Руководство администратора ПО «Система предотвращения подбора пароля к сервисам REST API»

1.Введение

1.1. Область применения

Настоящий документ предназначен для сотрудников эксплуатирующей организации и отражает основные функциональные возможности и порядок действий при выполнении операций, связанных с администрированием программного обеспечения «Система предотвращения подбора пароля к сервисам REST API» (далее - «Система»)

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Системы входят:

- Установка и настройка Системы
- Реализация планов устранения сбоев и нетиповых нештатных ситуаций
- Выполнение сбора и предоставление в вышестоящую линию технической поддержки информации для воспроизведения технических проблем и выработки решений по их разрешению
- Реализация рекомендаций по устранению нештатных ситуаций, полученных с вышестоящей линии поддержки
- Восстановление работоспособности Системы при сбоях в работе функциональных модулей
- Разработка решения по устранению технических проблем в работе функциональных модулей

1.3. Уровень подготовки администратора/оператора

Администратор/оператор (далее по тексту Администратор) Системы должен обладать знаниями Javascript, уметь пользоваться и настраивать среду функционирования контейнеров или систему оркестрации, используемую на предприятии.

Рекомендуемая численность персонала для эксплуатации Системы — 1 штатная единица.

Администраторы Системы должны пройти обязательную общую и специальную подготовку для работы с Системой.

Общая подготовка должна включать в себя получение знаний и навыков работы с Системой в качестве администратора.

Специальная подготовка должна включать в себя получение знаний и навыков в объеме, необходимом для выполнения своих должностных обязанностей

1.4. Перечень документации

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

- описание функциональных характеристик Системы
- описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения.

2. Установка Системы

В данном разделе будет описана установка Системы на Debian Linux. Предполагается, что были предварительно установлены также Docker, Docker Compose, а также, защищаемая система, содержащая REST API.

2.1. Системные требования к ПО

Минимальные аппаратные требования:

- Операционная система, способная запускать контейнеры. Предпочтительно Linux.
- Система управления контейнерной виртуализацией. Предпочтительно Docker Swarm или Kubernetes.
- Подключение к серверу очередей RabbitMQ
- Количество логических ядер процессора: 4
- Семейство процессоров: x86
- Частота процессора: 3.0. ГГц
- Объем установленной памяти: 16 Гб

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

- Debian 11 (Открытая лицензия GNU)
- Docker 24.0.2 (open-source community edition)
- Grafana Loki 2.6.1 (Открытая лицензия GNU)
- Grafna 9.2.2 (Открытая лицензия GNU)

2.1.3. Языки программирования

При разработке Системы был использован язык программирования GoLang 1.20 (открытая лицензия BSD)

2.2. Порядок установки

1. Создайте папку /home/app
2. Смонтируйте диск с дистрибутивом в папку /mnt
3. Скопируйте из дистрибутива исходники из папки /mnt в папку /home/app
4. Смените текущую папку на /home/app и выполните команды
   sudo chown 10001:10001 ./volumes/loki
   sudo chown 472:472 ./volumes/grafana
5. Отредактируйте файл docker-compose.yml, в соответствии с пунктом 3.2.1 данного документа
6. Создайте и отредактируйте файл настроек для модуля, в соответствии с пунктом 3.2.2 данного документа
7. Смените текущую папку на /home/app и выполните в ней команду
   docker compose -up -d --build
8. Войдите браузером на ваш сервер на порт 3000 в систему мониторинга с пользователем admin и паролем admin. Измените пароль на безопасный.

3. Настройка Системы

3.1. Общие сведения

В данном документе приводятся примеры настройки Системы с использованием среды Docker Compose. Настройка операционной системы, защищаемого сервиса REST API, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

3.2. Модуль проксирования и обработки запросов

3.2.1. Конфигурируемые параметры

Для корректной работы модуля чтения данных из MySQL требуется настроить сервер MySQL как master-сервер для репликации. Модуль также требует настройки следующих переменных окружения:

- HOST — имя хоста и IP, которые будет слушать Сервис. Пример смотрите ниже.
- SETTINGS_FILE — путь к файлу с настройками запросов. Файл подключается к контейнеру с помощью volume. В данной переменной окружения указывается локальный путь внутри контейнера, к которому был примонтирован volume.
- PROTECTED_HOST — адрес защищаемой системы.
- IP_CHECK_INTERVAL — интервал времени для подсчета количества запросов с одного IP.
- IP_CHECK_MAX — предельное количество запросов за интервал IP_CHECK_INTERVAL, приходящих с одного IP. Сервис подсчитывает любые, приходящие с этого IP, запросы. После того, как количество запросов превысит указанное значение, Сервис перестанет проксировать запросы в защищаемую систему и начнет выдавать на поступающие запросы ответ с HTTP статусом 403. По истечении интервала IP_CHECK_INTERVAL, Сервис возобновит проксирование запросов.
- PROTECTED_CHECK_INTERVAL — интервал времени для подсчета количества запросов на авторизацию для одного логина
- PROTECTED_CHECK_MAX — максимальное количество неудачных попыток авторизации для одного логина за период PROTECTED_CHECK_INTERVAL. После того, как количество запросов на авторизацию конкретного логина превысит указанное значение, Сервис перестанет проксировать запросы в защищаемую систему и начнет выдавать на поступающие запросы ответ с HTTP статусом 403. По истечении интервала PROTECTED_CHECK_INTERVAL, Сервис возобновит проксирование запросов.
- METRICS_PORT - порт к подсистеме проверки работоспособности.
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug

Пример настройки модуля:

```
protector:
  build:
    context: ./auth-protect/
  restart: always
  ports:
    - '80:80'
  volumes:
```

```
  - ./auth-protect/config.json:/app/config.json:ro
environment:
  HOST: :80
  PROTECTED_HOST: http://45.8.248.122:8080
  IP_CHECK_MAX: 3
  LOG_LEVEL: trace
  SETTINGS_FILE: /app/config.json
extra_hosts:
  - "host.docker.internal:host-gateway"
```

3.2.2. Настройка маршрутизации

Для того, чтобы Сервис мог предотвратить подбор пароля для конкретного логина, он должен иметь информацию об URL, по которому осуществляется авторизацию пользователя, а также должен уметь обрабатывать запрос чтобы получить из него логин пользователя. Для настройки этой функции, используется настроечный файл, путь к которому указывается в переменной окружения SETTINGS_FILE. Этот файл содержит в текстовом формате json — объект, который в свою очередь, содержит еще два объекта:

- requests — содержит описание маршрутов и обработчики поступающих данных

  Ключами объекта requests являются описатели маршрутов. Они формируются путем конкатенации HTTP-метода, разделителя и относительного пути.

  Префиксом названия команды является HTTP-метод. Поддерживаемые методы: GET, POST, PUT, DELETE. После префикса должен находиться разделитель |. Затем должен быть указан маршрут. Маршрут может включать в себя как параметры, так и символ *, позволяющий обрабатывать различные маршруты одной командой. Примеры:

  - GET|/users
  - GET|/users/:id
  - GET|/users/files/*
  - POST|/users/:id

  Значениями объекта requests являются строки, содержащие в себе JavaScript-код.

  JavaScript-код в конкретном запросе должен заполнять переменную login, чтобы Сервис мог подсчитывать количество неудачных попыток авторизации.

- responses — содержит обработчики ответов, отправляемых защищаемой системой.

  Ключами объекта responses являются описатели маршрутов, описанные выше.

  Значениями объекта responses являются строки, содержащие в себе JavaScript-код.

JavaScript-код в конкретном запросе должен заполнять переменную status. В случае, если значение этой переменной будет меньше 200 или больше 299, Сервис будет считать попытку авторизации неудачной и увеличит счетчик неудачных попыток авторизации на единицу.

Пример настройки маршрутизации:

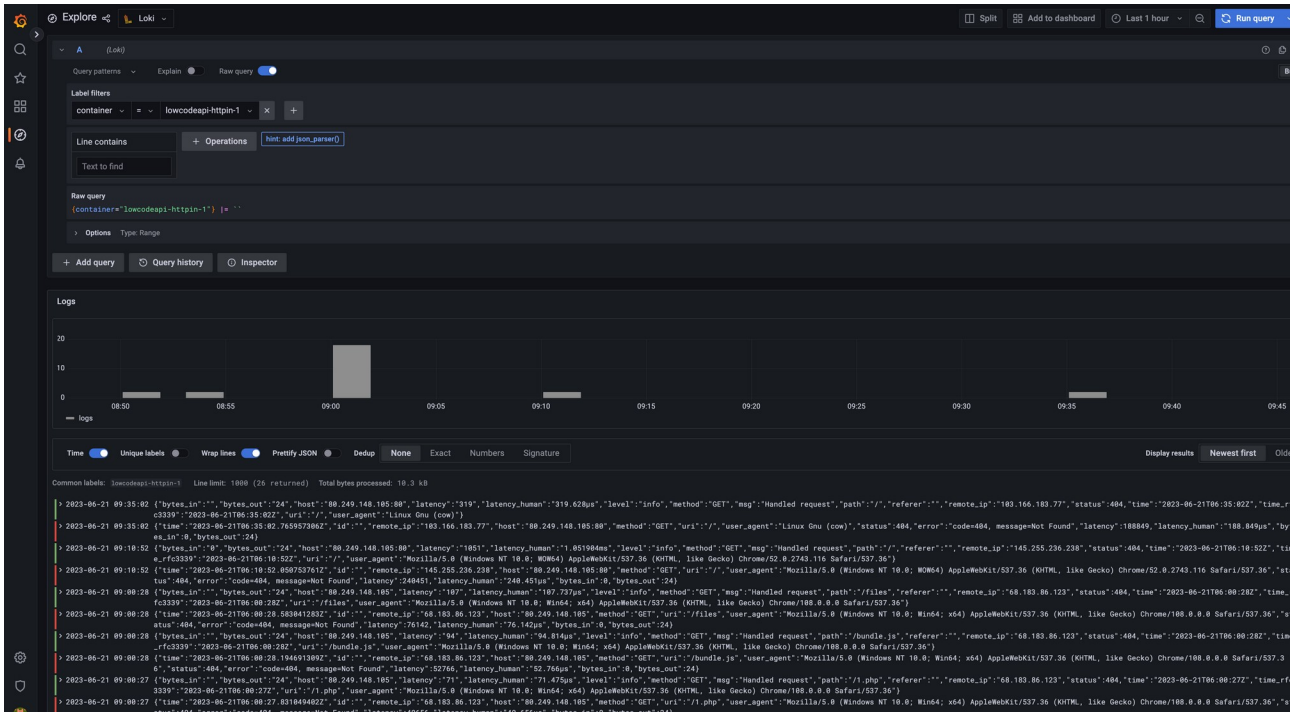
```
{
   "requests": {
      "POST|/login":"str = String.fromCharCode.apply(null, body); console.log(str); js = JSON.parse(str); login = js['login'];"
    },
   "responses": {    }
}
```

4. Система мониторинга

В качестве системы мониторинга используется Grafana Loki — это набор компонентов для полноценной системы работы с логами. Loki-стек состоит из трёх компонентов: Promtail, Loki, Grafana. Promtail собирает логи, обрабатывает их и отправляет в Loki. Loki их хранит. А Grafana умеет запрашивать данные из Loki и показывать их. Loki можно использовать не только для хранения логов и поиска по ним. Весь стек даёт большие возможности по обработке и анализу поступающих данных

Чтобы открыть интерфейс системы мониторинга, перейдите в браузере на IP Вашего сервера и порт 3000. Если Вы входите туда в первый раз, используйте логин admin и пароль admin. После первого входа система попросит Вас изменить пароль на безопасный.

Интерфейс выглядит так:



Выберите в меню пункт «Explore» - Вы увидите страницу поиска логов.

Сам запрос состоит из двух частей: selector и filter. Selector — это поиск по индексированным метаданным (лейблам), которые присвоены логам, а filter — поисковая строка или регэксп, с помощью которого отфильтровываются записи, определённые селектором.

Выберите в разделе Label filters в ниспадающем списке Label значение container, а в ниспадающем списке value выберите нужный контейнер. Выполните запрсо Run query и Вы увидите логи выбранного контейнера.