

Escuela Politécnica de Ingeniería  
Grado de Ingeniería Informática en Tecnologías de la  
Información

# Tecnologías Web

## Tema 4

Seguridad y aspectos legales en la Web



# Índice

- Autenticación/Autorización y Auditoría
- Ley de protección de datos

# Conceptos

- Autenticación (o acreditación)
  - 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

# Métodos de Autenticación

- Se basan en [...] el usuario
  - algo conocido por
  - algo poseído por
  - una característica física de
  - algo que hace
- Puede haber combinaciones de varios
  - Autenticación multifactor

# AAA en aplicaciones Web

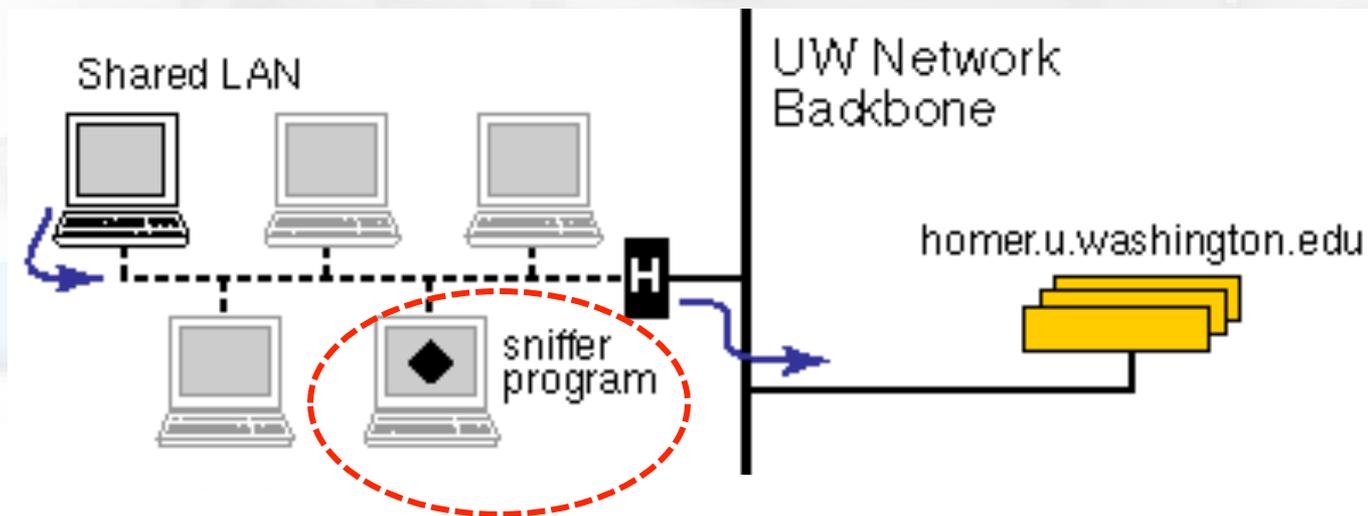
- Autenticación
  - Se delega en sistemas PAM
    - Pluggable Authentication Modules
  - Se puede emplear infraestructura PKI
- Autorización
  - Por contenedor, declarativa
  - Por aplicación, programática
- Auditoria
  - Por aplicación
  - Uso de frameworks

# Problemas de seguridad en redes

- Suplantación de servidores
- Suplantación de la identidad de un usuario
- Tráfico entre cliente y servidor
  - Monitorizarlo
  - Corromperlo
  - Desviarlo
  - Bloquearlo
- ...

# Transmisión en Web

- El tráfico en Internet puede ser espiado y alterado con mucha facilidad
- ¿Cómo intercambiar datos confidenciales?



# Packet Sniffing

The screenshot shows the HttpDetect application window. The main window displays a table of captured HTTP requests. The table has columns for No., Time, Client (IP:PORT), Server (IP:PORT), URL, File Size, and Status. Row 77 is highlighted, showing a request for /images/logo\_ibm.gif from 192.168.1.3:3061 to www.etherdetect.com:80. Below the table, the HTTP Request Header and HTTP Response Header for the selected request are displayed in plain text.

| No. | Time  | Client (IP:PORT)  | Server (IP:PORT)        | URL                            | File... | Status    |
|-----|-------|-------------------|-------------------------|--------------------------------|---------|-----------|
| 69  | Ju... | 192.168.1.3 :3057 | www.effetech.com :80    | /                              | 35660   | FIN, 200  |
| 70  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/style.css              |         | FIN, 304  |
| 71  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/logo_main.jpg          |         | FIN, 304  |
| 72  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/chinese_edition.gif    |         | FIN, 304  |
| 73  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/space.gif              |         | FIN, 304  |
| 74  | Ju... | 192.168.1.3 :3059 | www.effetech.com :80    | /images/arrow_small.gif        |         | FIN, 304  |
| 75  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/award_tucows_4ratel... |         | FIN, 304  |
| 76  | Ju... | 192.168.1.3 :3060 | www.etherdetect.com...  | /images/logo_ms.gif            | 628     | FIN, 200  |
| 77  | Ju... | 192.168.1.3 :3061 | www.etherdetect.com...  | /images/logo_ibm.gif           | 1217    | FIN, 200  |
| 78  | Ju... | 192.168.1.3 :3059 | www.effetech.com :80    | /images/award_FileHungry_5s... |         | FIN, 304  |
| 79  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/award_softwareseeke... |         | FIN, 304  |
| 80  | Ju... | 192.168.1.3 :3061 | www.etherdetect.com...  | /images/logo_mit.gif           | 259     | FIN, 200  |
| 81  | Ju... | 192.168.1.3 :3060 | www.etherdetect.com...  | /images/logo_ms.gif            |         | Requested |
| 82  | Ju... | 192.168.1.3 :3059 | www.effetech.com :80    | /images/award_webaward2002e... |         | FIN, 304  |
| 83  | Ju... | 192.168.1.3 :3058 | www.effetech.com :80    | /images/ed_small.gif           | 24269   | FIN, 200  |
| 84  | Ju... | 192.168.1.3 :3061 | www.etherdetect.com...  | /images/logo_cornell.gif       | 2027    | FIN, 200  |
| 85  | Ju... | 192.168.1.3 :3059 | www.effetech.com :80    | /images/flag_detail.gif        | 1026    | FIN, 200  |
| 86  | Ju... | 192.168.1.3 :3061 | www.etherdetect.com...  | /images/logo_reuters.gif       | 1822    | FIN, 200  |
| 87  | Ju... | 192.168.1.3 :3059 | www.etherdetect.com :80 | /images/flag_demo.gif          | 1013    | FIN, 200  |
| 88  | T...  | 192.168.1.3 :3058 | www.etherdetect.com :80 | /images/flag_demo.gif          | 1013    | FIN, 200  |

**HTTP Request Header**

```
GET /images/logo_ibm.gif HTTP/1.1
Accept: */*
Referer: http://www.etherdetect.com/
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: www.etherdetect.com
Connection: Keep-Alive
```

**HTTP Response Header**

```
HTTP/1.1 200 OK
Date: Sat, 07 Jun 2003 13:32:07 GMT
Server: Apache/1.3.27
Last-Modified: Mon, 14 Apr 2003 14:11:33 GMT
ETag: "bdae-4c1-3e9ac195"
Accept-Ranges: bytes
Content-Length: 1217
Keep-Alive: timeout=5, max=100
```

Ready Buffer: 3% URLs: 95 Packets: 393

Login y pass  
van en cabeceras  
HTTP. Contenidos en  
HTML (texto)

Cabeceras HTTP  
en texto plano



# Infraestructura de Seguridad

- En inglés PKI, Public Key Infrastructure
- Objetivos de las PKI
  - Control de acceso
  - Autenticación
  - Confidencialidad
  - Integridad
  - No repudio

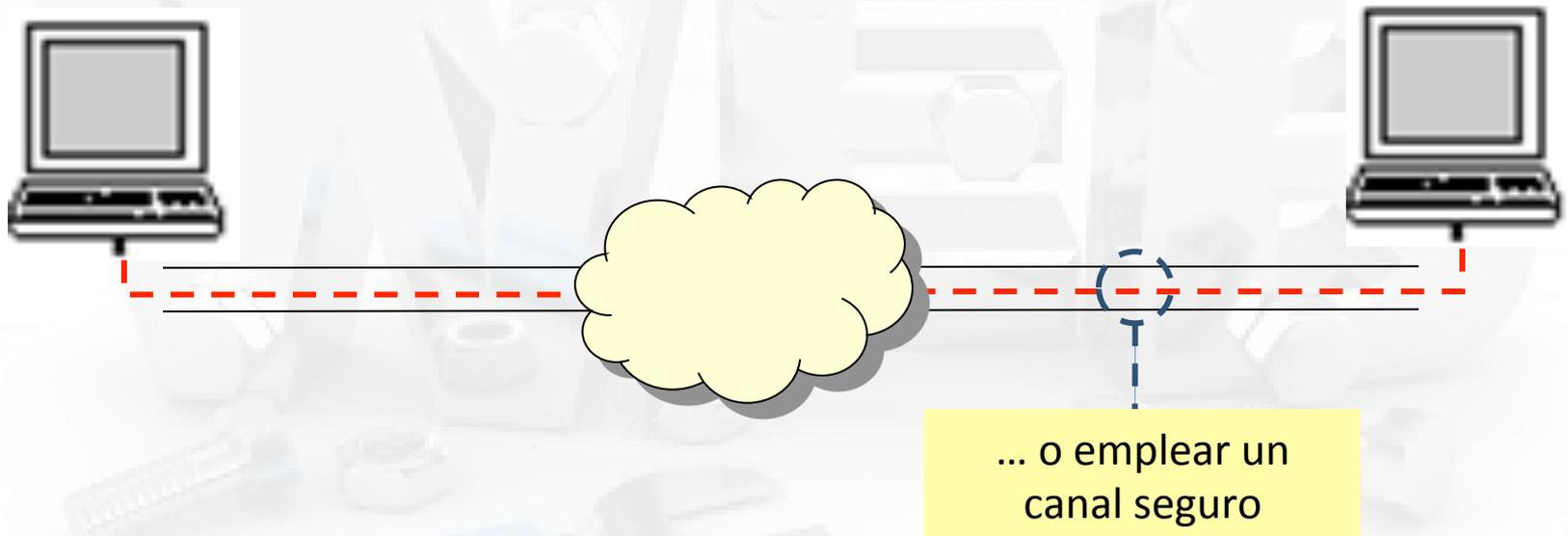
# Seguridad, alternativas

- Codificar el mensaje



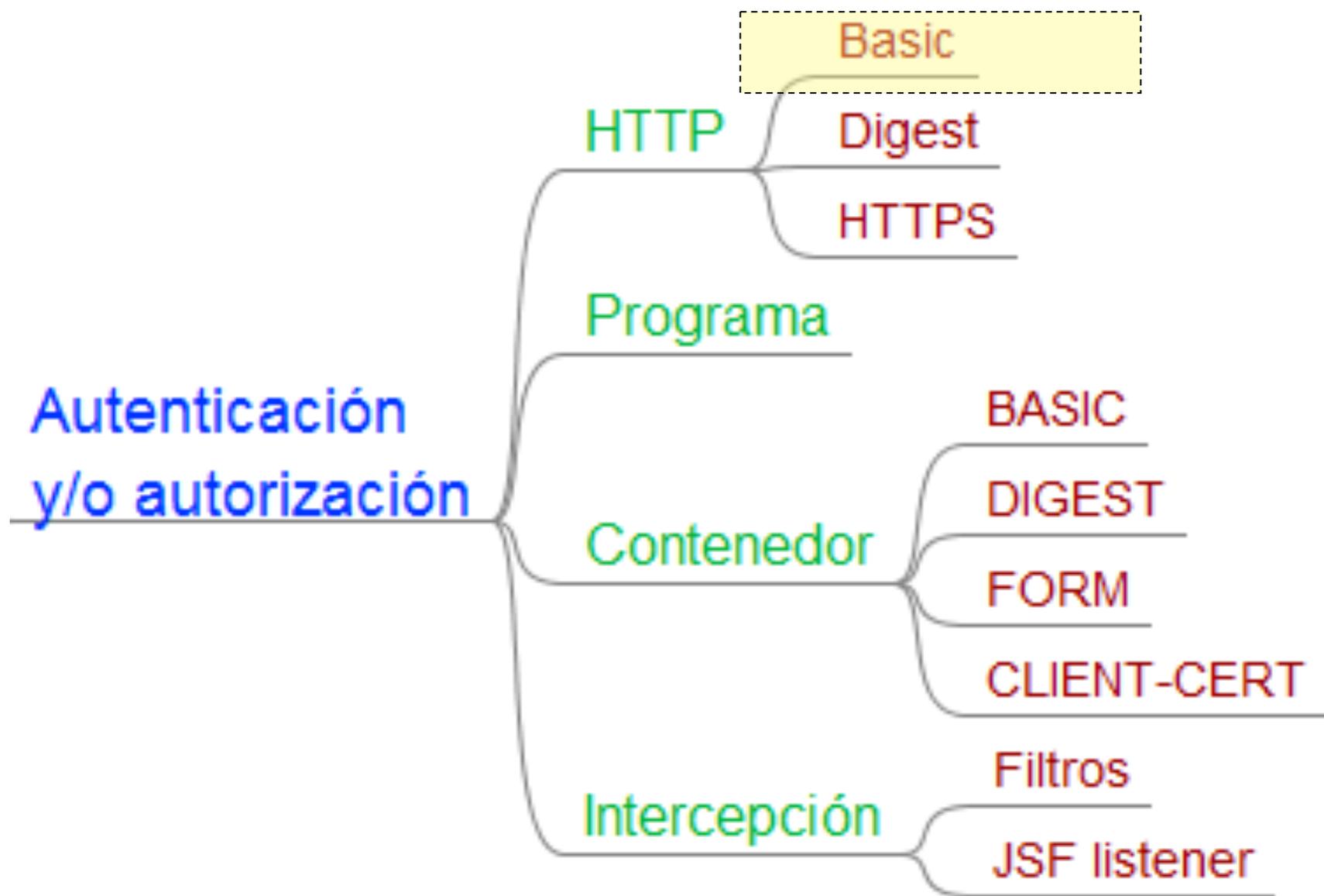
# Seguridad, alternativas

- Emplear un canal seguro



# Autorización

- Basada en listas de control de acceso (ACL)
  - Para cada recurso se especifica qué usuarios pueden hacer qué operaciones
- Basada en Roles
  - Rol: grupo de permisos.
  - Permiso: operación sobre un recurso.
  - A los usuarios se les asignan roles.



Basic

HTTP

Digest

HTTPS

Programa

Autenticación y/o autorización

Contenedor

BASIC

DIGEST

FORM

CLIENT-CERT

Intercepción

Filtros

JSF listener

# Autenticación HTTP Básica

- Cuando el navegador solicita un recurso (jsp, por ejemplo), el servidor solicita usuario y contraseña, que viajan codificadas en Base-64.
- Disponible desde versión 1.0 de HTTP
- Muy simple
- Poco seguro

# Autenticación HTTP Básica

Navegador

Servidor

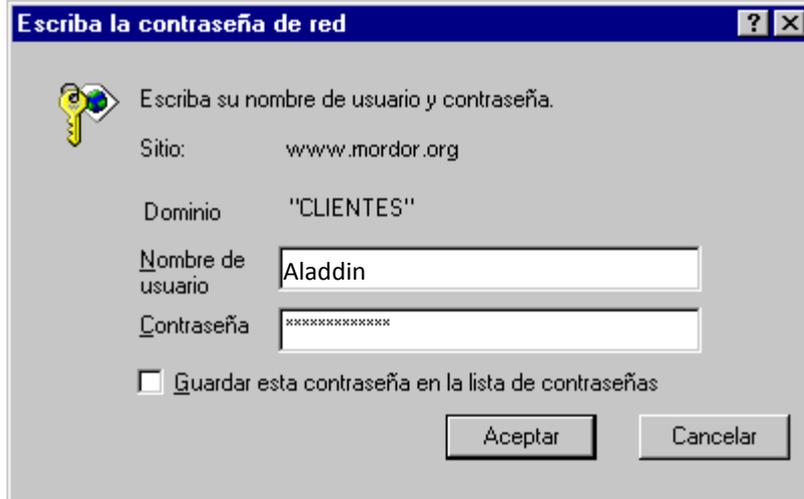
```
GET /privado/index.php HTTP/1.1
Host: example.com
```

```
HTTP/1.1 401 Unauthorised
Server: example.com/1.1
Date: Thu, 17 Oct 2006 19:18:15 GMT
WWW-Authenticate: Basic realm="example.com"
Content-Type: text/html
Content-Length: 311
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
"http://www.w3.org/TR/1999/REC-html401-1999
<HTML>
  <HEAD>
    <TITLE>Error</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT=
  </HEAD>
  <BODY><H1>401 Unauthorised.</H1></BODY>
</HTML>
```

# Autenticación HTTP Básica

- Login: **Aladdin**, pass: **open sesame**
- “Aladdin:open sesame”
  - QWxhZGRpbjpvvcGVuIHNIc2FtZQ==



Escriba la contraseña de red

Escriba su nombre de usuario y contraseña.

Sitio: www.mordor.org

Dominio: "CLIENTES"

Nombre de usuario: Aladdin

Contraseña: \*\*\*\*\*

Guardar esta contraseña en la lista de contraseñas

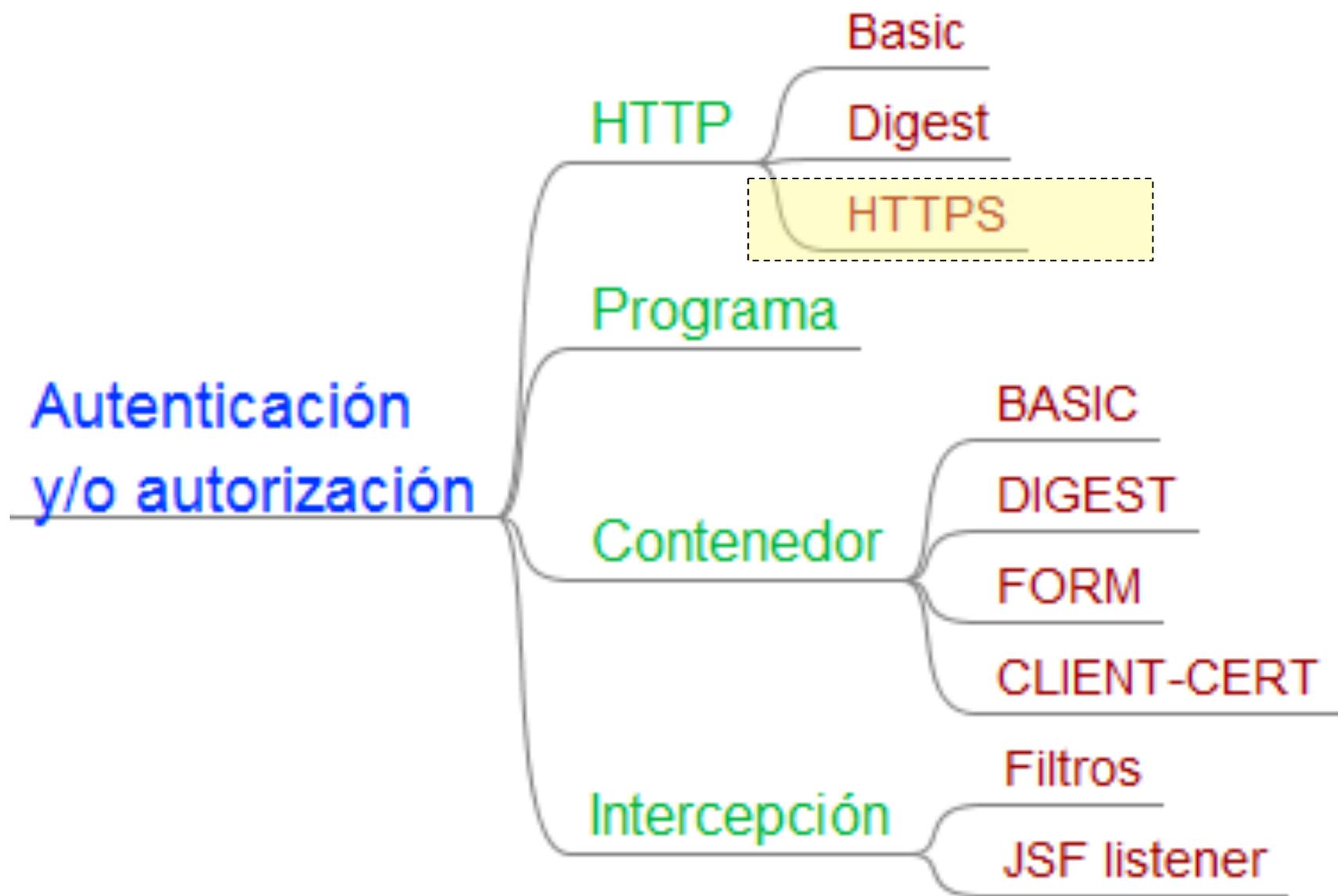
Aceptar Cancelar

Base64

```
GET /privado/index.php HTTP/1.1
Host: example.com
Authorization: Basic QWxhZGRpbjpvvcGVuIHNIc2FtZQ==
```

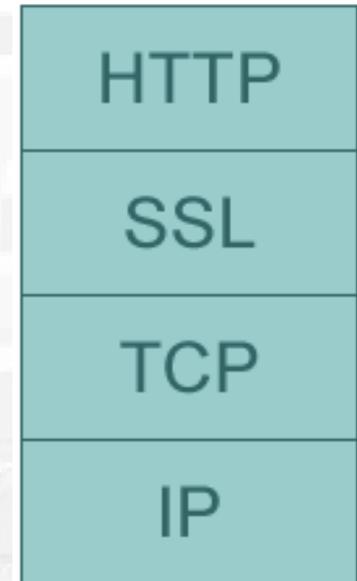
# Autenticación HTTP digest (hash)

- Desde la versión 1.1 de HTTP
- La clave viaja encriptada en base a un timestamp generado por el servidor en el momento de la petición
- Más difícil de decodificar, Más seguro.
- No está totalmente difundida ni aceptada por todos los navegadores y servidores.



# Seguridad para WEB (HTTPS)

- HTTPS = HTTP Secure
- HTTP sobre un protocolo de transporte seguro
  - HTTP → SSL → TCP
- SSL también es conocido como TLS según IETF

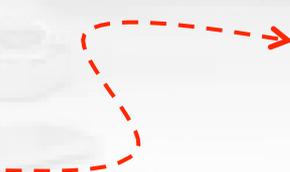


# Transmisión con HTTPs

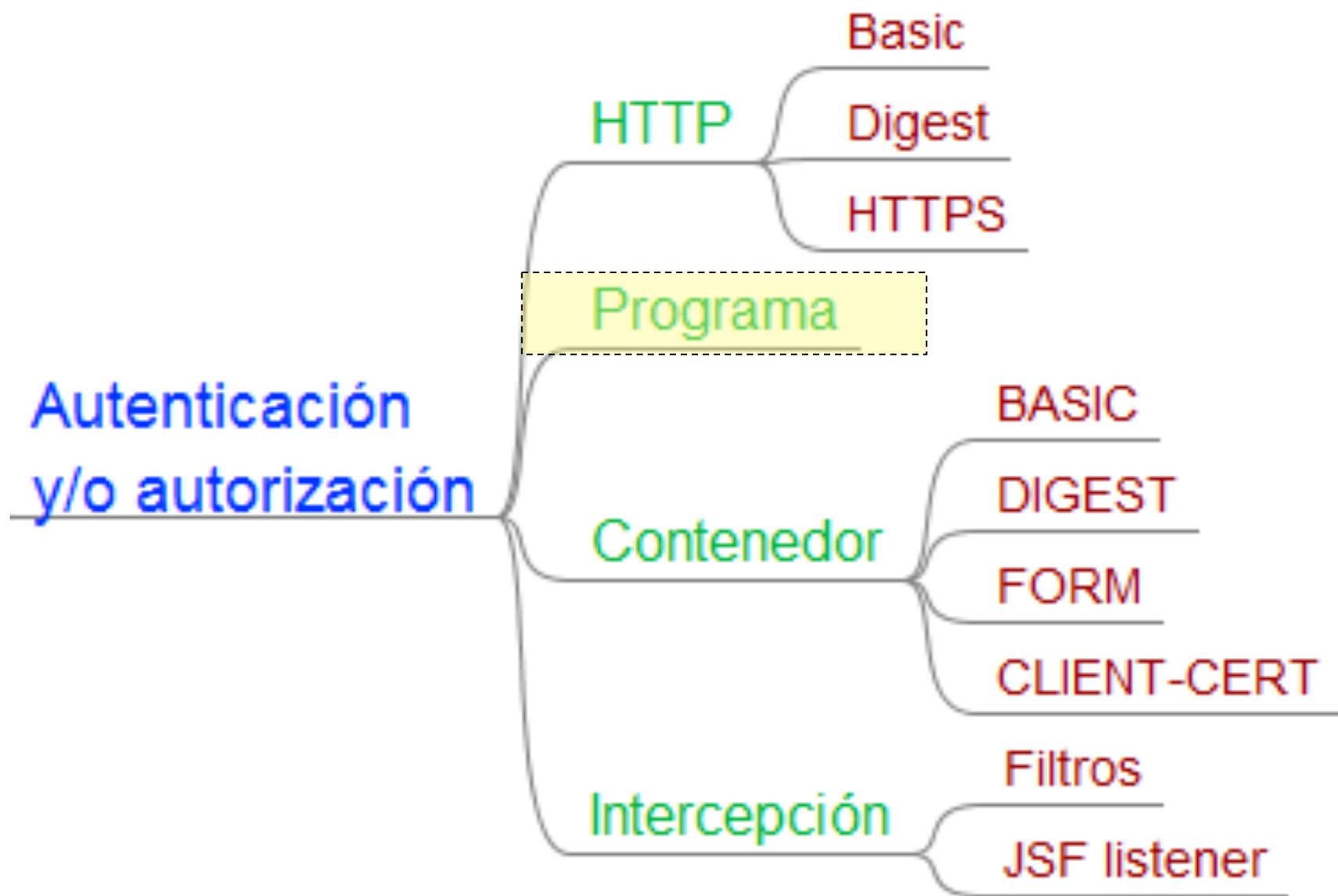
- Encripta todo el contenido de los paquetes, no sólo la clave.
- Se basa en mecanismos de Clave Pública (PKI): SSL (aka TLS)
- Requiere certificado(s) generado(s) por una autoridad certificadora (CEA)
- Hay un proceso “complicado” de *handshake* que garantiza la conexión segura.

# SSL (o TLS)

- El servidor tiene un certificado
  - opcionalmente el cliente tiene el suyo
- Permite intercambio seguro de datos (p.e. claves de sesión)
- Usa algoritmos de clave simétrica y asimétrica
- Ofrece:
  - Autenticación
  - Confidencialidad
  - Integridad
  - No repudio



Del servidor siempre, del cliente si tiene certificado



# Autenticación por Programa

- Todo lo tenemos que programar
- La aplicación coteja la contraseña contra su propio sistema de control de usuarios
- Hay que basarla en protocolos seguros de comunicación como HTTPs (SSL) para evitar que nadie “escuche” la contraseña interpretando los paquetes HTTP.
- Más tedioso de programar
- Ventajas: No dependemos de la Autenticación del entorno → Más portable.

# Notaneitor!

## Aplicación de gestión de alumnos

Debe identificarse para entrar en la aplicación

Login:

Password:

```
@ManagedBean(name="login")
public class LoginBean implements Serializable {
    private static final long serialVersionUID = 875296

    private String name = "";
    private String password = "";

    private String result = "login_form_result_valid";

    public String verify() {

        if ( validLogin(name, password) ) {
            return "success";
        }

        result = "login_form_result_error";
        return "login";
    }
}
```

# Notaneitor!

## Aplicación de gestión de alumnos

Opciones

[Listado de alumnos](#)

[Alta de alumno](#)

success

```
<ui:define name="titulo">
  #{msgs.login_form_tittle}
</ui:define>
```

```
<ui:define name="cuerpo">
<center>
```

```
  <h:form>
```

```
    <ui:include src="/snippets/login-form.xhtml"/>
```

```
    <h:panelGrid columns="2">
```

```
      <h:outputText id="msgAlta" value="#{msgs[login.result]}" />
```

```
    </h:panelGrid>
```

```
    <h:commandButton
```

```
      value="#{msgs.login_form_button_send}"
```

```
      action="#{login.verify}">
```

```
    </h:commandButton>
```

```
    <br />
```

```
  </h:form>
```

```
</center>
```

```
</ui:define>
```

```
<ui:define name="pie" />
```

```
<h:panelGrid columns="2">
```

```
  #{msgs.login_form_name_label}:<br>
```

```
  <h:panelGroup>
```

```
    <h:inputText
```

```
      id="name"
```

```
      value="#{login.name}"
```

```
      required="true"
```

```
      requiredMessage="#{msgs.login_form_name_required}" />
```

```
    <h:message for="name" />
```

```
  </h:panelGroup>
```

```
  #{msgs.login_form_password_label}:<br>
```

```
  <h:panelGroup>
```

```
    <h:inputSecret
```

```
      id="password"
```

```
      value="#{login.password}"
```

```
      required="true"
```

```
      requiredMessage="#{msgs.login_form_password_required}" />
```

```
    <h:message for="password" />
```

```
  </h:panelGroup>
```

```
</h:panelGrid>
```

# Login Form

# Ejemplo por programa

```
<navigation-rule>
  <from-view-id>/index.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{login.verify}</from-action>
    <from-outcome>success</from-outcome>
    <to-view-id>/main.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-action>#{login.verify}</from-action>
    <from-outcome>login</from-outcome>
    <to-view-id>/index.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

# Autenticación y/o autorización

HTTP

Basic

Digest

HTTPS

Programa

Contenedor

BASIC

DIGEST

FORM

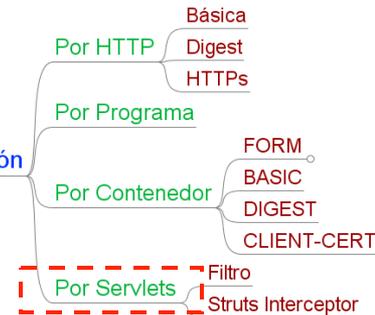
CLIENT-CERT

Intercepción

Filtros

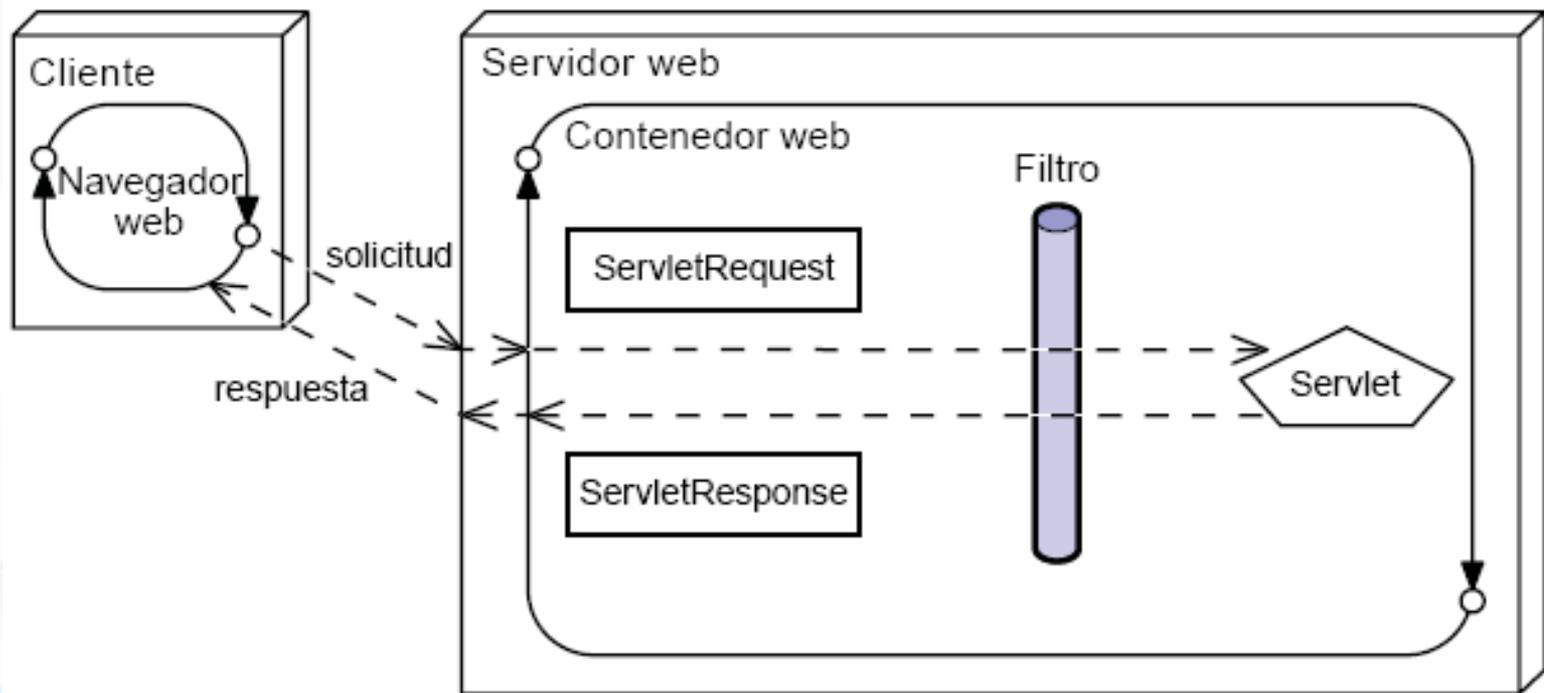
JSF listener

# Filtros Servlet HTTP



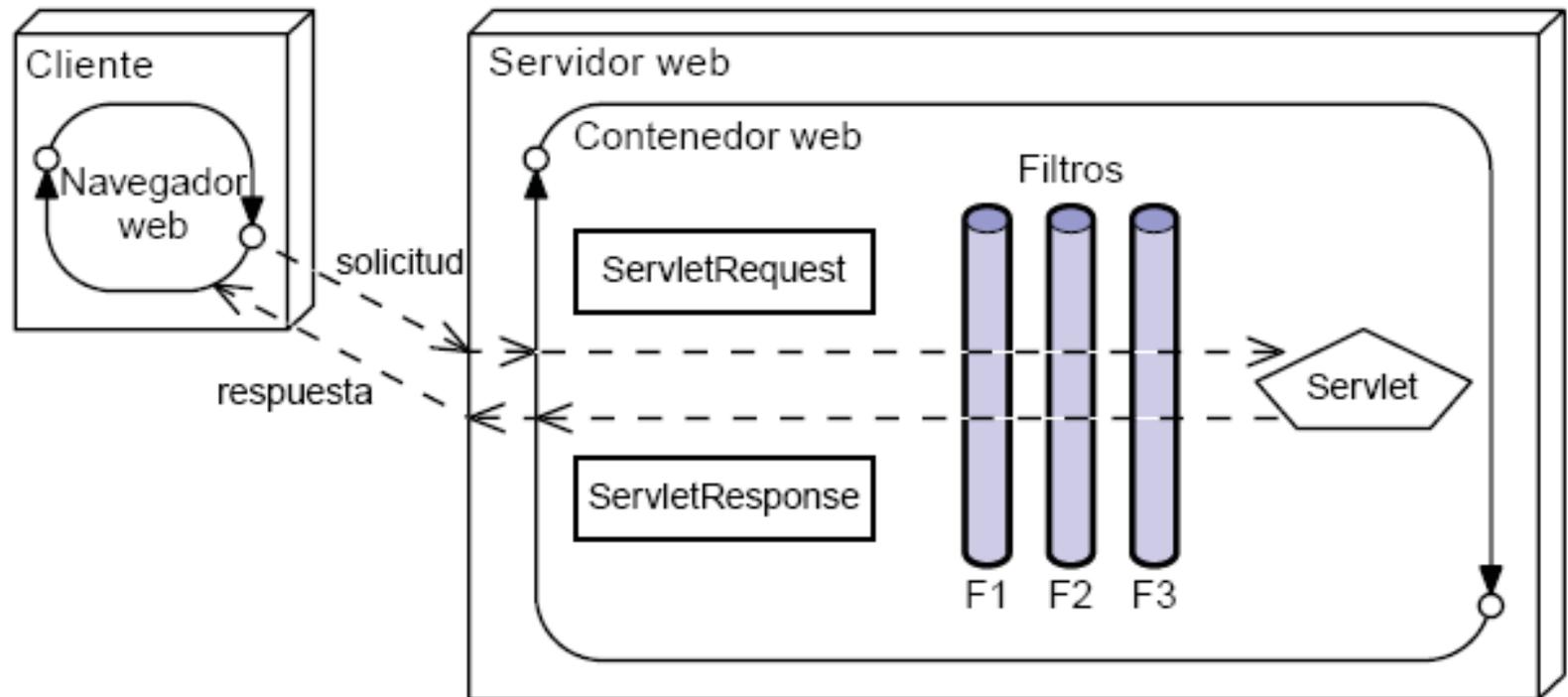
- Incorporados desde servlets 2.3.
- Interceptan la invocación del servlet
  - **Antes y después** de que sea invocado el propio servlet.
- Permiten examinar y modificar la request antes de que le llegue al servlet.
- Permite modificar el response y redirigir, en caso necesario, la petición a otro recurso distinto.
- Ideales para el control de acceso de usuarios autenticados

# Aplicación de filtros a solicitudes entrantes



Intercepción de solicitud y respuesta mediante filtro

# Cadenas de filtros



Varios filtros aplicados en cadena

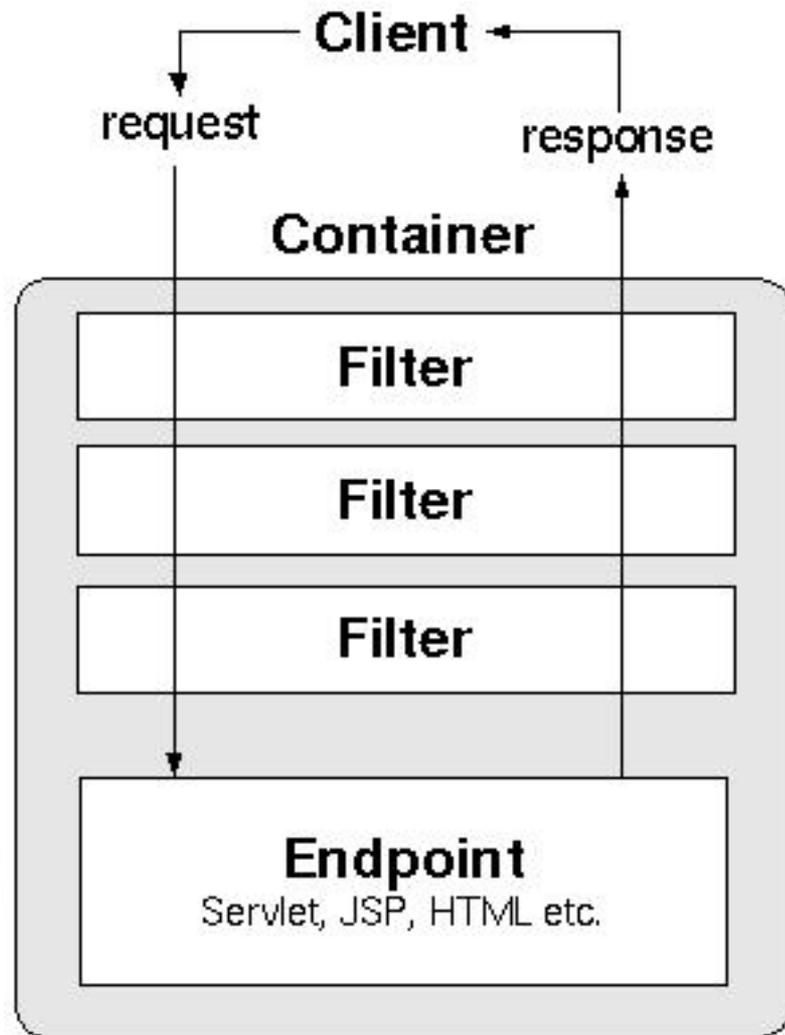
# Filtros, ventaja

- Permiten hacer autenticación por programa de forma transversal a la aplicación → el código de la aplicación no se modifica
- Lo habitual es redireccionar a formulario → necesario HTTPs
  - Sigue siendo autenticación por formulario
- A parte de autenticación permiten añadir de forma genérica “aspectos” a la aplicación

# Filtros Servlet HTTP

- Interfaz `javax.servlet.Filter`
- Tres métodos:
  - `void init(FilterConfig config) throws ServletException` Invocado antes de que el filtro entre en servicio. Permite configurar el filtro.
  - `void destroy()` Invocado cuando el filtro dejar de estar en servicio.
  - `void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException`: Método que implementará el filtrado.

# Interfaz filter y ciclo de vida

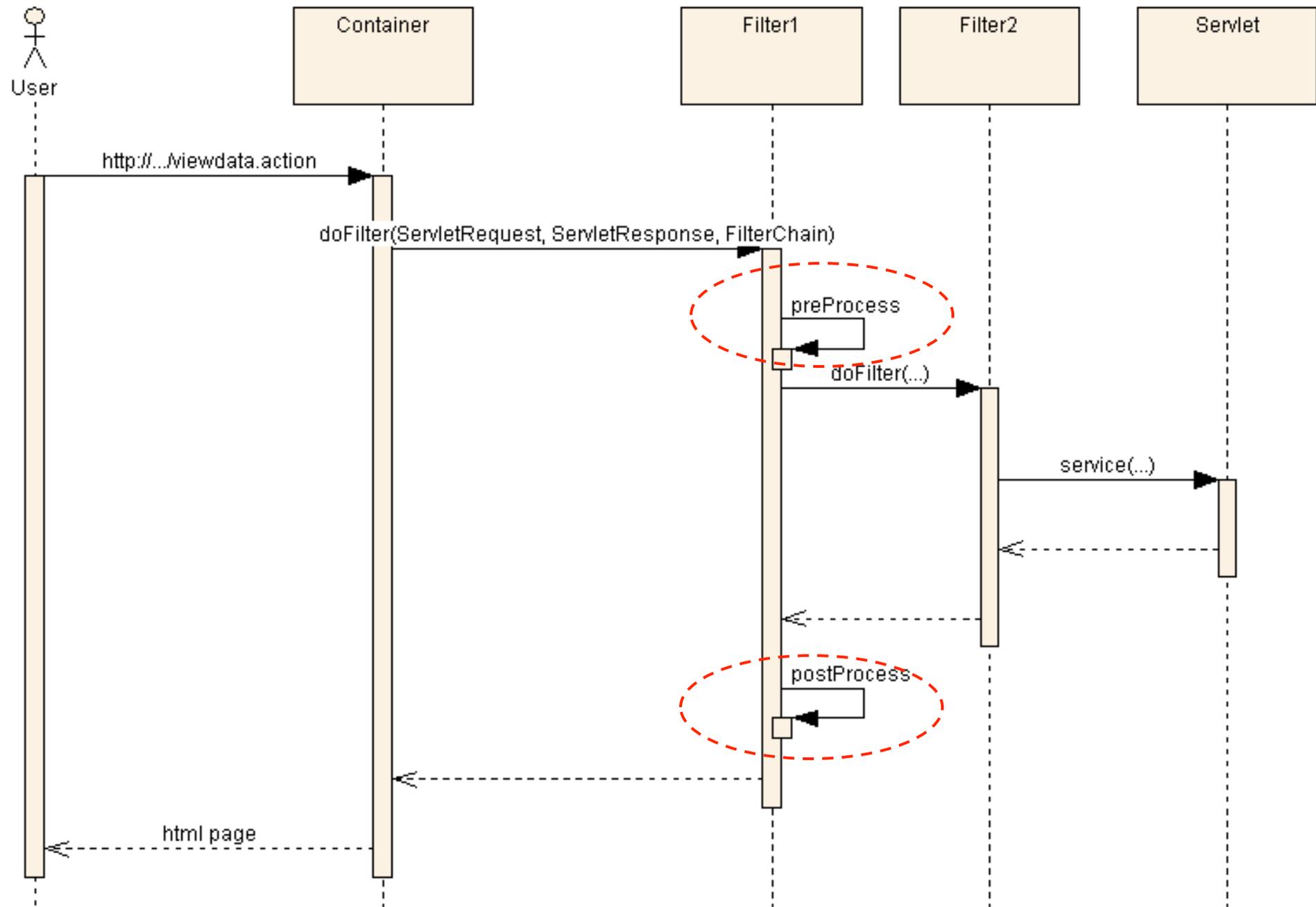


```
Filter  
..... ● destroy()  
..... ● doFilter(ServletRequest, ServletResponse, FilterChain)  
..... ● init(FilterConfig)
```

```
FilterConfig  
..... ● getFilterName()  
..... ● getInitParameter(String)  
..... ● getInitParameterNames()  
..... ● getServletContext()
```

# Procesamiento típico en un filtro

- Examinar las cabeceras de request
- [Si procede]Preprocesar la petición
- **Invocar al resto de la cadena de filtros**
- [Si procede]Postprocesar la petición
- A la ida puede:
  - Responder él a la request y terminar proceso
  - Abortar la ejecución → `UnavailableException`
  - Redirigir a otra URL
  - Añadir info a session, request o contexto
- A la vuelta puede:
  - Examinar la respuesta, modificarla
  - Examinar las cabeceras añadidas



# Ejemplo, redirección, Notación

```
@WebFilter ( dispatcherTypes = {DispatcherType.REQUEST },
description = "Filtro de seguridad",
urlPatterns = { "/"*"} },
initParams = {
    @WebInitParam(name = "LoginForm",
value = "/index.xhtml",
description = "Página de logeo")
    }
)
public class LoginFilter implements Filter {

    ....

}
```

# Ejemplo, redirección a login

```
public class LoginFilter implements Filter {
```

```
    FilterConfig config = null;
```

```
    public void doFilter(ServletRequest request, ServletResponse response,  
                        FilterChain chain) throws IOException, ServletException {
```

```
        // Si no es petición HTTP nada que hacer
```

```
        if (!(request instanceof HttpServletRequest)) {  
            chain.doFilter(request, response); return;  
        }
```

```
        HttpServletRequest req = (HttpServletRequest) request;
```

```
        // Si es la petición de login nada que hacer (si no entramos en bucle)
```

```
        String uri = req.getRequestURI().toLowerCase();  
        String contextpath = req.getContextPath().toLowerCase()+"/";  
        if ( uri.contains("index") || uri.equals(contextpath)) {  
            chain.doFilter(request, response);  
            return;  
        }
```

```
        // En el resto de casos se verifica que se haya hecho login previamente
```

```
        HttpServletResponse res = (HttpServletResponse) response;
```

```
        HttpSession session = req.getSession();
```

```
        if (session.getAttribute("LOGGEDIN_USER") == null) {
```

# Ejemplo, redirección a login

**// En el resto de casos se verifica que se haya hecho login previamente**

```
HttpServletResponse res = (HttpServletResponse) response;
```

```
HttpSession session = req.getSession();
```

**//Sino está logeado se le envia a la página de logeo (loginForm)**

```
if (session.getAttribute("LOGGEDIN_USER") == null) {
```

```
String loginForm = config.getInitParameter("LoginForm");
```

```
// Si no hay login, redirección al formulario de login
```

```
res.sendRedirect(req.getContextPath() + loginForm);
```

```
return;
```

```
}
```

**//Si ya está logeado se continua la cadena de logeo**

```
chain.doFilter(request, response);
```

```
}
```

```
public void init(FilterConfig fConfig) throws ServletException {
```

```
config = fConfig;
```

```
}
```

```
}
```

# Resumen Autorización



|                       | <b>Autenticar</b>                      | <b>Autorizar</b>        | <b>Transporte</b> |
|-----------------------|--|-------------------------|-------------------|
| <b>Protocolo HTTP</b> | Basic<br>Digest                        | ----                    | HTTPS             |
| <b>Por programa</b>   | Programado                             | Filtros<br>JSF listener |                   |
| <b>Por contenedor</b> | BASIC<br>DIGEST<br>FORM<br>CLIENT-CERT |                         |                   |



# LOPD

- Niveles de sensibilidad de los datos: bajo, medio y alto
- Formularios de autorización de incorporación de datos personales al archivo digital de la base de datos de la web comercial.
- Ejemplo:

Código Postal  \*

Ciudad  \*

País  \*

Teléfono

Identificación fiscal

Número de identificación  \* DNI / NIF / NIE

Acepta la política de privacidad \*

\* Campo requerido

Regístrate

## Identificate

Si ya eres cliente de Amazon.com, Amazon.co.uk, Amazon.de, Amazon.fr, Amazon.ca o Amazon.it, puedes identificarte con esa cuenta.

### ¿Cuál es tu dirección de e-mail?

Mi dirección de e-mail es:

### ¿Tienes una contraseña de Amazon.es?

No, soy un cliente nuevo.

Sí, ya tengo una contraseña:

[¿Has olvidado tu contraseña?](#)

Identificarse (servidor seguro) 

Al identificarte, aceptas nuestras [Condiciones de uso](#), nuestro [Aviso de privacidad](#) y nuestras [Condiciones de Cookies y publicidad en Internet](#).