Using JSON Securely

# Utilizar JSON con seguridad

JSON es especialmente fácil de utilizar en aplicaciones web porque JSON es JavaScript. Un texto JSON puede convertirse en una estructura de datos útil con la función eval:

var myData = eval('(' + myJSONText + ')');

(La concatenación de los paréntesis alrededor del texto JSON es una solución para una ambigüedad en la gramática de JavaScript).

Sin embargo, la función eval tiene terribles problemas de seguridad. ¿Es seguro utilizar eval para analizar un texto JSON? Actualmente, la mejor técnica para obtener datos de un servidor en un navegador web es a través de XMLHttpRequest. XMLHttpRequest sólo puede obtener datos del mismo servidor que produjo el HTML. eval ing texto de ese servidor no es menos seguro que el HTML original. Pero eso supone que el servidor es malicioso. ¿Y si el servidor es simplemente incompetente?

Un servidor incompetente podría no realizar correctamente la codificación JSON. Si construye textos JSON juntando algunas cadenas en lugar de utilizar un codificador JSON adecuado, podría enviar involuntariamente material peligroso. Si actúa como un proxy y simplemente pasa el texto JSON sin determinar si está bien formado, entonces podría volver a enviar material peligroso.

El peligro puede evitarse utilizando el método JSON.parse en lugar de eval(ver [http://www.JSON.org/json2.js)](http://www.JSON.org/json2.js). JSON.parse lanzará una excepción si el texto contiene algo peligroso. Se recomienda utilizar siempre JSON.parse en lugar de eval para defenderse de la incompetencia del servidor. También es una buena práctica para el día en que el navegador proporcione un acceso seguro a los datos de otros servidores.

Existe otro peligro en la interacción entre datos externos y innerHTML. Un patrón Ajax habitual es que el servidor envíe un fragmento de texto HTML que se asigna a la propiedad innerHTML de un elemento HTML. Esta es una práctica muy mala. Si el texto HTML contiene una etiqueta <script> o su equivalente, se ejecutará un script maligno. Esto también podría deberse a la incompetencia del servidor.

¿Cuál es el peligro en concreto? Si un script maligno consigue ejecutarse en tu página, obtiene acceso a todo el estado y capacidades de la página. Puede interactuar con tu servidor, y éste no podrá distinguir las peticiones malignas de las legítimas. El script maligno tiene acceso al objeto global, que le da acceso a todos los datos de la aplicación excepto a las variables ocultas en los cierres. Tiene acceso al objeto documento, que le da acceso a todo lo que ve el usuario. También da al script maligno la capacidad de dialogar con el usuario. La barra de direcciones del navegador y todo el cromo antiphishing indicarán al usuario que debe confiar en el diálogo. El objeto documento también da al script maligno acceso a la red, permitiéndole cargar más scripts malignos, o sondear sitios dentro de tu cortafuegos, o enviar los secretos que ha aprendido a cualquier servidor del mundo.

Este peligro es consecuencia directa del objeto global de JavaScript, que es de lejos la peor parte de las muchas partes malas de JavaScript. Estos peligros no están causados por Ajax o JSON o XMLHttpRequest o Web 2.0 (sea lo que sea eso). Estos peligros han estado en el navegador desde la introducción de JavaScript, y permanecerán hasta que JavaScript sea sustituido o reparado. Ten cuidado.