Universidad Nacional del Litoral

Proyecto Final de Carrera

Diseño de un sistema de detección de anomalías en redes de computadoras.

Informe de avance 4

Pineda Leandro
Córdoba
23 de marzo de 2018

1. Introducción

En las metodologías ágiles el trabajo es dividido en unidades llamadas *sprints* en los que se compromete la implemetanción de un conjunto de funcionalidades que entregan valor a los interesados. Una vez transcurridos cierto números de *sprints* donde se desarrollan las historias planificadas, puede comenzar una etapa adicional comunmente denominada *hardening spint* o *phase*. En esta fase, que comienza una vez que se terminan de implementar todas las funcionalidad y finaliza justo antes de la entrega de una versión final del software, los desarrolladores se enfocan en resolver cualquier problema de integración y ajustar pequeños defectos pendientes de etapas anteriores, sin incluir nuevas funcionalidades.

Este informe pretende mostrar las actividades realizadas en esta última fase de desarrollo. Se describirán las tareas realizadas para corroborar el funcionamiento de todos los componentes, pruebas de longevidad y los cambios realizados en el *frontend* del sistema. Además, se mencionarán modificaciones menores realizadas en el *backend*.

2. Pruebas realizadas

Estrictamente hablando, las pruebas de integración de software son realizadas cuando los componentes individuales de software son combinadas y usados como un todo. Para este desarrollo, y como se describió en el informe anterior, se integraron 3 módulos:

- Un *webservice* que encapsula toda la lógica de detección.
- Redis, una base de datos *clave-valor*.
- Mosquitto, una cola de mensajes.

Las pruebas preliminares mostraron que la integración no presentó problemas. El esfuerzo invertido en refactorizar código permitió definir interfaces sencillas que representan puntos de interacción concretos con el resto de los componentes, y por lo tanto minimizan la ocurrencia de fallos. Estos puntos están representados por los tres flujos que relacionan los módulos del sistema con el mundo exterior. Además, estas interfaces fueron diseñas para ser robustas ante pérdidas de conexión y fallos en la comunicación, dando la capacidad al sistema de recuperarse ante la ocurrencia de ciertos fallos.

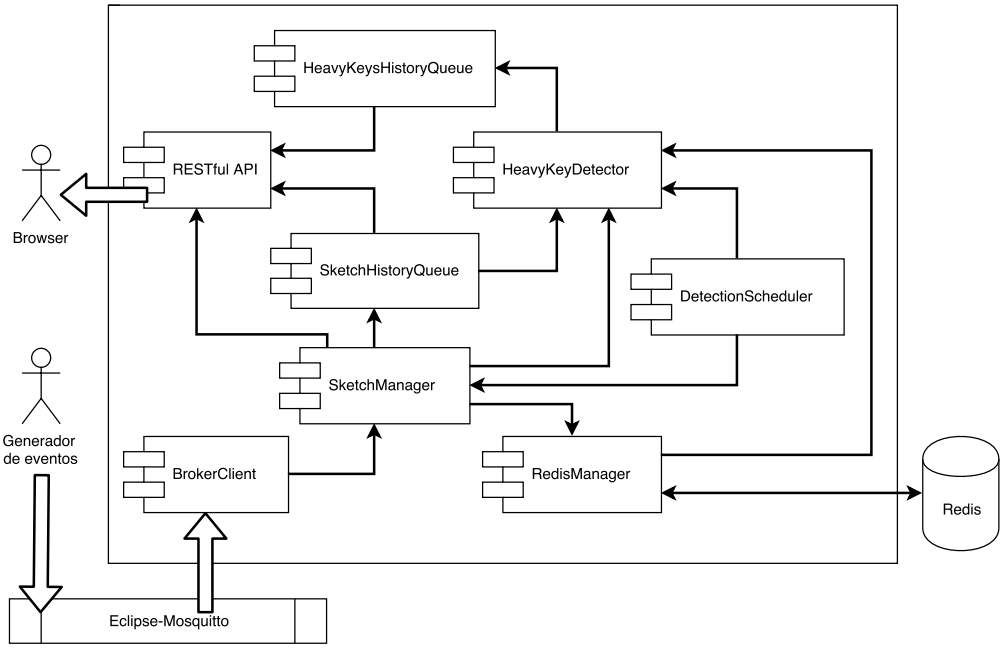

Figura 1: Componentes del sistema y su interacción (informe 3)

Las pruebas de longevidad permiten conocer la estabilidad del sistema. Son de larga duración y permiten corroborar el comportamiento del sistema con una carga apropiada, que simula condiciones reales de uso. Usualmente, se estima cual sería una carga típica para una aplicación dada y se somete a los componentes a un múltiplo de este valor: si cierta aplicación se estima va a generar 100 eventos por segundo, entonces las pruebas de longevidad van a generar 500 eventos por segundo durante un período de varias horas. De esta manera, los desarrolladores tienen cierta confianza en que, si no ocurren fallos cuando el sistema se somete a esta prueba, tampoco ocurrirán en condiciones normales.

Luego de someter al sistema a una carga máxima constante durante 7 horas y media, se observa que:

- Se procesaron más de 218 millones de eventos en 13000 épocas.
- Se identificaron 53063 *heavy hitters* y 3270 *heavy changers*.
- El consumo de memoria se mantiene estable durante toda la prueba.

- En promedio, el sistema procesó eventos a una velocidad de 8100 eventos por segundo.
- La API de *health* del sistema reporta el correcto funcionamiento de todos los componentes durante la duración de la prueba.

Como se mencionó en el informe anterior, se optó por implementar un método de detección que puede ser adaptado a cualquier caso de uso: esto es, lo eventos son tratados como cadenas de caracteres, y luego son transformados a valores enteros (*keys*) que los algoritmos del método necesita para su funcionamiento. Dado que el mismo mostró resultados satisfactorios, no se incluyeron cambios en esta iteración.

Respecto a los cambios en el *backend*, solo se agregó un nuevo recurso a la API para poder mostrar en la interfaz web aquellos eventos que fueron detectados como anómalos para una época dada.

3. Cambios realizados

La mayor parte del esfuerzo en este incremento se destinó en mejorar la interfaz de usuario. La misma está diseñada para mostrar todos los datos del sistema en una única vista, y actualiza la información en tiempo real sin necesidad de recargar la página. Además de incorporar detalles de estilo, se agregaron gráficas para mostrar la evolución ambas *heavy keys*. La cantidad de ocurrencias en un instante de tiempo se observan como barras verticales en el histograma. Cada vez que se selecciona una de esas barras, los eventos identificados como anómalos son mostrados en la lista ubicada a la izquierda del gráfico. Una captura de pantalla de la interfaz puede observarse en la figura 2.

Figura 2: Interfaz web del sistema

De la misma forma que en el incremento anterior, las instrucciones para ejecutar localmente el sistema pueden encontrarse en https://github.com/leandropineda/sketch-service. Las imagenes Docker utilizadas para esta entrega están etiquetadas a la última versión (v4) y se encuentran disponibles para su uso.

4. Conclusiones

La etapa de *hardening* en un desarrollo de software está planteada para ser corta y no incluir grandes cambios en el código. Por esto, el último informe resulta reducido en contenido. Si bien las pruebas realizadas no garantizan que el sistema no presentará fallo alguno, son suficientes para justificar que el mismo se comportará de manera adecuada con alta probabilidad.

Respecto a la interfaz de usuario, si bien es sencilla, presenta todos la información que se obtiene de los algoritmos de detección de una manera resumida y concisa. Esto fue pensado así dado que el usuario debe poder entender con un simple vistazo la información que está siendo procesada en tiempo real.