Функциональная спецификация программного обеспечения «Сервис верификации действий пользователя (SentryGuard)»

1. Описание

Настоящий документ содержит описание функциональных характеристик программного обеспечения «Сервис верификации действий пользователя (SentryGuard)» (далее «Сервис» или «SentryGuard»).

2. Среда функционирования продукта

Сервис функционирует как внешнее, независимое решение в среде контейнерной виртуализации. Предпочтительной средой являются системы оркестрации Docker Swarm или Kubernetes. Сервис развертывается отдельно от защищаемых систем и взаимодействует с ними только через четко определенные API и брокеры сообщений, не требуя интеграции на уровне прокси или глубокого внедрения в код приложения заказчика.

3. Функциональные требования:

Сервис SentryGuard предназначен для внешнего анализа пользовательской активности в режиме, близком к реальному времени. Он принимает события от защищаемых систем, анализирует их на предмет аномалий и угроз, и возвращает вердикт или инициирует реакцию через стандартные механизмы интеграции. Основной задачей является выявление паттернов, свидетельствующих о мультиаккаунтинге, использовании автоматизированных средств (ботов), мошеннических операциях и иных злоупотреблениях.

Сервис представляет следующую функциональность:

- API для приема событий (Push-модель): предоставляет защищаемым системам унифицированный HTTP(S) REST API для отправки событий о действиях пользователей. Каждое событие содержит стандартный набор метаданных (ID пользователя, тип действия, временную метку, IP, User-Agent и др.) и произвольные бизнес-атрибуты (сумма ставки, ID предмета и т.п.).
- Асинхронная обработка через брокер: принятые через API события немедленно помещаются в надежную очередь сообщений RabbitMQ. Это обеспечивает буферизацию нагрузки, отказоустойчивость и отделение приема трафика от его анализа.
- Гибкая система правил обнаружения на базе JavaScript (Low Code): конфигурация и создание правил обнаружения угроз осуществляется путем написания пользовательских JavaScript-функций. Данный подход обеспечивает максимальную гибкость и доступен для использования разработчикам любого уровня. Это позволяет реализовать любую логику проверки — от простых пороговых условий до сложных многошаговых корреляций и статистического анализа с использованием полной мощи языка JavaScript.
- Детекторы аномалий (реализуемые как JS-правила):
    - Детектор мультиаккаунтов: анализирует атрибуты сессий (IP-адрес, отпечаток устройства) и ищет совпадения с другими аккаунтами в истории.

    - Детектор аномальной скорости действий**:** подсчитывает количество событий определенного типа в заданном временном окне, используя исторические данные.
    - Детектор аномальных операций**:** сравнивает параметры действия (например, размер выигрыша) с пороговыми значениями или персональным профилем, построенным на основе прошлых событий.
- Для работы JS-правил, требующих исторического контекста, Сервис использует основную базу данных PostgreSQL (или TimescaleDB для оптимизации временных рядов). В базе хранятся сырые и агрегированные события за конфигурируемый период, доступные для запроса из кода правил.
- Механизм реагирования (веб-хуки / очередь команд): при срабатывании правила, Сервис не воздействует на пользователя напрямую. Вместо этого он генерирует структурированное событие-инцидент, содержащее ID пользователя, тип угрозы, оценку риска и контекст, возвращенный JS-функцией и отправляет это событие одним или несколькими способами:

    - Веб-хук (HTTP callback): На заранее настроенный URL защищаемой системы.
    - В очередь RabbitMQ: В специальный exchange/routing key (sentryguard.actions).
    - В Telegram Bot

- Управление правилами**:** предоставляет API для CRUD-операций с JS-правилами: создание, обновление, активация/деактивация. Правила хранятся в БД.
- Сервис предоставляет эндпоинт /metrics в формате Prometheus для подключения внешнего мониторинга
- Все этапы обработки, включая выполнение JS-правил (логи из utils.log()), логируются в структурированном формате (JSON) в stdout.

4. Системные требования к ПО

Минимальные аппаратные требования на один узел обработки:

- Операционная система, способная запускать контейнеры. Предпочтительно Linux.
- Система управления контейнерной виртуализацией. Предпочтительно Docker Swarm или Kubernetes.
- Количество логических ядер процессора: 4
- Семейство процессоров: x86_64
- Частота процессора: 2.5 ГГц
- Объём установленной оперативной памяти: 8 Гб
- Свободное дисковое пространство: 20 Гб

4.1. Минимальные требования к сторонним компонентам и/или системам, необходимым для установки и работы ПО

- Оркестратор/Среда исполнения**:** Docker 24.0+ (open-source community edition).
- Брокер сообщений (обязательный): RabbitMQ 3.11+ (Открытая лицензия Mozilla Public License).
- Хранилище данных (обязательное): PostgreSQL 14+ (Открытая лицензия PostgreSQL License) или TimescaleDB 2.10+ (лицензия Apache 2.0).
- Система мониторинга (опционально**):** Prometheus + Grafana для сбора метрик с /metrics.

4.2. Языки программирования

При разработке ядра Сервиса верификации действий пользователя (SentryGuard) был использован язык программирования Go (GoLang) 1.21+ (Открытая лицензия BSD). Система правил использует интегрированный движок JavaScript.

5. Модули

- **Модуль API-приемника (Ingestion API):** Предоставляет REST API (/api/v1/event) для приема событий. Отвечает за аутентификацию, валидацию и публикацию в RabbitMQ (sentryguard.events.raw). Stateless.
- **Модуль детекции аномалий (Detection Worker):** Основной stateful модуль. Потребляет события из очереди. Загружает из БД активные JS-правила и выполняет их в изолированной среде, передавая событие и историю. При обнаружении угрозы публикует инцидент в очереди для действий (sentryguard.actions) и оповещений (sentryguard.alerts).
- **Модуль реагирования и оповещения (Action Dispatcher):** Потребляет инциденты из очередей sentryguard.actions и sentryguard.alerts. Выполняет конечные действия: вызов веб-хуков и отправку уведомлений в Telegram.