Universidad de Oviedo

EPI GIJÓN

Universidad de Oviedo

Seguridad

Enol García González
Universidad de Oviedo
10 de noviembre de 2025

Contenidos

Conceptos de seguridad

- **Autenticación**. Asegurar que el usuario es quien dice ser.
- **Autorización**. Verificar que tiene acceso al recurso que solicita.
- **Auditoria**. Registrar todos los accesos de usuarios a recursos.

Autenticación

Para verificar que el usuario es quien dice ser necesitamos:

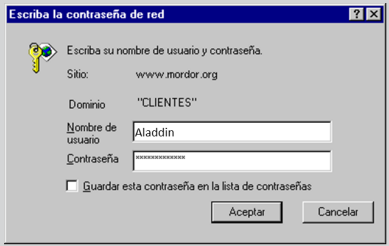
- **Algo que el usuario sabe**, como un pin o una contraseña
- **Algo que el usuario tiene**, como una llave de seguridad o un teléfono
- **Algo que el usuario es**, como su huella dactilar.
- **Algo que el usuario hace**, como por ejemplo el port knocking

Actualmente, es muy común la autenticación multifactor en la que se pide al menos 2 técnicas para verificar al usuario.

Autenticación

- La autenticación está disponible de formas muy primitivas desde HTTP 1.0 mediante un campo de la petición que se llama Authorization.
- En la versión 1.0 sólo existía el método BASIC que consistía en codificar en base 64 el nombre de usuario y la contraseña para verificar si el usuario tiene acceso al recurso.
- Los evidentes fallos de seguridad hicieron que la versión 1.1 de HTTP ya incluyese otro método conocido como DIGEST que incorpora el timestamp de la petición antes de hacer codificar en base 64 para un poco más de seguridad.
- Actualmente, a penas se usan estos métodos de autenticación y se desarrolla un método de autenticación programáticamente dentro de la aplicación.

Autenticación



Base64

```
GET /privado/index.php HTTP/1.1
Host: example.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

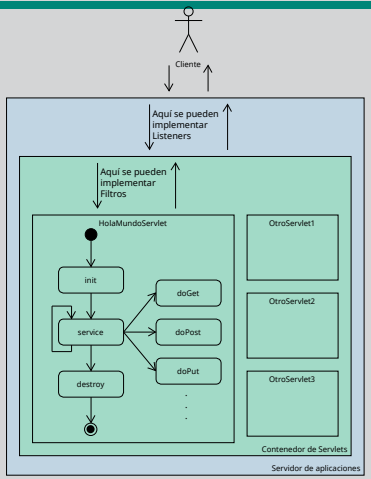
Autorización

RECUERDO: La autorización es el proceso de verificar si un usuario puede acceder a un recurso.

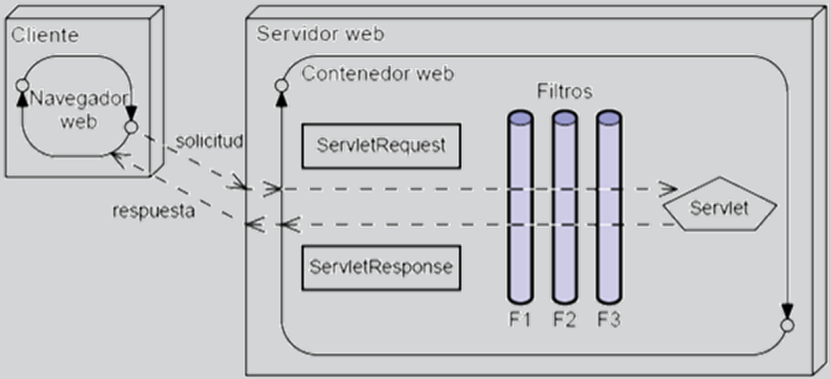
Aquí tenemos que tener en cuenta dos niveles:

- Servidor. Puede implementar diferentes sistemas de control de accesos: **Access Control List (ACL)**, **Discrectional Access Control (DAC)**, Mandatory Access Control (MAC), Role-based Access Control (RAC)
- Aplicación. A nivel de aplicación el control se hace sobre las vistas que puede ver un usuario. En JSF tenemos dos métodos para implementarla:
  - Filtros Servlet
  - JSF Listeners

Filtros



Filtros



Filtros

```
@WebFilter(dispatcherTypes={DispatcherType.REQUEST}, urlPatterns={"/*"})
class LoginFilterServlet implements Filter {
  public void doFilter(ServletRequest res, ServletResponse res, FilterChain ch)
      throws [...] {
    if (req.getSession().getAttribute("USER") == null) {
      res.sendRedirect("/login")
      return;
    }
    ch.doFilter(request, response);
  }
}
```

Filtros

Como casi todo en JSF se pueden declarar nuevos filtros de dos formas:

- Con la anotación WebFilter sobre la clase
- Con la etiqueta <filter> en el fichero web.xml

Filtros

No es obligatorio establecer ningún parámetro para declarar el filtro, pero es comun utilizar:

- **dispatcherType** para establecer que tipo de peticiones pasarán por el filtro. Puede ser: **REQUEST**, ERROR, FORWARD o INCLUDE.
- **urlPatterns** para definir que direcciones URL provocan que se procese el filtro y cuales están exentas. Por defecto todas procesan el filtro.
- **initParams**, que permite reutilizar un filtro con diferentes parámetros. En caso de establecer esta opción es obligatorio redefinir el método init que recibe los parámetros de inicio y definir como los almacena.

JSF Listener

Los listeners son clases que se ejecutan cuando sucede un evento. Hay 3 ámbitos de listeners:

- **Petición** que se crea heredando de la clase *ServletRequestListener*
- **Sesión** que se crea heredando de la clase *ServletSessionListener*
- **Contexto** que se crea heredando de la clase *ServletContextListener*

Con el listener de petición se puede modificar la petición antes y después de ser procesada, de forma similar a los filtros.
En comparación con los filtros, los listeners suelen ser una peor opción porque no tienen la misma capacidad de recibir parámetros y configurarse.

JSF Listener

Para crear un listener hay que:

1. Crear una nueva clase heredando de la clase correspondiente según el ámbito de listener que queramos crear.
2. Registrar el listener, que se puede hacer de dos formas:
   - Con la anotación @WebListener sobre la clase.
   - En el fichero web.xml con la etiqueta <listener>

HTTPS

- ¿Por qué HTTPS si ya existía HTTP?
- ¿Qué es HTTPS? HTTP "Secure"
- ¿Qué es seguro en HTTPS?

HTTPS

- Lo característico de HTTPS es que encriptan los datos de forma privada entre el servidor y el cliente.
- para la encriptación se utiliza un certificado SSL/TLS. Estos certificados pueden ser emitidos por cualquiera, incluso por mí.
- Para reforzar la seguridad los navegadores suelen marcar como no seguras las webs que no tienen un certificado válido emitido por una entidad certificadora reconocida.

Tipos de certificado

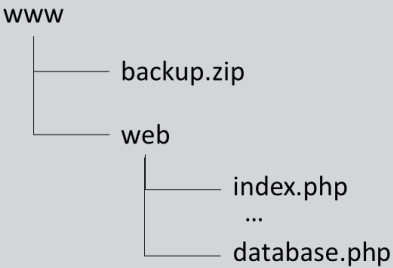
Dentro de los certificados emitidos por una entidad certificadora también hay 3 tipos:

- **Certificado con validación de dominio (DV SSL/TLS)**. Se valida que soy el propietario del dominio. Por ejemplo: www.uniovi.es
- **Certificado con validación de organización (OV SSL/TLS)**. Se valida que soy una organización y el certificado puede usarse en todos los dominios que tenga. Por ejemplo: campusvirtual.uniovi.es, sies.uniovi.es, intranet.uniovi.es, universidad-oviedo.es
- **Certificado con validación extendida (EV SSL/TLS)**. Es el certificado más estricto. Lo extiende una entidad certificadora tras hacer una comprobación rigurosa de todos los datos de la empresa.

Últimas consideraciones

- Todo software tiene vulnerabilidades de seguridad. Estamos utilizando servidores de aplicaciones con vulnerabilidades CONOCIDAS.
  - Vulnerabilidades de Wildfly 27
  - Vulnerabilidades de Postgresql
- Si ocultamos que servidor tenemos dificultaremos que se exploten esas vulnerabilidades.
- Desde el punto de vista de sistemas, hay que poner cortafuegos para que en caso de que las encuentren y la exploten los daños sean los menores posibles.

Casos reales



Normalmente se accede a
https://www.xxxxxxxx.com/web
¿Qué pasa si edito la url por https://www.xxxxxxxx.com? ¿Y por
https://www.xxxxxxxx.com/backup.zip?

Casos reales

```
<div id="eliminar">


<p id="rest" hidden>https://[redacted].asturias.es/[redacted]</p>
<p id="user" hidden>[redacted]</p>
<p id="pass" hidden>[redacted]</p>
<p id="url" hidden>https://[redacted].asturias.es/</p>


</div>
```