules.create

Создаёт новое правило.

Параметры:

- name — уникальное имя правила
- script — JavaScript-код проверки
- priority — приоритет выполнения
- is_active — признак активности

rules.update

Изменяет существующее правило.

Параметры:

- id — идентификатор правила
- name — новое имя (опционально)
- script — новый код (опционально)
- priority — новый приоритет (опционально)
- is_active — состояние активности (опционально)

rules.delete

Удаляет правило.

Параметры:

- id — идентификатор правила

rules.activate

Активирует правило.

Параметры:

- id — идентификатор правила

rules.deactivate

Отключает правило.

Параметры:

- id — идентификатор правила

4.4. Управление маршрутами приёма событий

Маршруты определяют, какие HTTP-запросы принимаются сервисом http-in и как преобразуется входящее сообщение.

routes.list

Возвращает список всех маршрутов.

routes.get

Возвращает маршрут по методу и URL.

Параметры:

- method — HTTP-метод
- url — путь

routes.create

Создаёт маршрут.

Параметры:

- method — HTTP-метод
- url — путь
- request_script — JavaScript преобразования входящего запроса

- response_script — сохранённый ответ (в текущей версии не используется)

routes.update

Изменяет маршрут.

Параметры:

- method — текущий метод
- url — текущий путь
- new_method — новый метод (опционально)
- new_url — новый путь (опционально)
- request_script — новый скрипт (опционально)
- response_script — новый скрипт ответа (опционально)

routes.delete

Удаляет маршрут.

Параметры:

- method — HTTP-метод
- url — путь