¡Recuérdame! Auto log-in web

Implementación para mantener las sesiones de los usuarios web iniciadas y consideraciones en cuanto a su seguridad.

Escrito por Alejandro Fernández

A Junio, 2019

INTRODUCCION

Permitir a los usuarios de una web mantener su sesión iniciada puede ser de gran utilidad para mejorar la usabilidad de una página. Pero esto puede ir acompañado de disgustos si no se lleva a cabo correctamente.

A día de hoy la mayoría de navegadores ofrecen la posibilidad de recordar contraseñas y usuarios para hacer el inicio de sesión lo más rápido posible, pero, ¿es esto una buena idea? El problema es que si no se tiene mucho cuidado de la implementación esta puede ser fácilmente vulnerable.

Por otro lado los gestores de contraseñas una buena alternativa. Lo malo es que según una encuesta publicada por cyclonis.com en 2018 revela que solo un 12% usan aplicaciones de este tipo, y un 20% más confían en sus navegadores web para esta tarea.

IMPLEMENTACIÓN

La implementación más básica consiste en guardar cierta información en una cookie en el dispositivo del cliente. y que esta sirva para verificar que el usuario dice ser quien es.

Esto quiere decir que quien tenga la cookie, podría hacerse pasar por el usuario legítimo de esta. Esto es lo que se llama secuestro de sesiones (session kidnapping) y se lleva a cabo mediante ataques como:

- Envío de cookies mediante http. Al enviarse mediante una petición http y no https cualquier usuario que puede estar entre medias (man in the middle attack) puede hacerse con la cookie con extrema facilidad. En algunas ocasiones hay cookies que se pueden enviar mediante https aunque la conexión se trate de https, por lo que es algo a tener en cuenta.
- Cross-site scripting o XSS es un ataque con el que se consigue ejecutar código javascript en una página externa. Mediante esto se puede llevar a cabo un robo de cookies.
- Malware o spyware en un dispositivo podría robar todas las cookies de los navegadores y enviarlas a un atacante con fines maliciosos.

Por la inherente inseguridad de las cookies nunca deben contener información vital de ningún tipo, mucho menos directamente contraseñas. Lo ideal es que sea algo estrictamente arbitrario. Aquí es cuando entran en juego los tokens, una serie de caracteres única generada aleatoriamente de forma criptográfica. A día de hoy la longitud recomendada para un token es de 128bits (16 bytes). Esto es suficiente para que en la actualidad sea impensable hacer un ataque de fuerza bruta.

Teniendo esto claro, aquí va un ejemplo del sistema en acción:

1. El usuario inicia sesión activando la casilla de "Recordar sesión".
2. El servidor verifica al usuario, y como ha marcado la casilla, genera un token, por ejemplo XXXXXXXXX, guarda este hash en la base de datos junto con su ID de usuario y crea una cookie llamada "rememberMe"=" userID;XXXXXXXXX" que envía al usuario, donde userID es su ID de usuario.
3. Cuando el usuario al día siguiente quiera volver a entrar a la web sin tener la sesión iniciada, el servidor recibirá la cookie "rememberMe", y si el hash del token XXXXXXXXX coinciden con el ID de dicho usuario en su base de datos iniciará la sesión automáticamente.

Es importante recordar que el token debe siempre guardarse como hash ya que este funciona esencialmente como si fuera una contraseña.

Como nota importante, asegurarse de usar siempre una funcion timing-safe al comparar hashes para evitar posibles timing-attacks.

Problemas: Todavía es vulnerable al robo de cookies, esto se discutirá en el siguiente apartado.

El ID de usuario es cualquier identificador único que señale a un solo usuario y pueda ser público. Aunque usar tan solo el hash sin estar acompañado del ID de usuario podría ser suficiente, es recomendable que siempre vayan juntos para fortalecer posibles ataques de fuerza bruta.

Si se diese el caso de que el servidor recibiese una cookie que contenga un ID con un token incorrecto es posible que se trate de un intento de falsificación

o una cookie anteriormente robada y quizás se deba informar al usuario y eliminar los tokens existentes.

Como medida extra de seguridad se puede añadir junto con la cookie un hash cifrado del contenido de la cookie con una contraseña que solo el servidor conozca, y al recibir la cookie verificar también este hash de forma inversa. De esta forma se hace imposible la falsificación de una cookie y poder identificar posibles ataques de fuerza bruta.

SEGURIDAD

Generalmente, cuanta mayor sea la seguridad, peor será la experiencia del usuario. Hay páginas que se pueden permitirse más riesgos a favor de la facilidad de uso, como por ejemplo, una página de preguntas y respuestas, mientras que para otras, como es el caso de aplicaciones bancarias, la seguridad es siempre lo primero.

Como una recomendación general sería conveniente establecer una fecha de caducidad al token (tanto en el cliente en su cookie como en el servidor). Un tiempo recomendado para una web estándar podría ser de un mes a tres meses. Además siempre es una buena idea requerir que el usuario escriba de nuevo su contraseña para hacer cambios u operaciones críticas.

Algunos métodos para aumentar la seguridad y tratar de verificar la legitimidad de una cookie son:

- La dirección IP suele ser una forma destacable para tratar de verificar a una persona en la red, estas son generalmente únicas para cada usuario y difícilmente suplantables. El problema es que estás cambian frecuentemente (semanalmente o incluso, en algunos casos extremos, diariamente) lo que dificulta su uso para identificar a un usuario. También en el caso de que el usuario cambie frecuentemente de puntos de acceso red imposibilitaría este método. Por esto usar estrictamente una comparación de las direcciones IP solo es recomendable para las webs con necesidades de seguridad más estrictas. Para el resto puede afectar significativamente la experiencia de usuario.

Lo bueno es que a partir de una dirección IP podemos derivar más información del usuario como el país, localidad, el proveedor de internet, o el ISP desde el que se ha realizado la petición. Que un usuario se conecte a una web desde una cafetería o trabajo con su portátil es algo habitual, mientras que si se conecte desde un país en el otro lado del mundo, no lo es. Es por esto que usar una información más genérica como el área geográfica, puede ser de gran utilidad para tratar de detectar un acceso no legítimo.

El Agente de Usuario (User Agent) es una cadena de caracteres que mandan habitualmente los navegadores para indicar a las webs el navegador, su versión y SO que se está usando para acceder. Su sintaxis es la siguiente:

Mozilla/<version> (<system-information>) <platform> (<platform-details>) <extensions>

Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36

(ver https://developer.mozilla.org/es/docs/Web/HTTP/Headers/User-Agent)

De la cadena superior podemos sacar que se está accediendo desde Chrome 51 en Windows 7 de 64bits. La versión del navegador cambia frecuentemente y no debería preocuparnos, salvo que se detectara una versión anterior a la del último acceso o una gran diferencia de versiones, que sería sospechoso. Por otro lado, como las cookies son dependientes de cada navegador, el navegador usado y SO debería permanecer siempre constante.

Nota: El Agente de Usuario es fácilmente falsificable por un atacante.

El Accept-language header nos proporciona los lenguajes establecidos por el usuario como predeterminado del navegador. Este dato se envía con cada solicitud del cliente: *es,en;q=0. 9*

Todos los datos anteriores los podemos obtener directamente desde la primera solicitud que haga un cliente a nuestra web, facilitando enormemente su implementación. Pero si todavía necesitamos más

seguridad podemos obtener más información de nuestros usuarios mediante Javascript.

Con Javascript podemos obtener más información del usuario como su zona horaria o la resolución de la pantalla, los plugins instalados en el navegador, que fuentes están instaladas en el equipo, la versión de WebGL, a veces, incluyendo el nombre del driver gráfico, y otros datos más.

Con todos estos datos podemos construir una huella bastante detallada de un usuario legítimo, y con esto, poder detectar a los atacantes con facilidad. Aun así, debemos tener en cuenta que hacer una huella muy detallada de nuestros clientes puede considerarse algo poco ético y que atenta contra la privacidad de los usuarios. Solo es recomendado usarlo para las partes más críticas de una aplicación.

(ver https://amiunique.org/)

Debemos ser conscientes de la seguridad que requiere nuestra web y encontrar un equilibrio entre la seguridad y usabilidad. Para los casos de seguridad más extremos, lo recomendable es no usar ningún sistema que permita dejar la sesión abierta (lo recomendable es recordar únicamente el usuario o email).

DIAGRAMA DE UNA POSIBLE IMPLEMENTACIÓN USANDO PHP