

## DAW Práctica 3.5: Mecanismos de seguridad en Apache Tomcat

En esta actividad vamos a implementar algunos de los mecanismos de seguridad que nos proporciona Apache Tomcat.

### Procedimiento:

Instrucciones:

1. Inicia nuestro servidor de aplicaciones de Apache y accede desde la URL a localhost:8080/manager. ¿Hay algún problema?
2. Dentro del directorio “conf” de Tomcat, ¿qué utilidad tiene el archivo tomcat-users.xml?
3. Siguiendo la guía que proporciona el propio fichero, crea un usuario con capacidad para acceder a la interfaz gráfica de manager (manager-gui). ¿Qué funciones tienen los diferentes roles que aparecen en el fichero?
4. Al acceder a manager nos muestra información interesante como las aplicaciones lanzadas o el estado del servidor (**añade una captura de estos apartados**) además de la posibilidad de gestionar las aplicaciones que tenemos iniciadas o desplegar alguna más añadiendo el .war entre otras funcionalidades.
5. Implementaremos la necesidad de autenticación para una de nuestras aplicaciones. Para ello, modificaremos su archivo web.xml desde Eclipse añadiendo las siguientes instrucciones:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Autenticacion</web-resource-name> 1
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>authentication-user</role-name> 2
  </auth-constraint>
</security-constraint>

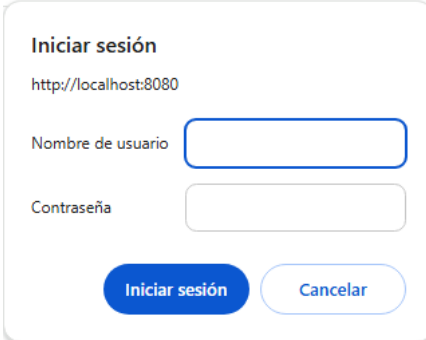
<login-config>
  <auth-method>BASIC</auth-method> 3
</login-config>
```

- (1) definen el nombre del recurso al que le vamos a aplicar la configuración de seguridad (puede ser cualquier nombre) y el patrón de las URLs a las que va a aplicar. En este caso como hemos puesto /\* se aplicará a toda la aplicación.
- (2) define el rol que deberán tener los usuarios que tengan permiso para conectarse.

- (3) configura el tipo de autenticación (Basic). También se podría definir una base de datos externa en la que almacenar los usuarios para aumentar la seguridad.
6. Una vez configurada la aplicación nos quedaría configurar los usuarios dentro del servidor. Para ello deberás modificar el archivo tomcat-users.xml añadiendo el siguiente contenido (el nombre del rol debe coincidir con el especificado en el archivo web.xml):

```
<role rolename="autenticacion-user"/>
<user username="prueba" password="prueba" roles="autenticacion-user"/>
```

7. Solo quedaría actualizar el .war, reiniciar el servidor y probar que al acceder nos pide autenticación.

Un formulario de inicio de sesión con el título "Iniciar sesión" y la URL "http://localhost:8080". Contiene dos campos de entrada: "Nombre de usuario" y "Contraseña". Debajo de los campos hay dos botones: "Iniciar sesión" (azul) y "Cancelar" (gris).

8. Ahora configuraremos el protocolo seguro HTTPS. Para ello, haremos uso de la herramienta Keytool. Keytool es una utilidad de línea de comandos incluida con el JDK (Java Development Kit) que se utiliza para administrar claves y certificados de seguridad.
9. Abriremos la consola en modo administrador y ejecutaremos el siguiente comando (define la utilidad de las extensiones que aparecen en el comando):

```
keytool -genkey -alias tomcat -keyalg RSA -keystore keystore.jks -keysize 2048
```

10. Si no puedes ejecutar el comando, piensa que es una herramienta que proporciona el JDK.
11. Responde a las preguntas de información para generar la clave y el certificado autofirmado.

12. El archivo recientemente creado, que almacena tanto la clave como el certificado debe estar situado en **RutaTomcat/conf**. Podemos moverlo o crearlo directamente en esa dirección.
13. Deberemos editar el archivo server.xml que, si se han seguido los pasos anteriores, debería encontrarse en la misma carpeta que el archivo .jks creado. En dicho fichero deberemos modificar el conector de https o crearlo con la siguiente configuración en caso de que no esté implementado:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" SSLEnabled="true"
    maxParameterCount="1000"
    >
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/keystore.jks"
            type="RSA" certificateKeystorePassword="tucontraseña" />
    </SSLHostConfig>
</Connector>
```

14. Reinicia Apache tomcat y prueba a acceder a localhost desde el puerto implementado para HTTPS (https://localhost:**PUERTOIMPLEMENTADO**).
15. Es posible que no se tenga acceso si el puerto implementado no está abierto. Para abrirlo en Windows:
  1. Ve al Panel de Control → Sistema y seguridad → Firewall de Windows Defender.
  2. Selecciona Configuración avanzada.
  3. Crea una nueva regla de entrada:
    - Tipo: Puerto.
    - Protocolo: TCP.
    - Puerto: 8443 (o el puerto configurado).
    - Permitir la conexión.
    - Asigna un nombre (por ejemplo: "Tomcat HTTPS").
16. Vuelve a intentar acceder utilizando HTTPS. El navegador nos muestra un aviso como hemos visto en prácticas anteriores (al ser un certificado autofirmado).



### La conexión no es privada

Es posible que los atacantes estén intentando robar tu información de **localhost** (por ejemplo, contraseñas, mensajes o tarjetas de crédito). [Más información sobre esta advertencia](#)

NET::ERR\_CERT\_AUTHORITY\_INVALID



[Activa la protección mejorada](#) para tener el máximo nivel de seguridad de Chrome

Avanzado

Volver a un sitio seguro

17. Muestra los detalles del certificado implementado desde el navegador.
18. Recuerda que dentro de Apache Tomcat tienes el directorio **logs**, donde podrás ver los diferentes motivos de posibles errores que te vayan sucediendo durante la implementación.
19. Utilizando el protocolo seguro intenta acceder a las aplicaciones creadas en prácticas anteriores.



20. Investiga que componentes web tiene Apache Tomcat. ¿Se ha utilizado alguno durante la realización de las últimas prácticas?