Librería genérica de aspectos para tratar las vulnerabilidades de seguridad más críticas en aplicaciones web

Diego Hernán Bellante
Hernán David Chanfreau

Facultad de Informática, UNLP, La Plata,
Provincia de Buenos Aires, Argentina.

Directores: Lic. Javier Díaz
Lic. Claudia Queiruga

POA: Programación Orientada a Aspectos

- **Concern** → cualquier materia de interés en un sistema de software.
  Es decir que un **concern** es alguna funcionalidad o algún requerimiento necesario en el sistema.

- Crosscutting concerns → Requerimientos secundarios o no-funcionales.
  Ej: La seguridad, el caching, la encriptación, el logging.

- POA → paradigma de programación propuesto por Gregor Kiczales.
  - Captura/modulariza **crosscutting concerns** de un sistema → Aspecto.
  - Complementa las metodologías existentes tales como la Programación Orientada a Objetos y la Programación Procedural.

- Beneficios de POA:
  - Mayor grado de modularización
  - Código menos confuso
  - Fácil evolución del sistema
  - Mayor reuso de código

POA: Programación Orientada a Aspectos

- Extensiones de lenguajes de programación para soporte de POA
  - Java → AspecJ
  - C++ → AspectC++
  - Smalltalk → Apostle
  - C → AspectC
- Etapas del desarrollo con POA (asimétrica)
  - Descomposición en aspectos.
  - Implementación de requerimientos principales
  - Implementación de aspectos.
  - Integración o *weaving*.

Seguridad y POA

- La **Seguridad** es un ejemplo clásico de un **concern** no-funcional. Diversidad de lugares y momentos de invocación.
- La seguridad es especificada o tratada en forma separada para luego ser integrada junto con el resto de la aplicación en una única unidad ejecutable.
- La aplicación a segurizar puede ser desarrollada y mantenida independientemente.
- Expertos pueden enfocarse en las soluciones de seguridad propiamente dicha sin tener en cuenta la lógica de negocios.

Identificación de las vulnerabilidades más importantes en las aplicaciones web

- **OWASP** → comunidad abierta dedicada a brindar asesoramiento de seguridad a organizaciones. http://www.owasp.org.
- **OWASP Top Ten** → amplio consenso sobre cuáles son las *fallas de seguridad más críticas en aplicaciones web.*
- Vulnerabilidades del **Top Ten** que pueden presentarse en aplicaciones desarrolladas en Java:

  A1) Fallas de Cross Site Scripting (XSS).

  A2) Fallas de Inyección.

  A6) Manejo Inadecuado de Errores.

  A7) Administración de Autenticación y Sesión Interrumpida.

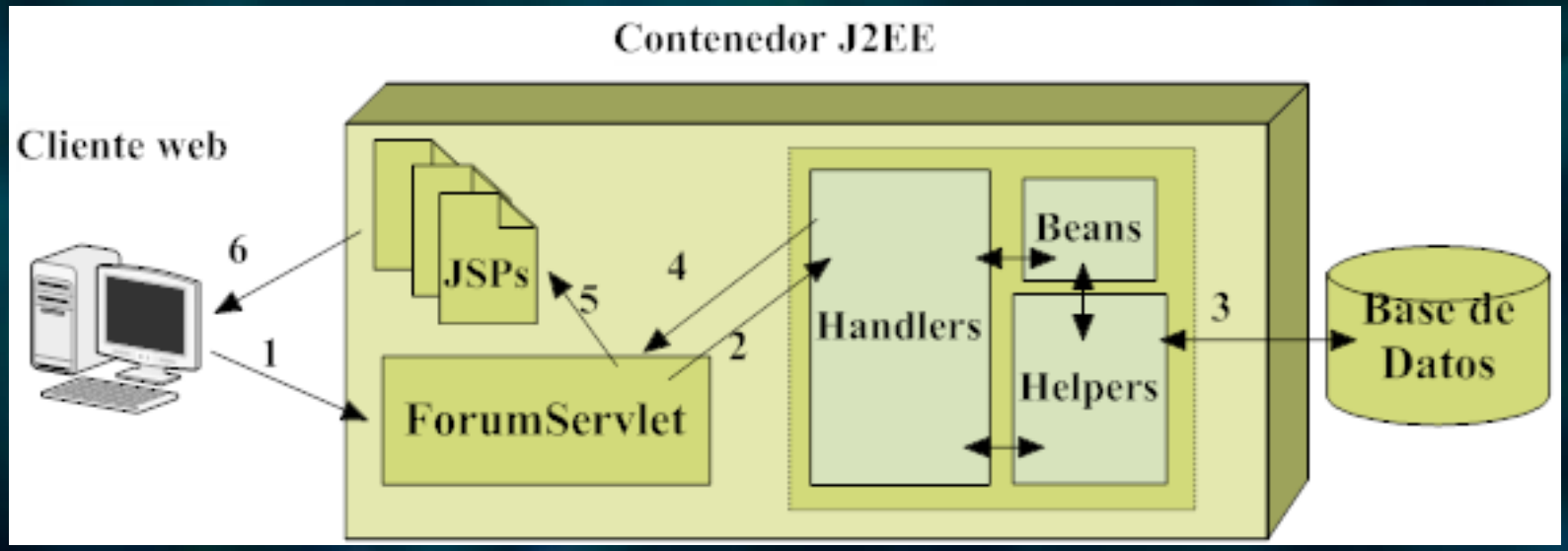
Reorganización de las vulnerabilidades

- Entrada no validada sintácticamente
  - XSS
  - Ataques de inyección (Inyección SQL)
- Autenticación y control de acceso deficientes
- Manejo inadecuado de errores

Un caso de prueba: Foro Vulnerable

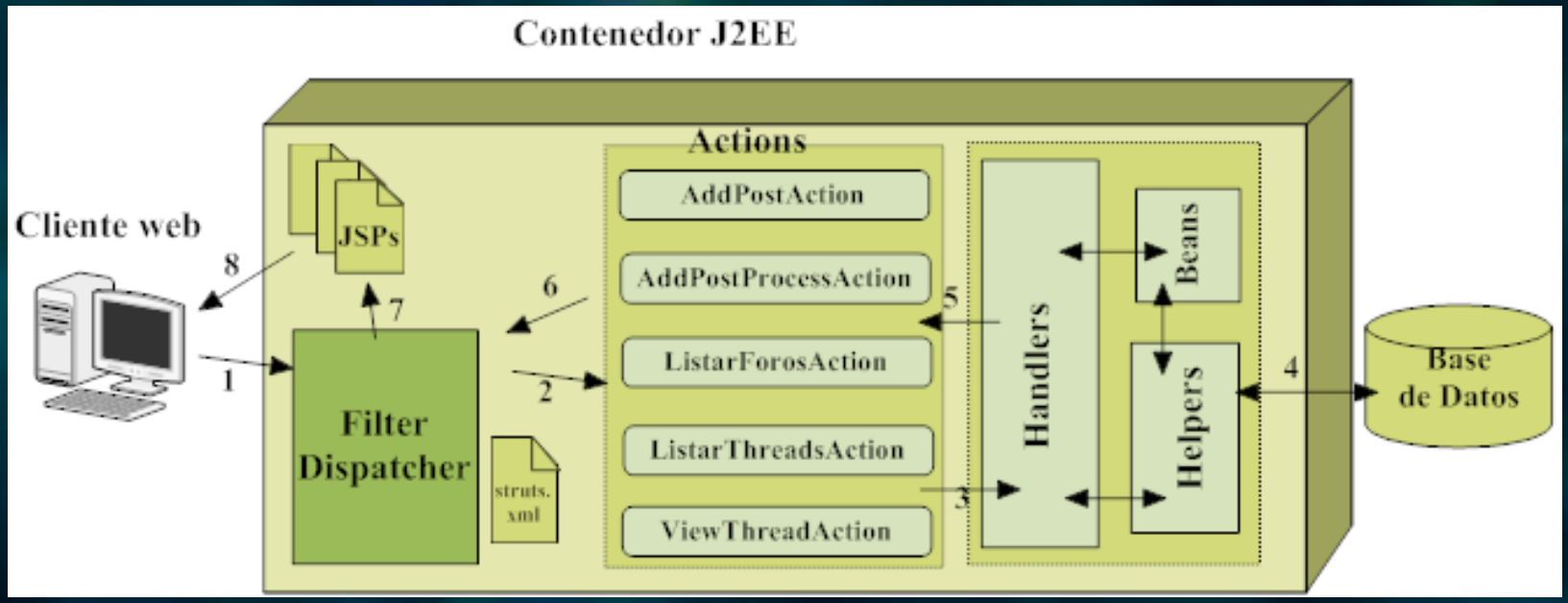
- Dominio elegido: **Foro Web**. Requiere una amplia interacción con el usuario, y que nos facilita el trabajo con todas las vulnerabilidades propuestas.

- **mvnForum:** foro de código fuente abierto y fácil de configurar, construido sobre la tecnología J2EE. http://www.mvnforum.com



Foro Vulnerable: Rediseño de mvnForum

- Modificación del código de **mvnForum** para convertirlo en vulnerable a diferentes ataques de seguridad.
- Incorporación del framework MVC Web ***Struts 2***.



Aspectos de Seguridad

- Comprobación de existencia de vulnerabilidades en Foro Vulnerable generando ataques de diferentes tipos.

- Implementación de Aspectos de Seguridad que tratan estas vulnerabilidades.
  - Basados en las propuestas de OWASP.
  - Implementados utilizando Java 6 en eclipse 3.3 con el plugin AspectJ Developement Tools 1.5.

- Incorporación a Foro Vulnerable mediante weaving en tiempo de compilación y verificación de funcionamiento.

AspectJ

- **Extensión del lenguaje Java para soporte de programación orientada a aspectos. http://www.eclipse.org/aspectj**
- **Los aspectos se escriben en Java extendido y se compilan a código de bytes.**
- **Weaving estático**
- **Por que elegimos AspectJ?**
  - Foco en aplicaciones web Java.
  - Popularidad.
  - Integración con IDEs.
  - Pertenece a la comunidad de software libre.
  - Extensión natural de Java
  - Respeta la especificación de bytecodes.
- **Conceptos importantes:**
  - Join point → punto identificable en la ejecución de un programa.
  - Pointcut → Captura Join points.
  - Advice → Código a ser ejecutado en un Join point.
  - Introduction → Introduce cambios estáticos.
  - Aspect → Análogo a “class” pero para la definición de un aspecto.

Ataque XSS – Cross Site Scripting

- Víctima: usuarios finales de una aplicación.
- Consiste el la inserción de código malicioso (scripts) en páginas web que serán vistas por otros usuarios.
- Lenguajes: HTML, JavaScript, VBScript, ActiveX, Shockwave, Flash, etc.
- Clasificación:
  - **Reflejado**: el código malicioso es utilizado inmediatamente para generar una página de respuesta al usuario.
  - **Almacenado**: el código malicioso es almacenado en un servidor antes de ser enviado al usuario.

Ataque XSS – Cross Site Scripting

- Tratamiento:
  - Validación mediante “lista blanca” de los datos de entrada.
    - Longitud
    - Tipo
    - Sintaxis
    - Reglas de negocio

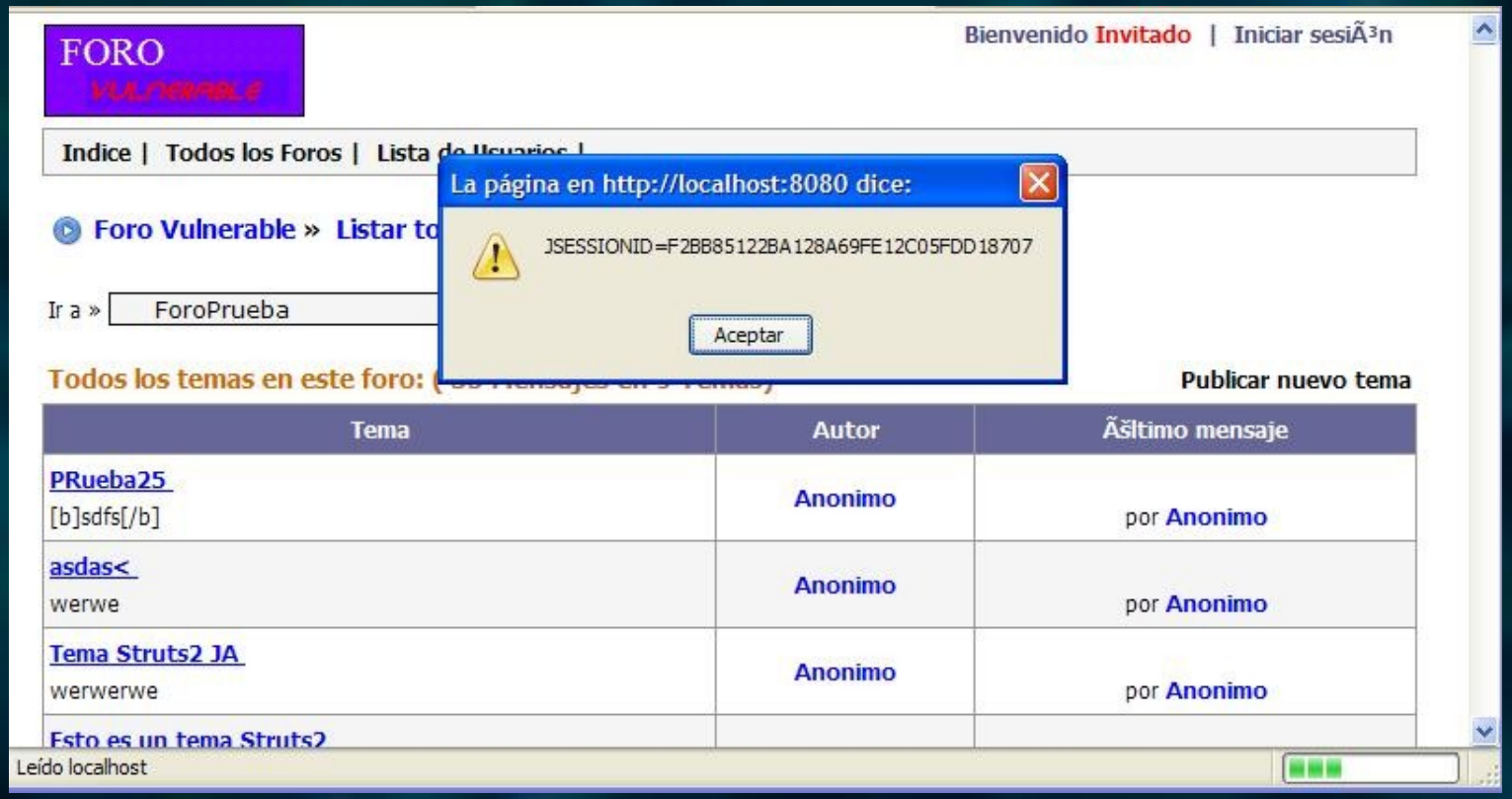
    No se intenta “sanar” datos potencialmente peligrosos.
  - Codificación apropiada de los datos de salida.
    - Conversión de caracteres a la entidad de codificación HTML apropiada

De:	A:
<	&lt;
>	&gt;
(	&#40;
)	&#41;
#	&#35;
&	&#38;

Ataque XSS a Foro Vulnerable

- Código JavaScript inyectado:

  http://coe.info.unlp.edu.ar:8081/foroVulnerable/listarThreads.action?forumID=1&sort="<SCRIPT>alert(document.cookie)</SCRIPT>



Ataque de Inyección: sentencias SQL

- Víctima: aplicación web.
- Se transmite código malicioso a través de una página web hacia otros sistemas (DBMS, SO).
- La entrada puede ser procesada inmediatamente o ser almacenada.
- **Inyección SQL**: se insertan sentencias SQL que son enviados al motor de base de datos para su ejecución.
- Clasificación:
  - In band: la información es retornada directamente por la aplicación web atacada.
  - Out of band: la información es retornada por otro canal (Ej.: e-mail).
  - Inferido: no existe información retornada. Debe ser inferida.

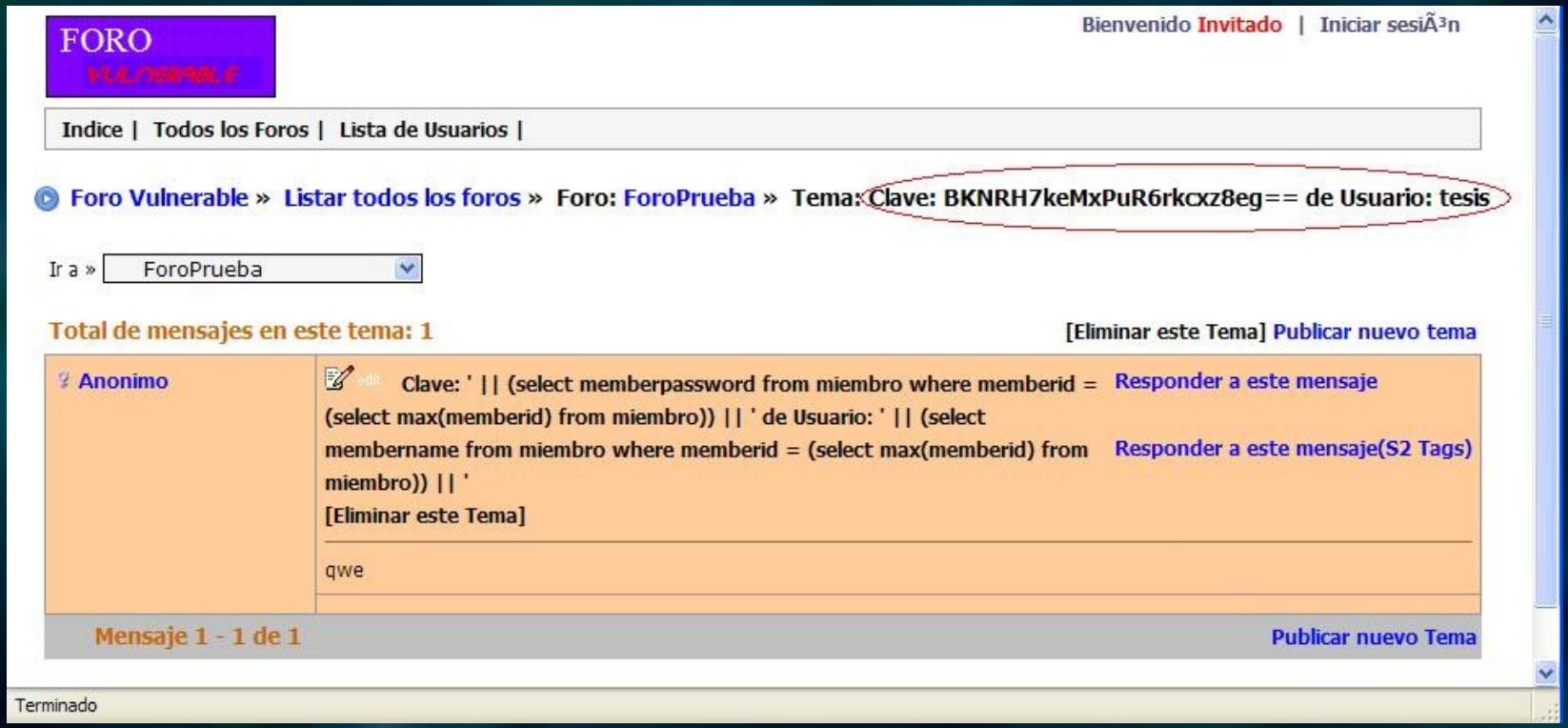
Ataque de Inyección: sentencias SQL

- Tratamiento:
  - Validación de los datos de entrada.
  - Todos los datos que se mandan al motor de base de datos no deben contener meta caracteres.
  - Métodos complementarios:
    - Utilizar parámetros tipados
    - No ejecutar el servicio de base de datos con usuarios con privilegios innecesarios.
    - Eliminar procedimientos almacenados obsoletos o restringir su acceso.
    - Mantener un logging de potenciales intentos de inyección SQL y bloquear la dirección IP en cuestión.
    - Controlar las consultas SQL generadas dinámicamente.

Ataque de inyección SQL a Foro Vulnerable

- Script SQL inyectado:

  Clave: ' || (select memberpassword from miembro where memberid = (select max(memberid) from miembro)) || ' de Usuario: ' || (select membername from miembro where memberid = (select max(memberid) from miembro)) || '



Aspecto de Seguridad para validación de datos de entrada

- **Aspecto de Validación***:*
  - Protección contra XSS e Inyección SQL
  - Validación de datos de entrada.
  - Motor de validación: Stinger 2.5.
  - Intercepta las llamadas a funciones expuestas.

Aspecto de Seguridad para validación de datos de entrada

```
public aspect AspectoValidacion{

   public pointcut ejecucionMetodoExecute():
        execution (public String  ActionSupport+.execute());

   before() throws Exception: ejecucionMetodoExecute() {

        . . .

        /* Validación con Stinger */

        . . .

   }
}
```

Autenticación y control de acceso deficientes

- Autenticación: proceso de verificación de identidad
  - Mínimamente método de usuario/password.
  - Protección de los tokens de sesión.
  - Encriptación del password enviado al servidor.
  - Almacenar en forma de hash las contraseñas.
  - No utilización de HTTP GET

- Control de acceso o Autorización: determina que acciones puede realizar un usuario en el sistema.
  - Roles o grupos de usuarios.
  - Centralización de reglas de control de acceso para facilitar el manejo y comprensión.

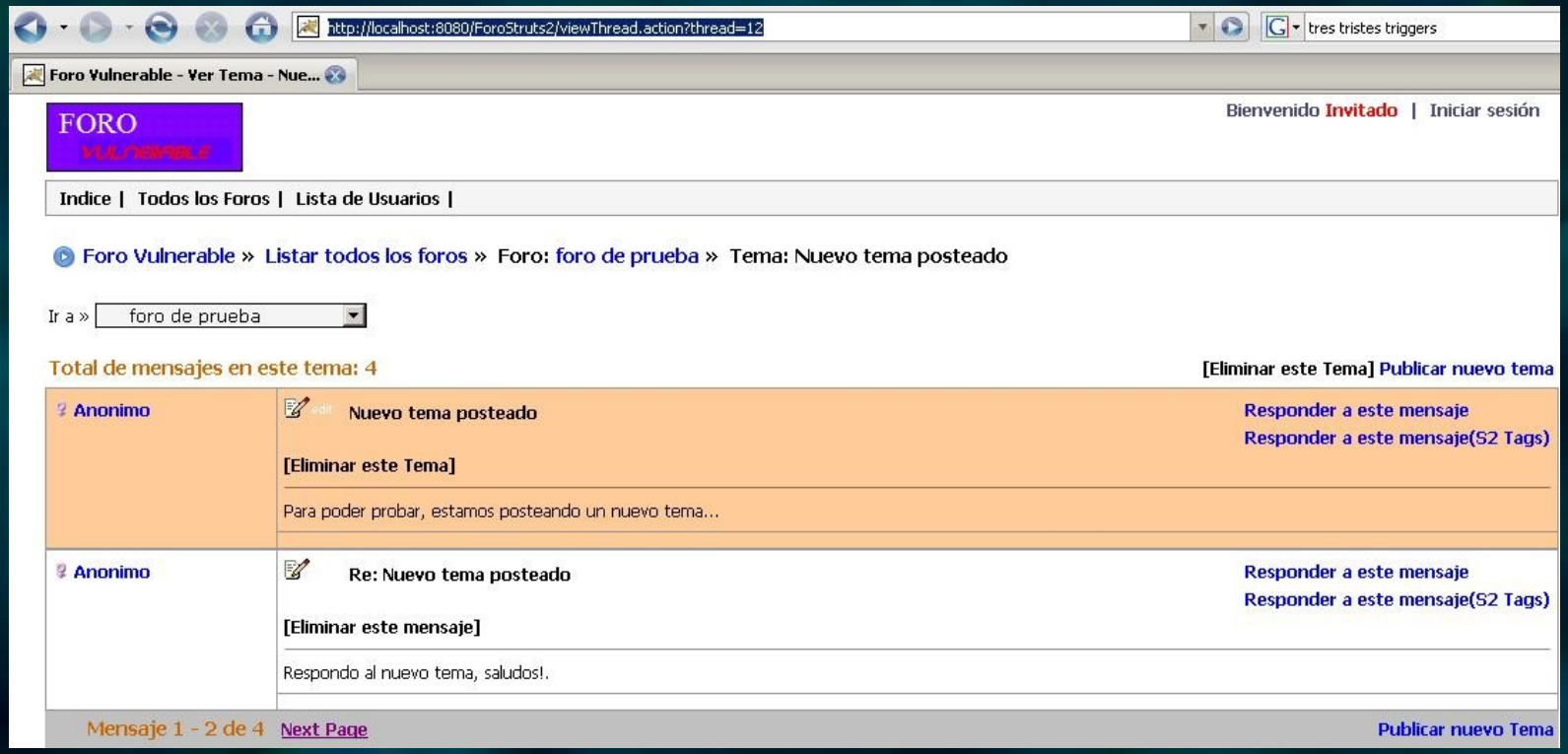
- Consecuencias
  - Comprometer cuentas o datos confidenciales de usuarios del sistema.
  - Permitir el acceso a funciones de la administración del sistema.

Ausencia de autenticación y control de acceso en Foro Vulnerable

- Se accede a la siguiente URL:

  http:// coe.info.unlp.edu.ar:8081/foroVulnerable/viewThread.action?thread=33

- La aplicación muestra el tema correspondiente al id 33 sin pedir autenticación:



Aspecto de Seguridad para Autenticación y Autorización

- **Aspecto de Autenticación y Autorización:**
  - Provee Autenticación y Autorización.
  - Solución basada en Jaas (Java Authentication and Authorization Service).
  - Incorpora control de acceso para las funciones expuestas.

Aspecto de Seguridad para Autenticación y Autorización

```
public aspect AspectoJaas{

   public pointcut ejecucionMetodoExecute():
        execution (public String  ActionSupport+.execute());

   before():  ejecucionMetodoExecute() {
        . . .

        /* Si el usuario no está logueado →
              Autenticación JAAS */
        . . .

        /* Autorización JAAS */

        . . .
   }
}
```

Manejo inadecuado de errores

- Se muestra al usuario final información detallada como rastreos de pila, volcados de base de datos y códigos de error.
- Toda información que un atacante recibe acerca de una aplicación es un arma muy valiosa.
  - Selección de mecanismos de ataque.
  - Ataques de negación de servicio.
- Generalmente los mensajes de error contienen información valiosa para el debugging, y están pensados para personas que deben solucionar dichos errores.

Manejo inadecuado de errores

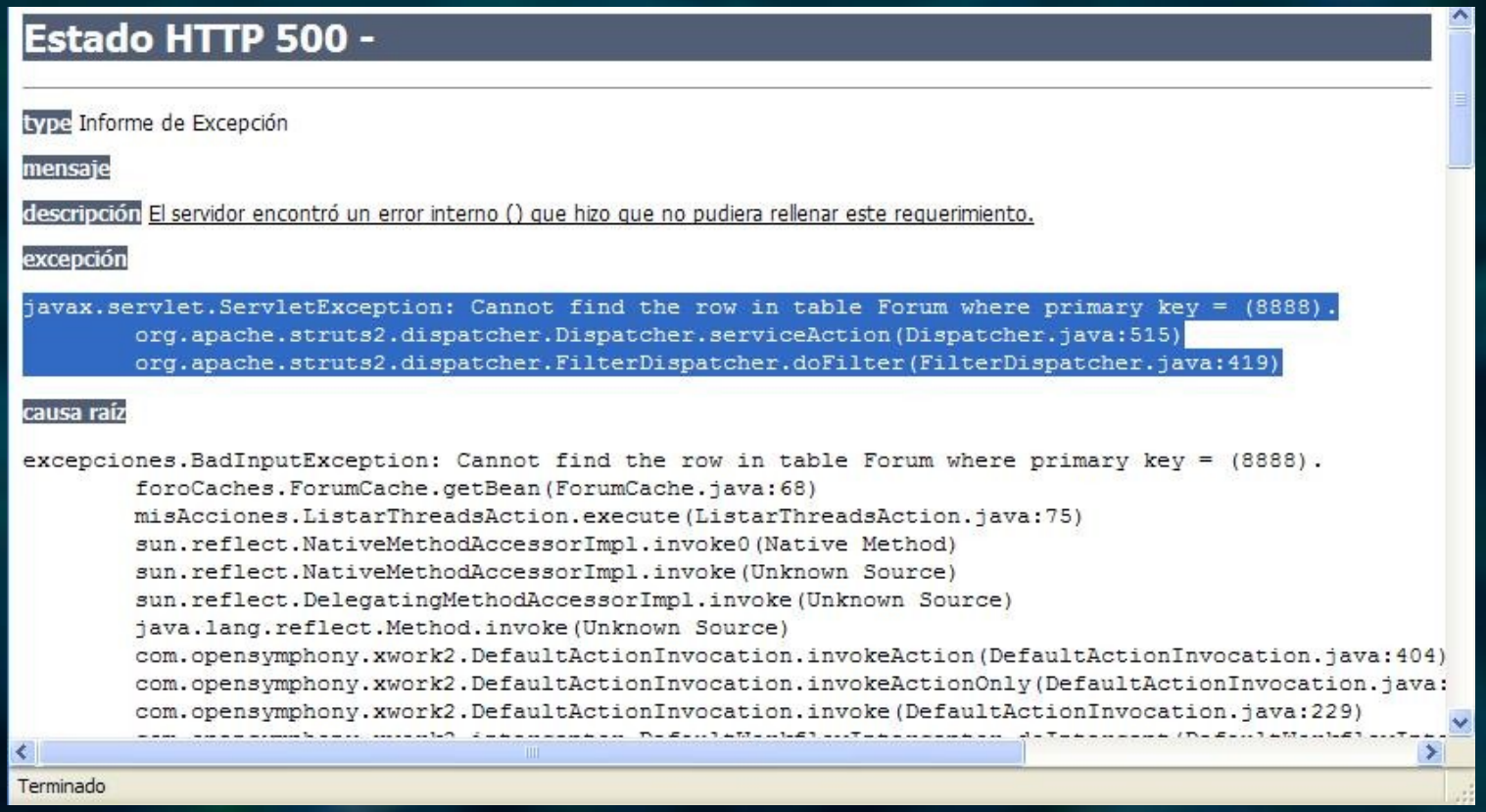
- Tratamiento:
  - Política clara para el manejo de errores.
  - Respuestas útiles, específicamente diseñadas para los usuarios finales.
  - Páginas por defecto para los errores más comunes y para manejo de excepciones.
  - Ciertos errores deben ser almacenados en lugares seguros para su posterior análisis.
  - Evitar tener manejadores de errores generales que no permitan distinguir entre diferentes tipos de errores.

Ausencia de manejo de errores en Foro Vulnerable

- Se accede a la siguiente URL:

  http://coe.info.unlp.edu.ar:8081/foroVulnerable/listarThreads.action?forumID=8888

- La aplicación muestra en pantalla el error de que no existe el foro con id 8888



Aspecto de Seguridad para manejo de errores

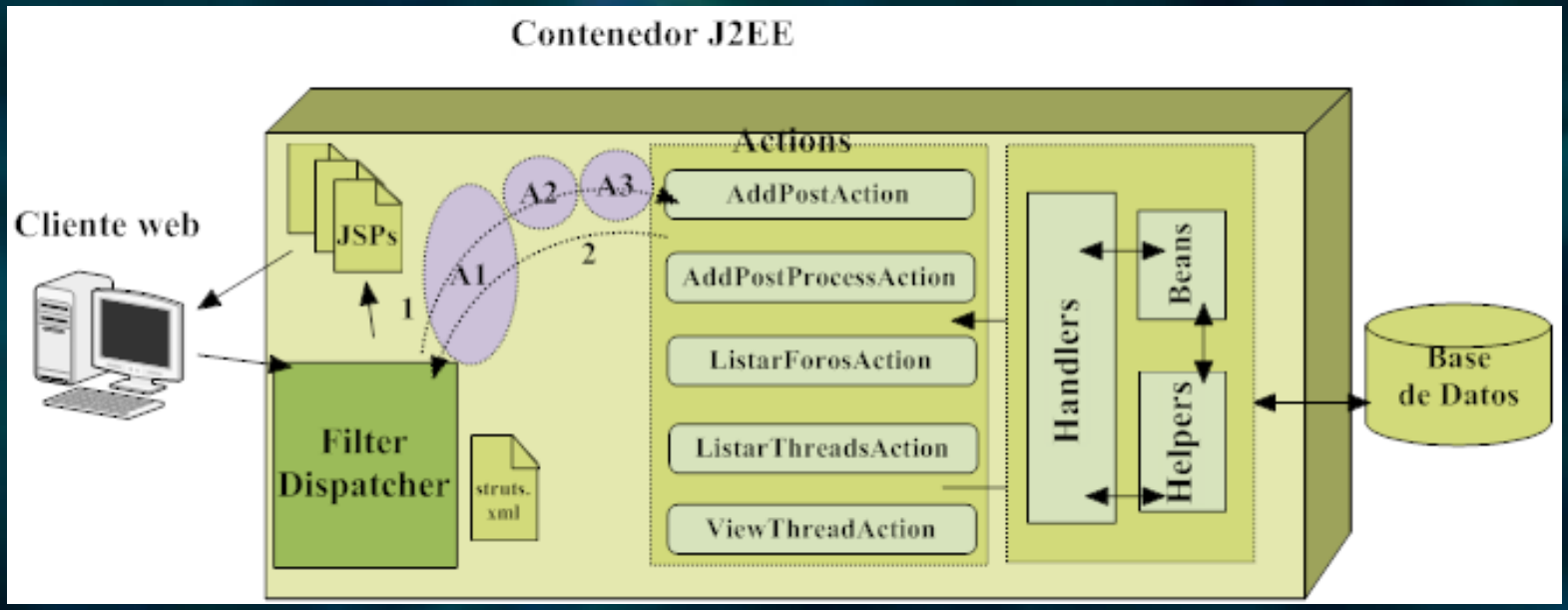
- **Aspecto para manejo de errores:**
  - Tratamiento genérico de errores.
  - Oculta información sensible.
  - Registra errores usando log4j.

Aspecto de Seguridad para manejo de errores

```
public aspect AspectoIEH {

    public pointcut catchingErrors()
          : execution (public String ActionSupport+.*());

    String around() : catchingErrors() {
          try {
                    . . .
                    String ret = proceed();
                    . . .
                    return ret;
          } catch ( . . .) {
                    /* Logging del error */
                    /* Redirección según error*/
          }
    }

    public pointcut catchingErrorsInterno():
          set (private LoginContext LasawLoginAction.lc) ;

    void around(): catchingErrorsInterno() {
          try {
                    proceed();
          } catch (. . .) {
                    /* Logging de error en autenticación */
                    /* Se dispara una nueva excepción */
          }
    }
}
```

Aplicando aspectos a Foro Vulnerable: Foro Seguro

- Aspectos de seguridad + Foro Vulnerable → **Foro Seguro**.
- La incorporación de los **aspectos de seguridad** es transparente para el desarrollo de **Foro Vulnerable**



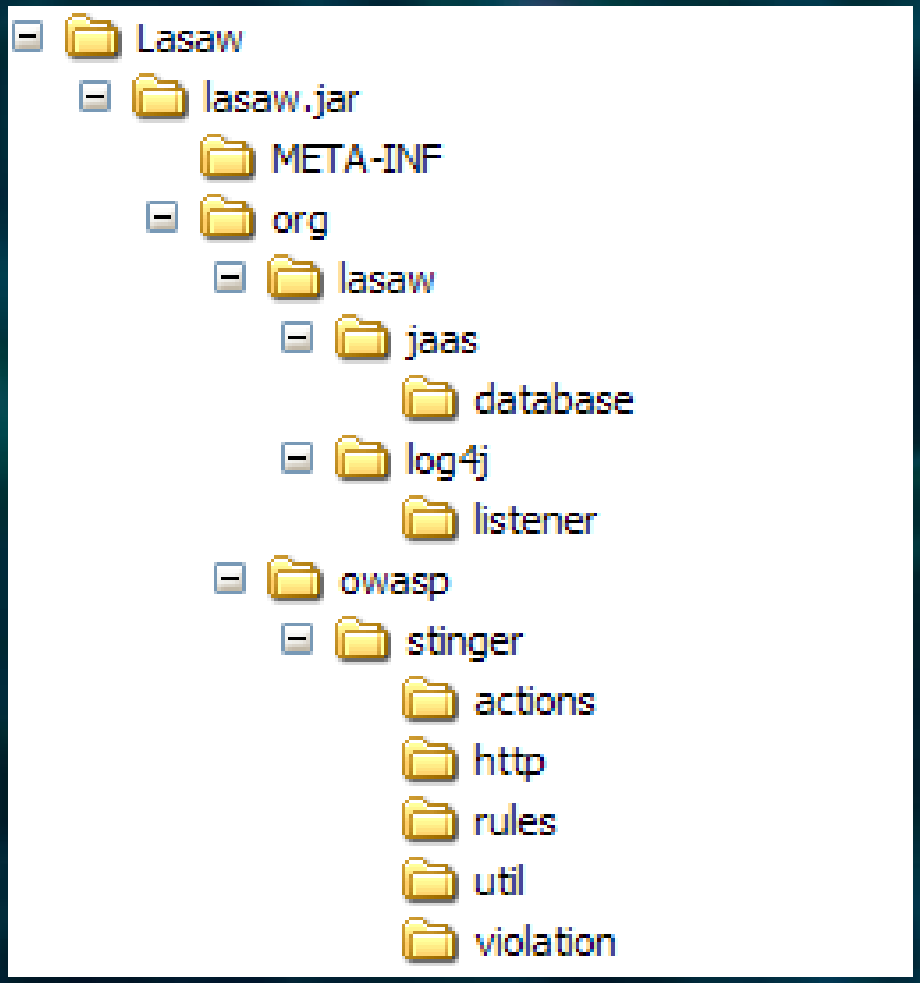
LASAW - Librería de Aspectos de Seguridad para Aplicaciones Web

- **LASAW** → Integración de los Aspectos de seguridad en una librería.

- Componentes de LASAW:
  - Aspectos de seguridad.
  - Aspecto de coordinación.
  - Excepciones LASAW.
  - Presentación LASAW.
  - Auxiliares de seguridad.

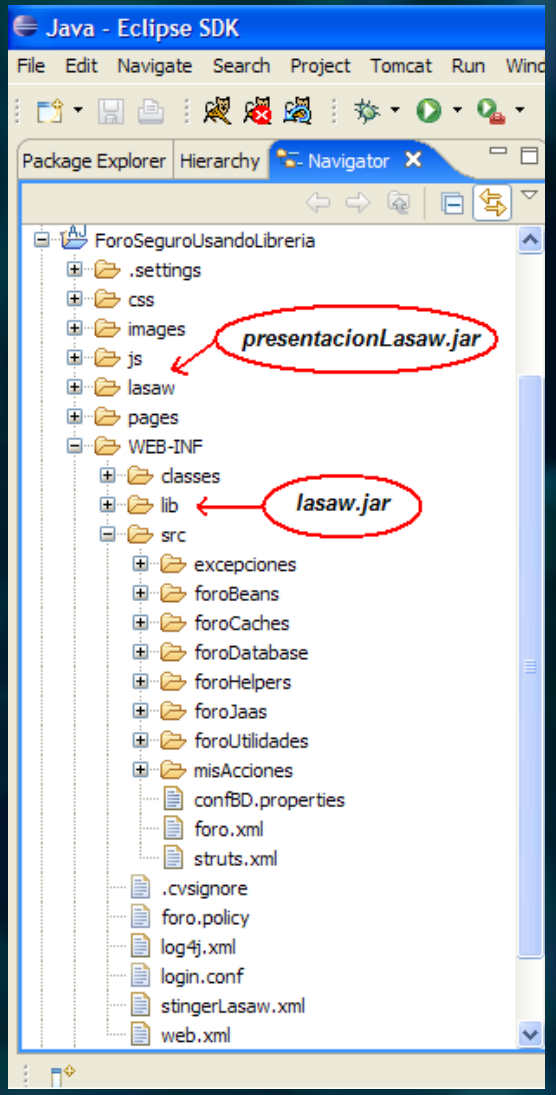
LASAW - Librería de Aspectos de Seguridad para Aplicaciones Web

- Distribución de LASAW
  - Compilación de aspectos
  - Generación de jars.
    - ***lasaw.jar***
    - ***presentacionLasaw.jar***



LASAW - Librería de Aspectos de Seguridad para Aplicaciones Web

- Una manera simple de utilizar LASAW
  - Agregar el lasaw.jar como librería a la aplicación.
  - Descomprimir presentacionLasaw.jar.
  - Weaving en tiempo de compilación.
  - Generación y despliegue de un archivo war.



GLASAW: LASAW Genérica

- Objetivo: lograr que **LASAW** pueda ser utilizada en un conjunto mayor de aplicaciones JAVA.
- Uso de mecanismos de generalización ofrecidos por **AspectJ** para construir soluciones más genéricas y reusables: **pointcuts abstractos.**
- Identificación de los puntos donde se desea hacer la personalización en cada aspecto: **"hot spots"**.
  - Variación de los lugares donde los aspectos van a ser aplicados.
  - Intercambio de las implementaciones de seguridad. Concepto de "mecanismo abstracto", de acuerdo al patrón de diseño **Strategy**.
- Los aspectos abstractos son extendidos para su utilización.
- Aspectos de LASAW como extensiones de los aspectos de GLASAW

Generalizando un aspecto de LASAW

```
public abstract aspect AspectoValidacionGenerico {

   public abstract pointcut ejecucionMetodo();

   before() throws Exception: ejecucionMetodo() {

         . . .
         request = Validador.obtenerRequest();

         response = Validador.obtenerResponse();

         Validador.validar(request,response);

         . . .

   }
}
```

Aspectos vs Servlet Filtros

- Ambos proveen intercepción
- Ventajas de POA
  - Granularidad → Permite intercepciones sobre cualquier objeto de una aplicación.
  - Aplicable sobre aplicaciones no necesariamente web.
  - Mejores posibilidades en el tratamiento de:
    - Autenticación y control de acceso deficientes
    - Manejo inadecuado de errores

Conclusiones

- Usando **Programación Orientada a Aspectos** se logra mejorar la modularidad de las aplicaciones.
- POA es efectivo para el tratamiento de seguridad en aplicaciones.
- Con POA, la implementación de la funcionalidad primaria de la aplicación permanece independiente a la de la funcionalidad secundaria, en nuestro caso Foro Seguro es funcionalmente independiente de su lógica de seguridad.
- **POA** generaliza el concepto de intercepción a distintas aplicaciones en diversos entornos, no necesariamente web.

Conclusiones

- Los aspectos de seguridad se integran sencillamente al resto de la funcionalidad en forma de librería.
- **LASAW** se puede incorporar en otras aplicaciones más allá de Foro Vulnerable.
- Generalización de **LASAW.** Reuso de las soluciones implementadas y posibilidad de agregar tratamiento para otras vulnerabilidades.
- Herramientas suficientemente maduras y amigables para el desarrollo de aspectos.

Trabajos Futuros

- Extensión de la librería para tratar nuevas vulnerabilidades.
- Generar distintas alternativas para las soluciones de seguridad incluidas en LASAW.
- Instanciación de la librería GLASAW para diferentes tipos de aplicaciones WEB Java.
- Estudio de POA dentro de la Ingeniería de Software y sobre el diseño de aplicaciones con POA.
- Análisis de la eficiencia del uso de aspectos utilizando weaving en tiempo de carga/ejecución.

Muchas gracias