Introducción a seguridad de las aplicaciones

Nos enfocamos en saber sobre vulnerabilidades en software, centralizado en el mundo de las aplicaciones.

Autenticación, autorización y control de acceso

Autenticación

- un usuario tiene que demostrar que es quien dice ser
- En la autenticación mediante usuario y contraseña se encuentran deficiencias que hacen que si las contraseñas no son robustas pueden ser vulnerables a ataques de fuerza bruta. Además las contraseñas deben ser guardadas cifradas. Si se usa un medio no seguro es posible que la contraseña no llegue encriptada.
- En la autenticación mediante tarjetas inteligentes se usa un certificado digital para la autenticación, pero se necesita un medio físico para su uso.
- Un certificado digital es como un pasaporte virtual. Consta de una pareja de claves criptográficas, una pública y una privada, creadas con un algoritmo matemático. El titular del certificado debe tener la clave privada.

Tipos de criptografía

- Simétrica: Clave pública y clave privada
- Asimétrica: Se usa la misma clave para cifrar y descifrar. El tener que distribuir la clave es problemático.

Autorización

Una vez autenticado el usuario entra en juego la política de autorización, que define que puede y que no puede hacer el usuario. Dependiendo del usuario se tendrá acceso a unos elementos u otros.

Control de acceso

Se hace cargo de controlar a que pueden acceder los usuarios.

Aplicaciones y servicios web con estado (stateful)

Son las aplicaciones más tradicionales, el servidor mantiene info de cada uno de los usuarios que se han conectado. Se crea una sesión para cada usuario la primera vez que se conecta, almacenando la información que se considere apropiada. Una sesión es una estructura de mapa en la que se almacenan atributos identificados por nombre, generalmente se almacena información de usuario, sus permisos, etc…

Se usa el API de servlets de java para las sesiones, que nos devuelve la sesión actual (HttpSession) usando una petición de cliente: HttpServetRequest.

- HttpSession HttpServletRequest.getSession(boolea create): Permite obtener la sesión asociada a la petición actual.
- HttpSession.setAttribute(String name, Object value): almacena un objeto en la sesión identificado por un nombre
- HttpSessión.getAttribute(String name): Permite obtener un objeto almacenado.

Para identificar a que cliente pertenece cada petición se utilizan las cookies, que suelen ir en la cabecera de una petición http. Estas cookies se almacenan en el navegador. Para identificar la sesión del usuario se usele usar la cookie JSESSIONID o ASP.NET_SessionId.

Las cookies son un método tradicional de almacenamiento de información del cliente y se usa principalmente para:

- Mantener la sesión: info del usuario, carrito de la compra, etc...
- Personalización: preferencias de usuario y ajustes
- Seguimiento: Seguimiento y análisis de comportamiento del usuario.

Un servidor envía las cookies al cliente con la cabecera HTTP: Set-Cookie El cliente envía las cookies al servidor en la cabecera HTTP: Cookie

```
Set-Cookie: <cookie-name> =<cookie-value>
```

Las cookies pueden tener varios atributos:

- Expires: Fecha en la que la cookie caduca
- Secure: se enviará la cookie por HTTPs
- HttpOnly: No serán accesibles con javascript

```
Set-cookie: User=admin;  secure; HttpOnly
```

- Domain: especifica los subdominios a los que se enviará la cookie.
- Path: Indica los subdirectorios a los que se enviará la cookie
- Expires: Fecha en la que expira
- Max-Age; número de segundos que puede durar la cookie
- SameSite: Un sitio web instala 2 cookies en el navegador, una normal y una con exención. El objetivo es que si llega un email malicioso con un enlace a un dominio que trata de suplantar el verdadero. La idea es que el navegador detecte que el sitio web es falso y no envíe la cookie de sesión al estar esta marcada con SameSite. Existen varios parámetros como **Strict** que solo envía la cookie desde el mismo dominio y **lax** que permite el envío de la cookie desde un dominio externo cuando la petición cambia visiblemente de URL.

Las cookies de terceros son las cookies que instala un sitio web diferente del que se está visitando, por ejemplo, las de publicidad.

Cuando el navegador no soporta cookies, una alternativa que permite manejar la sesión es la técnica de reescritura de URL (URL rewriting). En la actualidad esta funcionalidad se considera una vulnerabilidad. Mediante esta técnica el servidor modifica la URL solicitada por el cliente añadiéndole al final el identificador de sesión.

```
patata.es/docs/web/index.html
patata.es/docs/web/index.html;jsessionid=3642AKJDN
```

La desventaja de los servicios web statefull es que se incrementa el uso de memoria usada por el servidor y presenta dificultades a la hora de escalar servicios. **Sticky Sessions** es una técnica de replicación de aplicaciones con estado en la que el balanceador de carga es capaz de asignar siempre el mismo servidor al mismo cliente.

Aplicaciones y servicios web sin estado (stateless)

En este tipo de aplicaciones el servidor no almacena información sobre el estado del cliente. La información usada para identificar al usuario se envía en primer término por el servidor una vez se ha realizado el proceso de autenticación. Se suelen usar Tokens JWT que permiten implementar de forma eficiente y segura las comunicaciones del servidor.

Aplicaciones Web tradicionales y SPA