



SEGURIDAD EN APLICACIONES WEB CON APACHE TOMEE

Ing. Javier Mantilla Portilla



OWASP

The Open Web Application Security Project





- Quien soy ?
 - Especialista en Ingenieria de Software
 - 10 Años experiencia en desarrollo
 - Desarrollador JAVA , PHP
 - Autodidacta de nuevas tecnologías
 - Ingeniero de desarrollo UPB - Bucaramanga
 - Sector salud y gobierno



➤ Apache TomEE

➤ Arquitectura de la Aplicación

1. Verificación de los componentes a utilizar (A9)
2. Inyección SQL (A1)
3. Autenticación y gestión de sesiones (A2)
4. Uso perfiles y validación de objetos solicitados (A4,A7)
5. Manejo de Filter y Listener(A4)
6. Encriptar datos sensibles (A6)
7. Validación de parámetros (A3) (Js, ManagedBean)
8. Configuraciones de seguridad (A5)



OWASP

The Open Web Application Security Project

Apache Tomcat



7

+



Apache TomEE™





- Apache Tomcat™ es una aplicación de software de código abierto de las tecnologías Java Servlet y JavaServer Pages
- Java EE Web Profile (“Web Profile”), una descripción de la plataforma Java, Enterprise Edition dirigido específicamente a las aplicaciones web
- Apache TomEE se trata de un servidor de aplicaciones certificado como Java EE 6 Web Profile Compatible y basado en Tomcat
- Primefaces es una librería de componentes JSF para facilitar la creación de aplicaciones Web



- **TomEE:**

- **CDI** - Apache OpenWebBeans (Ej. @SessionScoped)
- **EJB** - Apache OpenEJB (Ej. @Entity, @stateless)
- **JPA** - Apache OpenJPA (Ej. javax.persistence, JPQL)
- **JSF** - Apache MyFaces
- **JSP** - Apache Tomcat
- **JSTL** - Apache Tomcat
- **JTA** - Apache Geronimo Transaction(Java Transaction API)
- **Servlet** - Apache Tomcat
- **Javamail** - Apache Geronimo JavaMail
- **Bean Validation** - Apache Bval (Ej. @NotNull(message="Image type must be specified."))



➤ Apache TomEE

➤ Arquitectura de la Aplicación

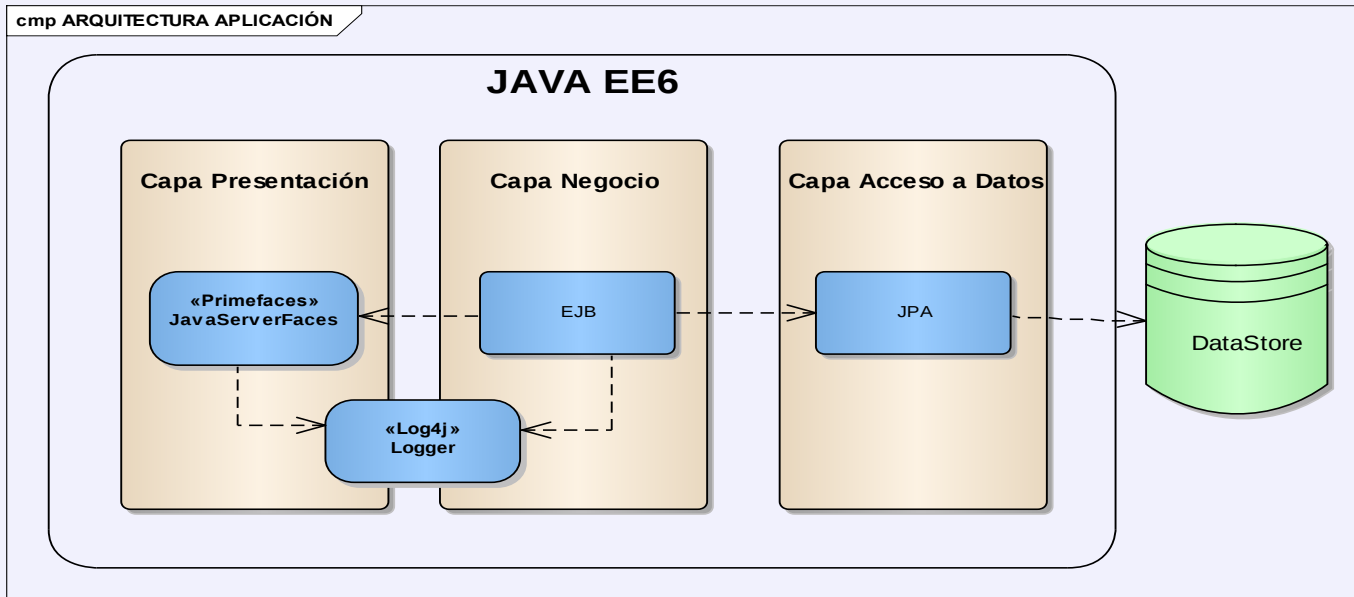
1. Verificación de los componentes a utilizar (A9)
2. Inyección SQL (A1)
3. Autenticación y gestión de sesiones (A2)
4. Uso perfiles y validación de objetos solicitados (A4,A7)
5. Manejo de Filter y Listener(A4)
6. Encriptar datos sensibles (A6)
7. Validación de parámetros (A3) (Js, ManagedBean)
8. Configuraciones de seguridad (A5)



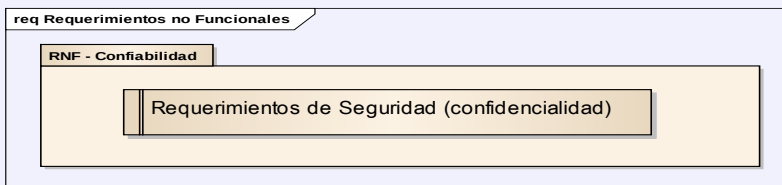
OWASP

The Open Web Application Security Project

- Manejo de capas



- Requerimientos No funcionales de seguridad





- Apache TomEE
- Arquitectura de la Aplicación
- 1. Verificación de los componentes a utilizar (A9)
- 2. Inyección SQL (A1)
- 3. Autenticación y gestión de sesiones (A2)
- 4. Uso perfiles y validación de objetos solicitados (A4,A7)
- 5. Manejo de Filter y Listener(A4)
- 6. Encriptar datos sensibles (A6)
- 7. Validación de parámetros (A3) (Js, ManagedBean)
- 8. Configuraciones de seguridad (A5)

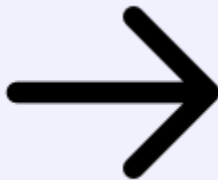
Verificación componentes a utilizar (A9)



OWASP

The Open Web Application Security Project

- Identificar componentes y versión utilizada
- Políticas de actualización de componentes sobre vulnerabilidades corregidas
- Actualizaciones de seguridad.



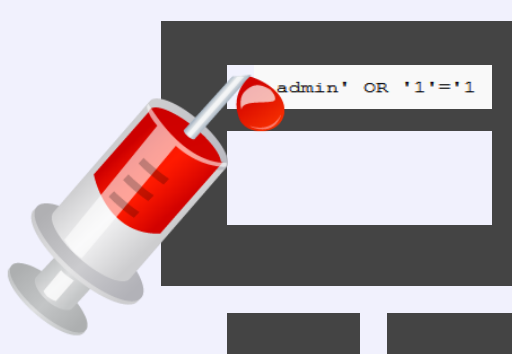


- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. Uso perfiles y validación de objetos solicitados (A4,A7)
 5. Manejo de Filter y Listener(A4)
 6. Encriptar datos sensibles (A6)
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)



- En Java para SQL, utilice un PreparedStatement con parámetros ligados
- Utilización de ORM (HIBERNATE)

```
conn = pool.getConnection( );
String sql = "select * from user where username='" + username + "' and password='" + password + "'";
stmt = conn.createStatement();
rs = stmt.executeQuery(sql);
if (rs.next()) {
    loggedIn = true;
    out.println("Successfully logged in");
} else {
    out.println("Username and/or password not recognized");
}
```



```
String selectStatement = "SELECT * FROM User WHERE userId = ? ";
PreparedStatement prepStmt = con.prepareStatement(selectStatement);
prepStmt.setString(1, userId);
ResultSet rs = prepStmt.executeQuery();
```

```
select * from user where username='admin' OR '1'='1' and password=' '
```

Inyección SQL (A1)



OWASP

The Open Web Application Security Project

Apache TomEE

```
persistence.xml
Design Source History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence-1.0.xsd">
5   <persistence-unit name="sibico-PU" transaction-type="JTA">
6     <provider>org.hibernate.ejb.HibernatePersistence</provider>
7     <jta-data-source>sibico-ds</jta-data-source>
8     <class>com.bramanti.sibico.model.persistentes.AplAuditoria</class>
9     <class>com.bramanti.sibico.model.persistentes.AplMenu</class>
10    <class>com.bramanti.sibico.model.persistentes.AplModulo</class>
11    <class>com.bramanti.sibico.model.persistentes.AplRolMenu</class>
12    <class>com.bramanti.sibico.model.persistentes.AplRol</class>
13    <class>com.bramanti.sibico.model.persistentes.AplUsuario</class>
14    <class>com.bramanti.sibico.model.persistentes.AplUsuarioRol</class>
15    <class>com.bramanti.sibico.model.persistentes.ConContrato</class>
16    <class>com.bramanti.sibico.model.persistentes.ConContratoDetalle</class>
17    <class>com.bramanti.sibico.model.persistentes.GenGrupo</class>
18    <class>com.bramanti.sibico.model.persistentes.GenLista</class>
19    <class>com.bramanti.sibico.model.persistentes.GenTercero</class>
20    <exclude-unlisted-classes>true</exclude-unlisted-classes>
21    <properties>
22    </properties>
23  </persistence-unit>
24  <persistence-unit name="sibico-huella-PU" transaction-type="JTA">
25    <provider>org.hibernate.ejb.HibernatePersistence</provider>
26    <jta-data-source>sibico-huella-ds</jta-data-source>
27    <class>com.bramanti.sibico.model.persistentes.BioHuella</class>
28    <exclude-unlisted-classes>true</exclude-unlisted-classes>
29    <properties>
30    </properties>
31  </persistence-unit>
32 </persistence>
```

```
@Entity
@Table(name = "apl_usuario")
@NamedQueries({
    @NamedQuery(name = "AplUsuario.findAll", query = "SELECT a FROM AplUsuario a")})
public class AplUsuario implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "IdUsuario")
    private Long idUsuario;
    @Basic(optional = false)
    @NotNull
    @Column(name = "Codigo")
    private String codigo;
    @Basic(optional = false)
    @NotNull
    @Column(name = "Clave")
    private String clave;
    @Basic(optional = false)
    @NotNull
    @Column(name = "Nombre")
    private String nombre;
    @Basic(optional = false)
    @NotNull
    @Column(name = "FechaRegistro")
    @Temporal(TemporalType.TIMESTAMP)
    private Date fechaRegistro;
    @Basic(optional = false)
    @NotNull
    @Column(name = "Habilitado")
    private boolean habilitado;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
    private List<GenLista> genListaList;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
    private List<ConContrato> conContratoList;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "apl_usuario")
    private List<AplUsuarioRol> aplUsuarioRolList;
```

Inyección SQL (A1)



OWASP

The Open Web Application Security Project

Apache TomEE

```
@Stateful
public class UsuarioServicesBean implements UsuarioServicesLocal, Serializable {

    /**
     * Entity Manager para la Unidad de Persistencia de la base de datos de sibico.
     */
    @PersistenceContext(unitName = UtilRecursosLogica.SIBICO_PU)
    private transient EntityManager em;

    @Override
    public AplUsuario getUsuarioByCodigo(String codigo, String contrasena) throws BramantiException {
        AplUsuario usuario = null;
        try {
            String jpql = "SELECT u FROM AplUsuario u WHERE u.codigo= :codigo AND u.habilitado=true "
                + "and u.clave= :clave";
            usuario = this.em.createQuery(jpql, AplUsuario.class).setParameter("codigo", codigo)
                .setParameter("clave", contrasena)
                .getSingleResult();
            usuario.getAplUsuarioRolList().size();
            for (AplUsuarioRol usuarioRol : usuario.getAplUsuarioRolList()) {
                usuarioRol.getAplUsuarioRolPK().toString();
                usuarioRol.getAplRol().toString();
                for (AplRolMenu rolMenu : usuarioRol.getAplRol().getAplRolMenuList()) {
                    rolMenu.getAplRolMenuPK().toString();
                }
            }
        } catch (NoResultException nr) {
            usuario = null;
        } catch (Exception e) {
            LOGGER.error("getUsuarioByCodigo -->" + e.getMessage(), e);
            throw new BramantiException(e.getMessage(), e);
        }
        return usuario;
    }
}
```

```
@Transactional(value = TransactionAttributeType.REQUIRES_NEW)
private void guardarPersistenteUsuario() throws BramantiException {
    try {
        List<AplUsuarioRol> listaRoles = this.usuarioUtil.getUsuario().getAplUsuarioRolList();
        this.usuarioUtil.getUsuario().setAplUsuarioRolList(null);
        this.em.persist(this.usuarioUtil.getUsuario());
        for (AplUsuarioRol usuarioRol : listaRoles) {
            usuarioRol.getAplUsuarioRolPK().setIdUsuario(
                this.usuarioUtil.getUsuario().getIdUsuario());
            this.em.persist(usuarioRol);
        }
        this.usuarioUtil.getUsuario().setAplUsuarioRolList(listaRoles);
    } catch (Exception e) {
        throw new BramantiException(e.getMessage(), e);
    }
}
```

```
@Transactional(value = TransactionAttributeType.REQUIRES_NEW)
private void actualizarPersistenteUsuario() throws BramantiException {
    try {
        String jpql = "UPDATE AplUsuarioRol b SET b.habilitado = false "
            + "WHERE b.aplUsuarioRolPK.idUsuario = :idUsuario ";
        Query queryMensaje = em.createQuery(jpql);
        queryMensaje.setParameter("idUsuario", this.usuarioUtil.getUsuario().getIdUsuario());
        queryMensaje.executeUpdate();
        this.em.merge(this.usuarioUtil.getUsuario());
    } catch (Exception e) {
        throw new BramantiException(e.getMessage(), e);
    }
}
```



- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. Uso perfiles y validación de objetos solicitados (A4,A7)
 5. Manejo de Filter y Listener(A4)
 6. Encriptar datos sensibles (A6)
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)

Autenticación y gestión de sesiones (A2)



OWASP

The Open Web Application Security Project

Apache TomEE

- Usar siempre https para cualquier URL autenticados
- Si se almacenan claves en una base de datos, deben ser encriptados.
- Los tiempos de espera de la sesión a un tiempo tan bajo como sea posible para reducir el riesgo de exposición a alguien que se olvida de cerrar la sesión.
- Parámetros en el Login de tipo POST no GET (JSF son post)



```
AplUsuario usuarioValidar = this.getFacadeServices().getUsuarioServices()
    .getUsuarioByCodigo(aplUsuario.getCodigo(), UtilSeguridad.encriptarPassword(this.contrasena));
```

```
public static String encriptarPassword(String password)
    throws NoSuchAlgorithmException {
    MessageDigest messageDigest = MessageDigest.getInstance("SHA-1");
    messageDigest.update(password.getBytes());
    byte[] datosByte = messageDigest.digest();
    StringBuilder stringBuffer = new StringBuilder();
    for (int i = 0; i < datosByte.length; i++) {
        stringBuffer.append(
            Integer.toString((datosByte[i] & 0xff) + 0x100, 16).substring(1));
    }
    return stringBuffer.toString();
}
```

web.xml

```
<session-config>
  <session-timeout>
    30
  </session-timeout>
</session-config>
```

IdUsuario	Codigo	Clave
1	13544171	7c4a8d09ca3762af61e59520943dc26494f8941b
4	37896334	7c4a8d09ca3762af61e59520943dc26494f8941b

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

Autenticación y gestión de sesiones (A2)

Apache TomEE



OWASP

The Open Web Application Security Project

```
@ManagedBean
@SessionScoped
public class SessionManagedBean {

    //<editor-fold defaultstate="collapsed" desc="Atributos">
    /**
     * Usuario.
     */
    private AplUsuario usuario;
    /**
     * Idioma de la aplicacion.
     *
     * @return idioma
     */
    private String idiomaAplicacion;

    //</editor-fold>
    Getter / Setter

    Metodos
}
```

```
private void asignarUsuarioSession() throws BramantiException {
    try {
        if (this.getSessionManagedBean() == null || (this.getSessionManagedBean() != null
            && this.getSessionManagedBean().getUsuario() == null)) {
            SessionManagedBean session = new SessionManagedBean();
            JsflUtil.getRequest().getSession().setAttribute("creado", true);
            session.setUsuario(this.aplUsuario);
            session.setIdiomaAplicacion(idiomaAplicacion);
            JsflUtil.getRequest().getSession().setAttribute(
                "sessionManagedBean", session);
            JsflUtil.getRequest().getSession().setAttribute(
                "userName", this.aplUsuario.getNombre());
        }
        if (getSessionManagedBean().getUsuario() != null) {
            FacesContext.getCurrentInstance().getExternalContext().
                .redirect("faces/vista/generales/principal.xhtml");
        } else {
            FacesContext.getCurrentInstance().getExternalContext().
                .redirect("faces/login.xhtml");
        }
    } catch (IOException e) {
        JsflUtil.addErrorMessage(JsflUtil.getMensajeError());
        LOGGER.error("Método asignarUsuarioSession -->" + e.getMessage(), e);
    }
}
```

```
public static HttpServletRequest getRequest() {
    return (HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest();
}
```

<https://code.google.com/p/owasp-esapi-java/>

Autenticación y gestión de sesiones (A2)



OWASP

The Open Web Application Security Project

Apache TomEE

```
public class SessionFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filter = filterConfig;
    }

    @Override
    public void doFilter(ServletRequest sr, ServletResponse srl, FilterChain fc) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) sr;
        HttpServletResponse response = (HttpServletResponse) srl;
        boolean allowFilterChain = redirectToAvoidJsessionId((HttpServletRequest) sr, (HttpServletResponse) srl);
        // Obtenemos el parámetro sId
        loginPage = UtilRecursosVista.getStringRecursos("https") + request.getServerName() + ":" + request.getServerPort()
            + UtilRecursosVista.getStringRecursos("contextoAplicacion");
        fullLoginPage = UtilRecursosVista.getStringRecursos("contextoAplicacion") + "faces/login.xhtml";

        if (request.getRequestURI().equals(UtilRecursosVista.getStringRecursos("contextoAplicacion"))
            || request.getRequestURI().equals(fullLoginPage)) {
            if (allowFilterChain) {
                fc.doFilter(sr, srl);
            }
        } else if (request.getSession(false) == null
            && request.getRequestId() != null
            && !request.isRequestedSessionIdValid()) {
            //cerrarSesion(request);
            if ("partial/ajax".equals(request.getHeader("Faces-Request"))) {
                response.setContentType("text/xml");
                response.getWriter().append("<?xml version='1.0' encoding='UTF-8'?'>").
                    printf("<partial-response><redirect url='%s'></redirect></partial-response>", loginPage);
            } else {
                response.sendRedirect(loginPage);
            }
        }
    }
}
```

```
<filter>
    <description>Filtro para validar la sesión del usuario</description>
    <filter-name>SessionFilter</filter-name>
    <filter-class>com.bramanti.sibico.backingbean.generales.SessionFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>SessionFilter</filter-name>
    <url-pattern>*.xhtml</url-pattern>
</filter-mapping>
```



web.xml



- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. **Uso perfiles y validación de objetos solicitados (A4,A7)**
 5. Manejo de Filter y Listener(A7)
 6. Encriptar datos sensibles (A6)
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)



- Verificar las referencias directas por usuario o sesión
- Comprobar el acceso de los usuarios a objetos o paginas permitidas.
- La implementación del mecanismo debería negar todo acceso por defecto, requiriendo el establecimiento explícito de permisos a roles específicos para acceder a cada funcionalidad
- El proceso para gestión de accesos y permisos debería ser actualizable y auditable fácilmente



OWASP

The Open Web Application Security Project

Uso perfiles y validación de objetos solicitados (A4,A7)

Apache TomEE

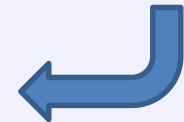


```
public BancariaFzrconbFormBean() {  
    descargarArchivo = Boolean.FALSE.booleanValue();  
    listaArchivos = new ArrayList<SelectItem>();  
    this.isUsuarioAutorizado(EMenu.FZRCONB.getId());  
}
```



```
public void isUsuarioAutorizado(Integer idMenu) {  
    try {  
        MenuFormBean menu = new MenuFormBean();  
        Boolean isAutorizado = menu.getPermisoMenu(idMenu);  
        if (!isAutorizado) {  
            FacesContext.getCurrentInstance().  
                .getExternalContext().redirect("../generales/principal.xhtml");  
        }  
    } catch (Exception e) {  
        e.getMessage();  
    }  
}
```

```
public Boolean getPermisoMenu(Integer idMenu) {  
    Boolean isPermiso = Boolean.FALSE;  
    //isPermiso = Boolean.TRUE;  
    try {  
        for (AplUsuarioRol usuarioRol : this.getSessionManagedBean().getUsuario().getAplUsuarioRolList()) {  
            if (usuarioRol.getHabilitado()) {  
                for (AplRolMenu rolMenu : usuarioRol.getAplRol().getAplRolMenuList()) {  
                    if (rolMenu.getAplRolMenuPK().getIdMenu() == idMenu.intValue()) {  
                        isPermiso = Boolean.TRUE;  
                        break;  
                    }  
                }  
            }  
            if (isPermiso) {  
                break;  
            }  
        }  
    }  
    } catch (Exception e) {  
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(  
            FacesMessage.SEVERITY_ERROR, e.getMessage(), e.getMessage()));  
    }  
    return isPermiso;  
}
```





- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. Uso perfiles y validación de objetos solicitados (A4,A7)
 5. Manejo de Filter y Listener(A7)
 6. Encriptar datos sensibles (A6)
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)



- Filter permite modificar el response y redirigir, en caso necesario, la petición a otro recurso distinto.
- Filter Ideales para el control de acceso de usuarios autenticados
- Filter Interceptan la invocación del servlet ANTES de que sea invocado el propio servlet



```
<filter>
  <description>Filtro para validar la sesión del usuario</description>
  <filter-name>SessionFilter</filter-name>
  <filter-class>com.bramanti.sibico.backingbean.generales.SessionFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>SessionFilter</filter-name>
  <url-pattern>*.xhtml</url-pattern>
</filter-mapping>
```

Manejo de Filter y Listener (A7)



OWASP

The Open Web Application Security Project

Apache TomEE

```
public class SessionFilter implements Filter {
    public void doFilter(ServletRequest sr, ServletResponse srl, FilterChain fc) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) sr;
        HttpServletResponse response = (HttpServletResponse) srl;
        boolean allowFilterChain = redirectToAvoidJsessionId((HttpServletRequest) sr, (HttpServletResponse) srl);
        loginPage = UtilRecursosVista.getStringRecursos("https") + request.getServerName() + ":" + request.getServerPort()
            + UtilRecursosVista.getStringRecursos("contextoAplicacion");
        fullLoginPage = UtilRecursosVista.getStringRecursos("contextoAplicacion") + "faces/login.xhtml";
        if ((request.getRequestURI().equals(UtilRecursosVista.getStringRecursos("contextoAplicacion"))
            || request.getRequestURI().equals(fullLoginPage)) && allowFilterChain) {
            fc.doFilter(sr, srl);
        } else if (request.getSession(false) == null && request.getRequestId() != null
            && !request.isRequestedSessionIdValid()) {

            if ("partial/ajax".equals(request.getHeader("Faces-Request"))) {
                response.setContentType("text/xml");
                response.getWriter().append("<?xml version='1.0' encoding='UTF-8'?'>").
                    printf("<partial-response><redirect url='%s'></redirect></partial-response>", loginPage);
            } else {
                response.sendRedirect(loginPage);
            }
        } else {
            // Obtenemos el sessionManagedBean
            SessionManagedBean sessionManagedBean = getSessionManagedBean(request);
            if (sessionManagedBean == null || sessionManagedBean.getUsuario() == null) {
                irLogin(response);
            } else {
                if (allowFilterChain) {
                    fc.doFilter(sr, srl);
                }
            }
        }
    }
}
```

Manejo de Filter y Listener (A7)



OWASP

The Open Web Application Security Project

Apache TomEE

Monitoreo de
sesiones

```
<listener>
<description>Collector de sesiones importante para los web services.</description>
<listener-class>
    com.bramanti.sibico.backingbean.generales.HttpSessionCollector
</listener-class>
</listener>
```

```
public class HttpSessionCollector implements HttpSessionListener {
    /**
     * Mapa de sesiones.
     */
    private static final Map<String, HttpSession> SESSIONS = new HashMap<String, HttpSession>();

    @Override
    public void sessionCreated(HttpSessionEvent event) {
        HttpSession session = event.getSession();
        session.setAttribute("userName", "XX");
        SESSIONS.put(session.getId(), session);
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent event) {
        SESSIONS.remove(event.getSession().getId());
    }

    public static HttpSession find(String sessionId) {
        return SESSIONS.get(sessionId);
    }
}
```

```
public static void eliminarSesionesWS(String sessionActual) {
    List<String> listaSesionesRemover = new ArrayList<String>();
    for (HttpSession session : SESSIONS.values()) {
        if (!sessionActual.equals(session.getId())
            && session.getAttribute("creado") == null
            && session.getAttribute("userName") == "XX") {
            listaSesionesRemover.add(session.getId());
        }
    }
    for (String sid : listaSesionesRemover) {
        HttpSession ss = SESSIONS.get(sid);
        ss.invalidate();
        SESSIONS.remove(sid);
    }
}
```



- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. Uso perfiles y validación de objetos solicitados (A4,A7)
 5. Manejo de Filter y Listener(A7)
 6. **Encriptar datos sensibles (A6)**
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)



- Asegúrese de aplicar algoritmos de cifrado fuertes y estándar así como claves fuertes y gestiónelas de forma segura.
- Deshabilite el autocompletar en los formularios que recolectan datos sensibles
- Encriptar mensajes XML en cada transmisión (<http://santuario.apache.org/>)

Encriptar datos sensibles (A6)



OWASP

The Open Web Application Security Project

Apache TomEE

```
public static void main(String[] args) {
    String key = "lasquinceletras$"; //llave
    String iv = "0123456789ABCDEF"; // vector de inicialización
    String cleartext = "claveBaseDatos";
    try {
        System.out.println("Texto a encriptar: " + cleartext);
        System.out.println("Texto encriptado: " + encrypt(key, iv, cleartext));
        //System.out.println("Texto desencriptado: " + decrypt(key, iv, encrypt(key, iv, cleartext)));
        System.out.println("Texto desencriptado: " + decrypt(key, iv, "PuxgLDqwx1y8PY8CSZt/xg=="));
    } catch (Exception ex) {
        Logger.getLogger(MainEncryptAES.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

debug:

```
Texto a encriptar: claveBaseDatos
Texto encriptado: PuxgLDqwx1y8PY8CSZt/xg==
Texto desencriptado: claveBaseDatos
```

```
public static String encrypt(String key, String iv, String cleartext) throws Exception {
    Cipher cipher = Cipher.getInstance("CIPHER");
    SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(), "ALGORITMO");
    IvParameterSpec ivParameterSpec = new IvParameterSpec(iv.getBytes());
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivParameterSpec);
    byte[] encrypted = cipher.doFinal(cleartext.getBytes());
    return new String(encodeBase64(encrypted));
}

public static String decrypt(String key, String iv, String encrypted) throws Exception {
    Cipher cipher = Cipher.getInstance("CIPHER");
    SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(), "ALGORITMO");
    IvParameterSpec ivParameterSpec = new IvParameterSpec(iv.getBytes());
    byte[] enc = decodeBase64(encrypted);
    cipher.init(Cipher.DECRYPT_MODE, skeySpec, ivParameterSpec);
    byte[] decrypted = cipher.doFinal(enc);
    return new String(decrypted);
}
```

```
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
```



- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. Uso perfiles y validación de objetos solicitados (A4,A7)
 5. Manejo de Filter y Listener(A7)
 6. Encriptar datos sensibles (A6)
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)

Validación de parámetros (A3) (Js, ManagedBean)



OWASP

The Open Web Application Security Project

Apache TomEE

- Las validaciones de datos de entrada, deben realizarse siempre del lado del servidor, no sólo en el lado del cliente. Los atacantes pueden evitar la validación realizada del lado del cliente modificando valores antes de realizar verificaciones o remover por completo esta validación.
- Validaciones Internas. De cualquier forma, los navegadores permiten desactivar el JavaScript, razón por la cuál se hace necesaria la validación de los campos y de su formato desde el script final. Esto es importante ya que los ataques maliciosos podrían intentar enviar código por un formulario, que al ser leído o desplegado podría ejecutarse y causar serios problemas.

Validación de parámetros (A3) (Js, ManagedBean)

Apache TomEE



OWASP

The Open Web Application Security Project

Usuario

Campo Obligatorio

Contraseña

Campo Obligatorio

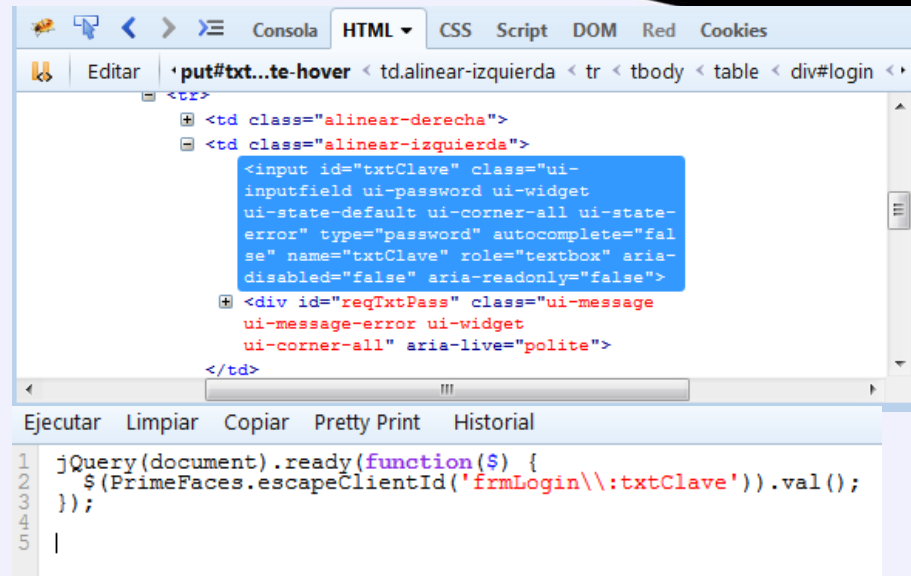
 Ingresar

Usuario

Contraseña

 Ingresar

 Debe digitar el usuario y la contraseña



The screenshot shows a web browser's developer console with the HTML and JavaScript code for a login form. The HTML code defines the structure of the login form, including the user and password input fields and the login button. The JavaScript code uses jQuery to validate the user and password fields before submitting the form.

```
<tr>
  <td class="alinear-derecha">
    <td class="alinear-izquierda">
      <input id="txtClave" class="ui-inputfield ui-password ui-widget ui-state-default ui-corner-all ui-state-error" type="password" autocomplete="false" name="txtClave" role="textbox" aria-disabled="false" aria-readonly="false">
      <div id="reqTxtPass" class="ui-message ui-message-error ui-widget ui-corner-all" aria-live="polite">
    </td>
  </tr>
```

```
jQuery(document).ready(function($){
  $(PrimeFaces.escapeClientId('frmLogin\\:txtClave')).val();
});
```

```
<p:commandButton value="#{msg.lbl_ingresar}"
  id="btnIngresar"
  icon="fa fa-user"
  onStart="ajaxCargarOnStart()"
  onComplete="mostrarMensaje();PF('dlgCargando').hide();"
  process="frmLogin"
  update="frmLogin"
  action="#{loginFormBean.envioIngresar()}"
/>
```

```
private Boolean validarParametros() {
  Boolean isValido = Boolean.TRUE;
  if ("".equals(aplUsuario.getCodigo())) {
    isValido = Boolean.FALSE;
  }
  if ("".equals(this.contrasena)) {
    isValido = Boolean.FALSE;
  }
  return isValido;
}
```

```
@ManagedBean
@ViewScoped
public class LoginFormBean extends FormBean implements Serializable {

  public void envioIngresar() {
    try {
      if (this.validarParametros()) {
        this.autenticarSistema();
      } else {
        JsUtil.addWarningMessage("Debe digitar el usuario y la contraseña");
      }
    } catch (BramantiException e) {
      JsUtil.addErrorMessage(JsUtil.getMensajeError());
      LOGGER.error("Método envioIngresar -->" + e.getMessage(), e);
    }
    RequestContext.getCurrentInstance().update("msg");
  }
}
```



- Apache TomEE
- Arquitectura de la Aplicación
 1. Verificación de los componentes a utilizar (A9)
 2. Inyección SQL (A1)
 3. Autenticación y gestión de sesiones (A2)
 4. Uso perfiles y validación de objetos solicitados (A4,A7)
 5. Manejo de Filter y Listener(A7)
 6. Encriptar datos sensibles (A6)
 7. Validación de parámetros (A3) (Js, ManagedBean)
 8. Configuraciones de seguridad (A5)



- Mantener y desplegar las nuevas actualizaciones y parches de software, y también las librerías de código utilizadas (A9)
- Arquitectura de aplicación que proporcione una separación segura entre los componentes (Lógica de negocio. Clave BD).
- Auditoria a los logs de la aplicación y del sistema.
- Configuración de alertas
- Desactivar servicios no utilizados
- Mensajes de error excesivamente informativos
- Directorios con permisos indebidos

Configuraciones de seguridad (A5)



OWASP

The Open Web Application Security Project

Apache TomEE

```
mkdir /srv/apache-tomee-jaxrs-1.5.2/keystore
```

```
keytool -genkey -alias tomcat -keyalg RSA -keystore /srv/apache-tomee-jaxrs-1.5.2/keystore/cticbga.jks
```

```
sistemas@Apps:~$ /opt/jdk1.6.0_45/bin/keytool -genkey -alias tomcat -keyalg RSA  
-keystore /srv/apache-tomee-jaxrs-1.5.2/keystore/cticUPBbga.jks -validity 365
```

```
sudo nano /srv/apache-tomee-jaxrs-1.5.2/conf/server.xml
```

```
<Connector port="8443"  
    protocol="org.apache.coyote.http11.Http11Protocol"  
    SSLEnabled="true"  
    maxThreads="150"  
    scheme="https"  
    secure="true"  
    clientAuth="false"  
    sslProtocol="TLS"  
    keystoreFile="keystore/cticbga.jks"  
    keystorePass=""  
    keyAlias="tomcat"  
</>
```

Clave del certificado

```
sudo service tomee stop  
sudo service tomee start
```

Configuraciones de seguridad (A5)



OWASP

The Open Web Application Security Project

Apache TomEE

```
javier@srvDesarrollo:~$  
javier@srvDesarrollo:~$  
javier@srvDesarrollo:~$  
javier@srvDesarrollo:~$ /srv/apache-tomee-jaxrs-1.5.2/bin/digest.sh -a SHA cas  
cas :296d14181c0b209f6fe251d193e3f5ebd3ff725d  
javier@srvDesarrollo:~$
```

Clave a encriptar.....

resultado

```
javier@srvDesarrollo:~$ sudo nano /srv/apache-tomee-jaxrs-1.5.2/conf/tomcat-user  
s.xml
```

GNU nano 2.2.6

Archivo: /srv/apache-tomee-jaxrs-1.5.2/conf/tomcat-users.xml

Modificado

```
-->  
<!-- Activate those lines to get access to TomEE GUI -->  
<role rolename="tomee-admin" />  
<role rolename="manager-gui" />  
<role rolename="admin-gui"/>  
<user username="cticbga" password="296d14181c0b209f6fe251d193e3f5ebd3ff725d" roles="tomee-admin,manager-gui,admin-gui" />  
<user username="tomee" password="296d14181c0b209f6fe251d193e3f5ebd3ff725d" roles="tomee-admin" />  
<user password="ide" roles="manager-script,admin" username="ide"/>  
</tomcat-users>
```

Pegar la clave

```
<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
```

```
  <Realm className="org.apache.catalina.realm.MemoryRealm" digest="SHA" />
```

valor a colocar.

```
<!-- SingleSignOn valve, share authentication between web applications  
Documentation at: /docs/config/valve.html -->
```

```
<!--
```

```
<Valve className="org.apache.catalina.authenticator.SingleSignOn" />
```

```
-->
```

```
<!-- Access log processes all example.
```

```
Documentation at: /docs/config/valve.html
```

```
Note: The pattern used is equivalent to using pattern="common" -->
```

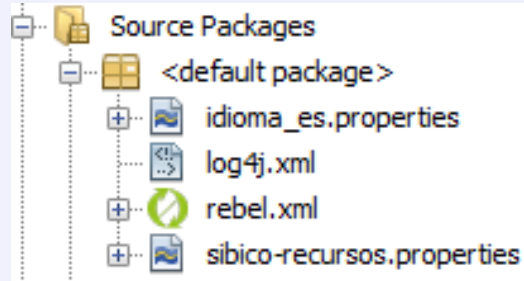
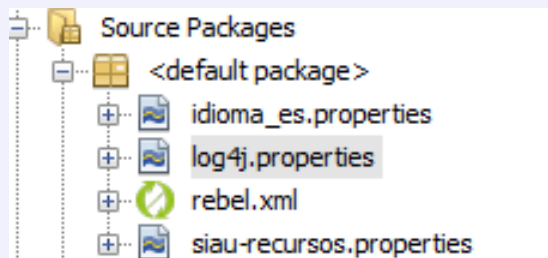
```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" pattern="%h %l %u %t &quot;%r&quot; %s$
```

```
</Host>
```

```
<!-- Use the LockOutRealm to prevent attempts to guess user passwords
```



- Log4j - Apache Software Foundation
- Permite elegir la salida y el nivel de granularidad de los mensajes o “logs”.
- En tiempo de ejecución y no a tiempo de compilación.
- Niveles (trace, debug, info, warn, error, fatal).



```
#####
### Configuracion para LOCAL                                     ###
#####
# log4j.rootLogger = INFO, LOGFILE, CONSOLE, MAIL
log4j.rootLogger = INFO, LOGFILE, CONSOLE

log4j.appender.CONSOLE = org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout = org.apache.log4j.PatternLayout
#log4j.appender.CONSOLE.layout.ConversionPattern = %d{ABSOLUTE} [%t] %-5p %c %x - %m%n
log4j.appender.CONSOLE.layout.ConversionPattern = %d{yyyy-MM-dd HH:mm:ss} %c{1} [%p] %m%n

#####
### Configuracion Comun                                         ###
#####
log4j.appender.LOGFILE = org.apache.log4j.RollingFileAppender
log4j.appender.LOGFILE.file = ${catalina.base}/logs/siau-web.log
log4j.appender.LOGFILE.append = true
# log4j.appender.LOGFILE.DatePattern = '. 'yyy-MM-dd
log4j.appender.LOGFILE.maxFileSize = 50MB
log4j.appender.LOGFILE.maxBackupIndex = 20
log4j.appender.LOGFILE.layout = org.apache.log4j.PatternLayout
#log4j.appender.LOGFILE.layout.ConversionPattern = %d{ABSOLUTE} [%t] %-5p %c %x - %m%n
log4j.appender.LOGFILE.layout.ConversionPattern = %d{yyyy-MM-dd HH:mm:ss} %c{1} [%p] %m%n
```



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
  <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.out"/>
    <param name="Threshold" value="INFO" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d{ABSOLUTE} [%t] %-5p %c %x - %m%n"/>
    </layout>
  </appender>
  <!-- log all logs to a separate log file every day -->
  <appender name="STDOUT" class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="${catalina.base}/logs/biblioteca-web.log" />
    <param name="Threshold" value="INFO" />
    <param name="Append" value="true" />
    <param name="MaxFileSize" value="1MB" />
    <param name="MaxBackupIndex" value="20" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d{ABSOLUTE} [%t] %-5p %c %x - %m%n"/>
    </layout>
  </appender>
  <appender name="MAIL" class="org.apache.log4j.net.SMTPAppender">
    <param name="BufferSize" value="512" />
    <param name="SMTPHost" value="email.upbbga.edu.co" />
    <param name="SMTPUsername" value="apps" />
    <param name="From" value="apps@mail.upbbga.edu.co" />
    <param name="To" value="javier.mantillap@upb.edu.co" />
    <param name="Subject" value="[ERROR LOG BIBLIOTECA/WEB] ErrorList" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d{ABSOLUTE} %5p %c{1}:%L - %m%n" />
    </layout>
    <filter class="org.apache.log4j.varia.LevelRangeFilter">
      <param name="LevelMin" value="error" />
      <param name="LevelMax" value="fatal" />
    </filter>
  </appender>
  <root>
    <priority value="INFO" />
    <appender-ref ref="STDOUT" />
    <appender-ref ref="CONSOLE" />
    <appender-ref ref="MAIL" />
  </root>
</log4j:configuration>
```



```
import org.apache.log4j.Logger;

@ManagedBean
@ViewScoped
public class LoginFormBean extends FormBean implements Serializable {

    private static final Logger LOGGER = Logger.getLogger(LoginFormBean.class);

    private void autenticarSistema() throws BramantiException {
        try {
            AplUsuario usuarioValidar = this.getFacadeServices().getUsuarioServices()
                .getUsuarioByCodigo(aplUsuario.getCodigo(), UtilSeguridad.encriptarPassword(this.contrasena));
            if (usuarioValidar != null) {
                if (!usuarioValidar.getAplUsuarioRolList().isEmpty()) {
                    this.aplUsuario = usuarioValidar;
                    this.asignarUsuarioSession();
                } else {
                    JsفUtil.addWarningMessage(JsfUtil.getMensaje("msg_error_rol_sistema"));
                }
            } else {
                JsفUtil.addWarningMessage(JsfUtil.getMensaje("msg_error_autenticar"));
            }
        } catch (BramantiException e) {
            LOGGER.error("Método autenticarSistemaBiblioteca -->" + e.getMessage(), e);
            throw new BramantiException(e.getMessage(), e);
        } catch (NoSuchAlgorithmException ex) {
            LOGGER.error("Método autenticarSistemaBiblioteca -->" + ex.getMessage(), ex);
            throw new BramantiException(ex.getMessage(), ex);
        }
    }
}
```



- Java EE incluye anotaciones para simplificar la implementación del control de acceso de usuarios.
 - `@DeclareRoles`: Especifica los roles de seguridad asociados con el Bean.
 - `@RolesAllowed`: Sólo los roles declarados en esta anotación podrán invocar el método. Se pueden especificar los roles permitidos a nivel de clase o a nivel de método.
 - `@PermitAll`: Permite la invocación por todos los roles definidos. Se puede especificar a nivel de clase o a nivel de método.
 - `@DenyAll`: Bloquea a todos los roles de manera que ninguno podrá llamarlo. Únicamente a nivel de método.
 - `@RunAs`: Se aplica a nivel de clase e indica el rol con el que se ejecuta
el Bean: `@RunAs("admins")`



- javax.annotation.security.RolesAllowed
- javax.annotation.security.PermitAll
- javax.annotation.security.DenyAll
- javax.annotation.security.RunAs
- javax.annotation.security.DeclareRoles

```
@Stateless
@DeclareRoles({"committer", "contributor"})
public class OpenSourceProjectBean implements Project {

    @Resource SessionContext ctx;

    @RolesAllowed({"committer"})
    public String svnCommit(String s) {
        ctx.isCallerInRole("committer"); // Referencing a Role
        return s;
    }

    @RolesAllowed({"contributor"})
    public String submitPatch(String s) {
        return s;
    }
}
```

```
@Stateless
@DeclareRoles({"committer", "contributor"})
@RolesAllowed({"committer"})
public class OpenSourceProjectBean implements Project {

    public String svnCommit(String s) {
        return s;
    }

    public String svnCheckout(String s) {
        return s;
    }

    @RolesAllowed({"contributor"})
    public String submitPatch(String s) {
        return s;
    }
}
```

<http://tomee.apache.org/security-annotations.html>



OWASP

The Open Web Application Security Project

- <http://tomee.apache.org/tomcat-java-ee.html>
- <http://tomcat.apache.org/tomcat-7.0-doc/index.html>
- [https://www.owasp.org/index.php/Preventing SQL Injection in Java](https://www.owasp.org/index.php/Preventing_SQL_Injection_in_Java)
- <http://logging.apache.org/log4j/1.2/>
- <http://es.wikipedia.org/wiki/Log4j>
- <http://tomee.apache.org/security-annotations.html>
- <https://www.unidata.ucar.edu/software/thredds/current/tds/reference/TomcatSecurity.html>
- <https://www.digicert.com/es/instalar-certificado-ssl-tomcat.htm>
- <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=securitySSLKeytool>
- <http://jaehoo.wordpress.com/2012/05/22/enable-ssl-secure-socket-layer-in-apache-tomcat-5-5-6-y-7/>
- <http://primefaces.org/>



OWASP

The Open Web Application Security Project

¡ Muchas Gracias !

Javier Orlando Mantilla Portilla.

jmantillap@gmail.com

Javier.mantillap@upb.edu.co

www.javiermantilla.co

