Cuestiones Práctica 1 Servicios Telemáticos

Carlos Ortega Marchamalo y Pablo Collado Soto

1 Esquema de la actualización de datos mediante SSE (Server-Sent Events)

Para analizar este paradigma de actualización de datos hemos recurrido a *w3schools*. Tal y como se explica lo primero que tenemos que hacer es comprobar que existe soporte para SSE. Si ésto se cumple, entonces procedemos a crear un objeto "EventSource", que no es más que un objeto que "escucha" al elemento identificado por la URL que le especifiquemos esperando "oír" un evento. Este elemento puede ser un script escrito en JavaScript, por ejemplo. En caso de que se produzca un evento, se llamará al método "onmessage" de "EventSource". Para que todo funcione, tenemos que asignarle una función a dicho método, de manera que sea llamada cuando recibamos un evento. Cabe destacar que a esta función se le pasa el parámetro "event". Este parámetro contiene la información del evento que estamos recibiendo y el propio objeto EventSource se tiene que encargar de pasárselo a "onmessage", de lo contrario no podríamos acceder a la información del evento y solo podríamos responder con información local a los mismos, pero no se podría intercambiar información. Sabríamos que ha ocurrido un evento, pero no tendríamos ningún otro tipo de información...

La "magia" de SSE ocurre en la función que le asignamos a "onmessage". Aquí podríamos hacer operaciones sobre cualquier elemento del documento HTML tal y como solemos hacer. Podemos localizar los elementos en base a su ID y cambiar el texto que contienen entre otras cosas. Esta acción es la que nos ocupa.

Tal y como se puede apreciar, el escenario anterior se traduce casi de manera literal para nuestro caso. Debemos definir un proceso en nuestro servidor cuya interfaz con el cliente sea un elemento localizable mediante una URL como por ejemplo un script escrito en JavaScript. Al descargar el cliente la página le "dirá" al servidor que cada vez que este script genere un evento se lo pase, es decir, el cliente empieza a escuchar a esta URL. Cada vez que se genere un evento se actualizará el script de JS y enviará dicho evento. El cliente, que estaba escuchando, lo recibirá y llamará al método "onmessage" que ya habíamos definido en la primera página que se bajaba el cliente. Tendremos que "buscar" los datos a actualizar, que vendrán dentro del propio evento y colocarlos en un elemento de HTML que tengamos definido para este fin. Al poder "buscar" información dentro del evento podemos actualizar varios campos con un solo mensaje del servidor. Para un solo dato ésto es insignificante, pero

cuando hay que hacer muchas modificaciones este aspecto pasa a ser vital.

2 *Hardening* del servidor

Tal y como se recomendaba en el enunciado de la práctica hemos optado por intentar hacer nuestro servidor un poco más seguro. Para ello hemos modificado la configuración general del servidor localizada en */etc/apache2/apache2.conf*. Una de las medidas que hemos tomado ha sido establecer que nuestro servidor no sea "indexable". Anteriormente si akguien introducia la URL de una ruta que colgara de la raíz del servidor válida podía acceder a todos los archivos. Tras deshabilitar dicha opción en la configuración de la raíz del sitio web hemos conseguido que, de no introducir la URL de un archivo visible salte un error **403 Forbidden**. Lo que es más, hemos optado por ocultar la "firma" del servidor en estas páginas de error. Si nos fijamos, esta página contiene la versión del servidor además del tipo, lo que supone una facilidad para alguien que quiera adquirir información con idea de perpetrar un ataque. Hemos hecho algo análogo para el campo de cabecera "server" de las respuestas HTTP que manda nuestro servidor. Ahora en vez de contener la versión podemos ver que solo se contiene la palabra "Apache" en dicho campo, lo que supone menos información "gratuita". Esto se puede comprobar monitoreando el tráfico intercambiado con WireShark. Finalmente, hemos optado por deshabilitar las configuraciones particulares de cada carpeta accesible a los usuarios a través del documento *.htaccess*. Preferimos mantener todas las configuraciones centralizadas bajo */etc/apache2*.

Tras investigar un poco hemos visto que existe un ataque relativamente sencillo contra servidores de Apache. Se llama **SlowLoris**, y se basa en mandar peticiones de HTTP parciales con lo que consigue mantener las conexiones vivas y así agotar los recursos del servidor. Hemos intentado mitigarlo a través de la configuración del módulo *Request Timeout* que viene instalado por defecto en Apache 2.4, nuestra versión. No obstante, por mucha configuración que hiciéramos hemos sido incapaces de contrarrestar el ataque que lanzábamos nosotros mismos contra el servidor. A pesar de esto, hemos aumentado nuestro conocimiento sobre la seguridad en HTTP y hemos visto algunos de los problemas más comunes de nuestra arquitectura de servidor. La configuración se puede encontrar en */etc/apache2/apache2.conf*.

3 Notas sobre el uso de AppCache

Tras rascarnos la cabeza durante un buen rato hemos visto que los principales navegadores actuales han "deprecado" la utilización del AppCache. Lo que es más, a no ser que la página que esté cargada sea segura (es decir, que use HTTPS) esta funcionalidad está deshabilitada. Por ello, configuremos lo que configuremos no vamos a ver ningún cambio; no aparecerá nada en la caché del navegador... Todo esto se ha podido comprobar gracias a las herramientas de desarrollador de los distintos navegadores.

A pesar de que no sea necesario, hemos incluido los archivos requeridos para implementar esta característica en caso de que pudiera ejecutarse.