Securización Port@firmas

CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Documento elaborado por: Francisco José Cantero Villar

Revisado por: Miguel J. Vázquez Rebollo

Versión: 1.2.0
Lugar y fecha: Sevilla, 23 de Junio de 2006

Contenido:

1 Objetivos

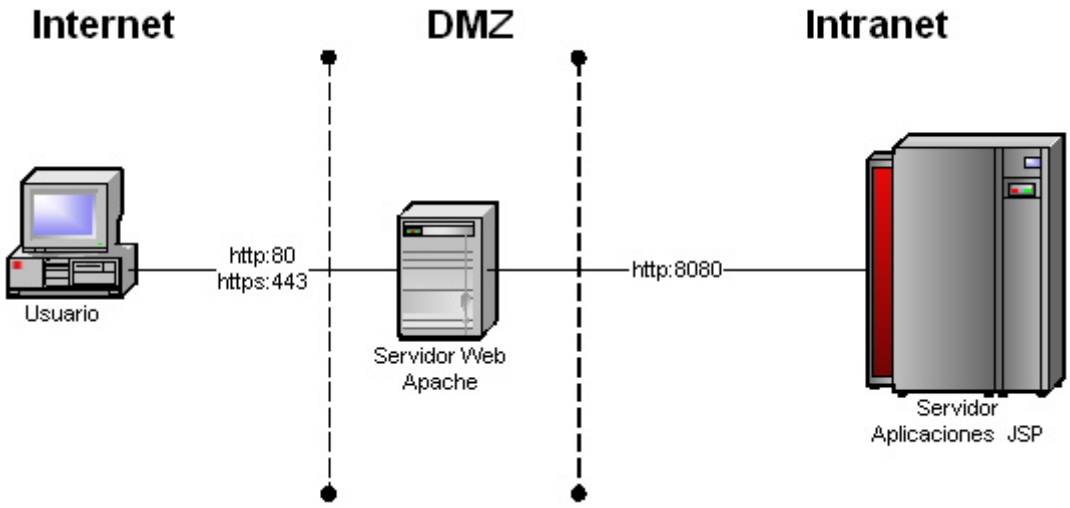
El presente documento tiene como objetivos indicar los distintos mecanismos para securizar la instalación de la aplicación Web de Port@firmas mediante distintas técnicas existentes e implementadas sobre los servidores de aplicaciones Apache y Tomcat.

Veremos primero como cifrar las comunicaciones entre los clientes y el servidor de aplicaciones para que las informaciones no sean legibles salvo por los participantes de la comunicación y seguidamente control de acceso a los servicios web de Port@firmas mediante usuario y clave, para que solo puedan acceder a los mismos aplicaciones que nosotros controlemos o bien podamos denegarles el servicio si nos fuese necesario.

Comentar que introducir cifrado y autenticación entre servidores provoca una caída del rendimiento de los mismos dado que tienen que deben hacer un doble esfuerzo a la hora de cifrar descifrar informaciones. Las técnicas a continuación se van a ver son complementarias entre sí, de tal forma que podemos aplicar solo aquellas que nos interesen.

2 HTTPS: Secure Socket Layer (SSL)

Generalmente la arquitectura de servidores Web viene definida por un servidor fachada Apache publicado al exterior y que presta servicio al contenido estático del portal y servicios básicos de autenticación de usuarios y cifrado mediante https, y redirección de las peticiones a las distintas aplicaciones a los servidores internos no publicados al exterior.



1. Esquema básico de arquitectura Web.

Esta arquitectura tiene la ventaja que lleva la carga de cifrado a este servidor intermedio, y deja que los servidores de aplicaciones sobre los que ruedan las aplicaciones más pesadas funcionen en entornos sin cifrado más livianos.

Generalmente para habilitar el cifrado de Port@firmas en este tipo de arquitectura solo tendremos que ponernos en contacto con los administradores de sistema que controlan el servidor Apache para que nos establezcan lo que se conoce como un PROXYPATH, es decir enrutar una dirección externa hacia nuestro servidor de aplicaciones interno.

Ejemplo: La ruta externa https://servidorWeb/pfirma se enruta a la dirección interna del Port@firmas de nuestro servidor JSP, http://servidorAplicaciones:8080/pfirma.

Importante: Si se desea publicar el Port@firmas por una ruta que no se directamente /pfirma, sino por ejemplo /consejería/pfirma, se ha de crear un alias en el servidor de aplicaciones con la misma ruta apuntando a /pfirma, de tal forma que el PROXYPATH se haría de la siguiente forma:

https://servidorWeb/consejeria/pfirma → http://servidorJSP:8080/consejeria/pfirma

Así nos aseguramos que las cookies de sesión se construyen correctamente en ambos servidores y no se produce pérdidas de sesión.

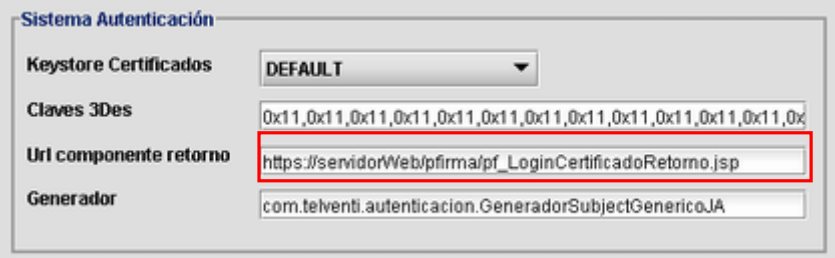
2.1 Configuración Port@firmas HTTPS vía Apache

Llegados a este punto ya tenemos accesible Port@firmas mediante una capa de cifrado SSL de todas sus informaciones con el cliente, solo nos falta revisar que todas las configuraciones del mismo se establecen bajo el mismo protocolo para evitar saltos de HTTPS a HTTP y viceversa.

2.1.1 Configuración de autenticación en @firma

Para funcionar sobre HTTPS deberemos revisar la ruta de retorno de autenticación indicada en el documento ***DOP Port@firmas, punto 4.1, componente de retorno*.**

El valor que deberemos poner será la ruta externa que hemos configurado en el servidor Web, dado que será la ruta con la cual llame el cliente una vez se realice el proceso de autenticación de usuario.



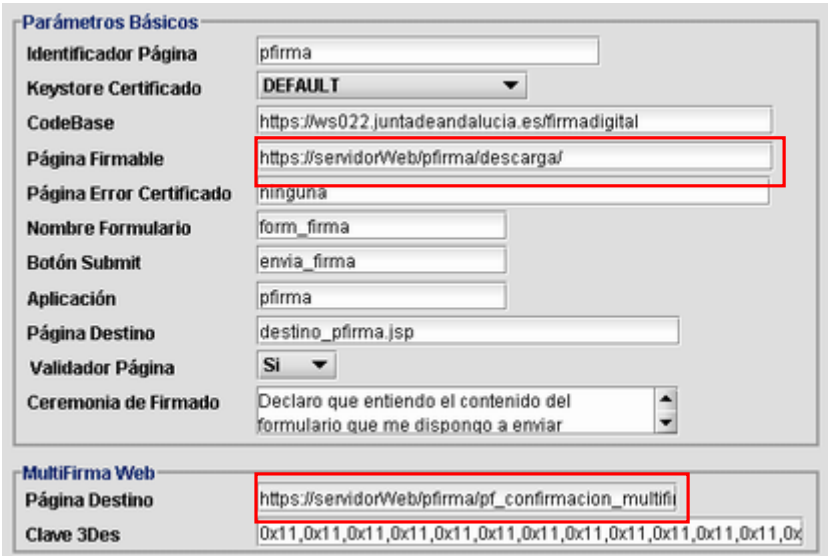
2. Modificación de la Url componente retorno.

Como se aprecia en la imagen, hemos sustituido el valor de "**Url componente retorno**" que accedía directamente al servidor JSP (http://servidorJSP:8080/pfirma/...), por el retorno a través del servidor web externo (https://servidorWeb/pfirma/...).

2.1.2 Configuración de firma web en @firma

En el mismo punto de dicho documento DOP se define la configuración para habilitar la firma de formularios web dentro de Port@firmas. En este caso deberemos también revisar que las rutas que requiere @firma para realizar la firma web de los mismos siga la misma coherencia y se hagan a su vez en modo HTTPS.

Para ello, en la configuración de @firma para aplicaciones web revisaremos los parámetros básicos "**página firmable**", y en multifirma web "**página destino**", llevando a cabo los mismos cambios que en el caso de autenticación, sustituyendo las referencias del servidor de aplicaciones por el servidor Web vía HTTPS.

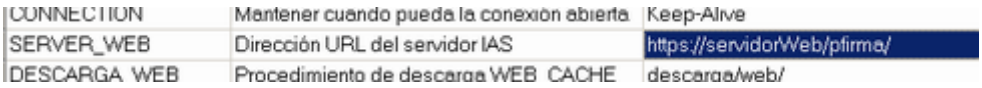


3. Configuración de firma web.

2.1.3 Configuración de constantes en Port@firmas

El último cambio que debemos hacer se realiza dentro de una de las constantes de Port@firmas, por tanto entraremos con un cliente SQL, bien TOAD/TORA o RAPTOR y editaremos los valores de la tabla PF_CONSTANTES.

Nos centraremos en el valor (V_CONSTANTE) de la variable SERVER_WEB, y al igual que los casos anteriores cambiaremos la configuración para que pase por el servidor web.



4. Constante SERVER_WEB

Tras esto ya hemos acabado de configurar la aplicación para que funcione correctamente por HTTPS, de forma que a partir de ahora los usuarios deberán entrar por la dirección:

https://servidorWeb/pfirma

2.2 Configuración Port@firmas HTTPS vía Tomcat

En el esquema de la arquitectura web expuesta en el gráfico 1, se aprecia que la

comunicación entre el servidor web y el servidor de aplicaciones se hacen en modo HTTP, pero si nos interesa podemos hacer que esta comunicación se haga a su vez vía HTTPS configurando correctamente un nuevo conector en el mismo que establezca la comunicación bajo SSL.

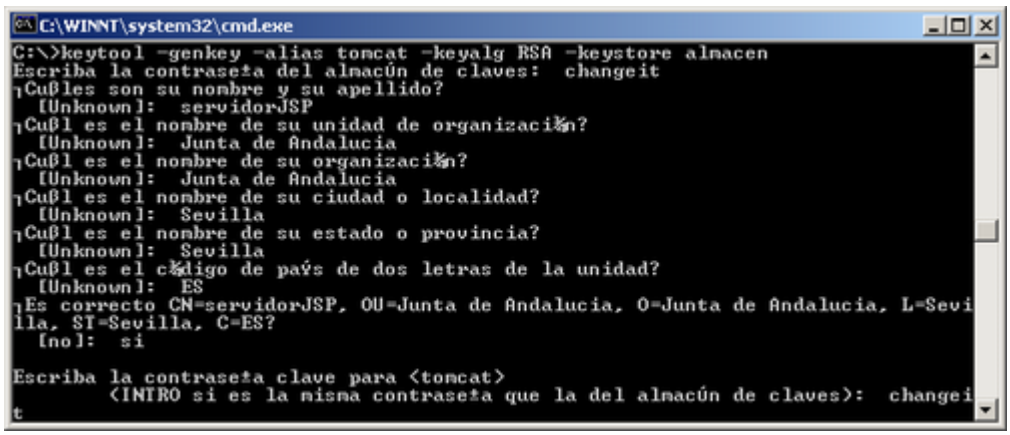
Si no disponemos de un servidor Apache publicado al exterior, pero nos interesa que los usuarios ataquen directamente a nivel de intranet el servidor Tomcat donde está desplegada la aplicación podemos introducir fácilmente una capa de securización bajo SSL.

Veamos el caso del ejemplo para un Apache Tomcat.

2.2.1 Creación de un certificado para el servidor Tomcat

Lo primero es crear un almacén de certificados con un certificado en su interior que se llame “tomcat”, para ello deberemos usar la herramienta de java *keytool*, con el que podremos hacer un certificado interno.

Para ello vamos a ver el caso de ejemplo de crear un certificado “tomcat” dentro de un almacén de certificado que guardaremos en “C:\almacen”



5. Creación de certificado para SSL.

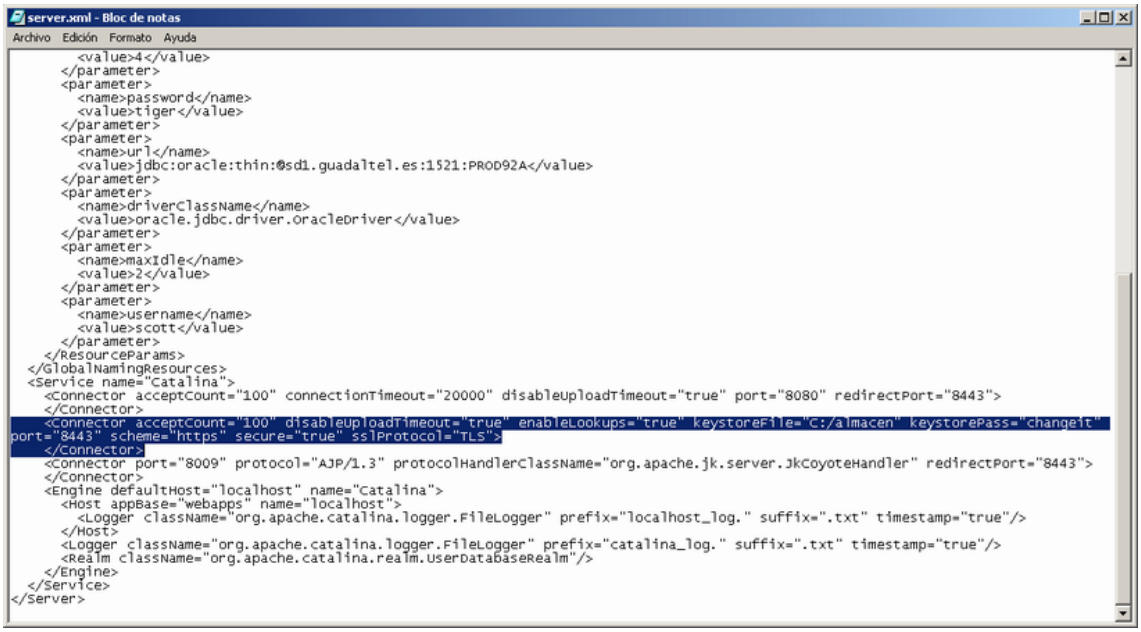
Importante es tener en cuenta que debemos poner como “nombre y apellido” el nombre del servidor de aplicaciones JSP o bien su dirección IP, para que coincida el certificado publicado con el nombre de la máquina, si no, es frecuente que luego de problemas para la interfaz SSL en WebService.

Tras esto ya tenemos un certificado para poder pasar al siguiente punto.

2.2.2 Configurar el conector SSL en Tomcat

Para ello deberemos editar el fichero server.xml que se encuentra dentro del directorio

del Tomcat, en la carpeta “/conf”. Haremos una copia del mismo por seguridad y editaremos en el original en la parte del final las siguiente líneas.

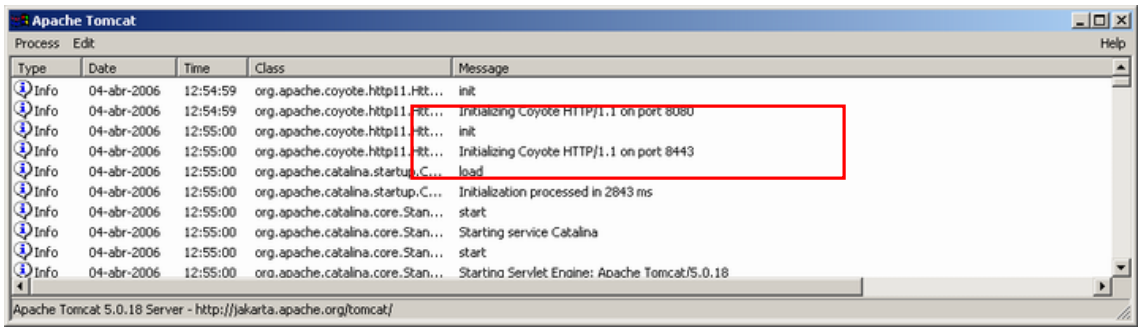


6. Server.xml del servidor Tomcat.

La línea en concreto a añadir es la que sigue:

```
<Connector acceptCount="100" disableUploadTimeout="true" enableLookups="true" keystoreFile="C:/almacen"
keystorePass="changeit" port="8443" scheme="https" secure="true" sslProtocol="TLS">

</Connector>
```

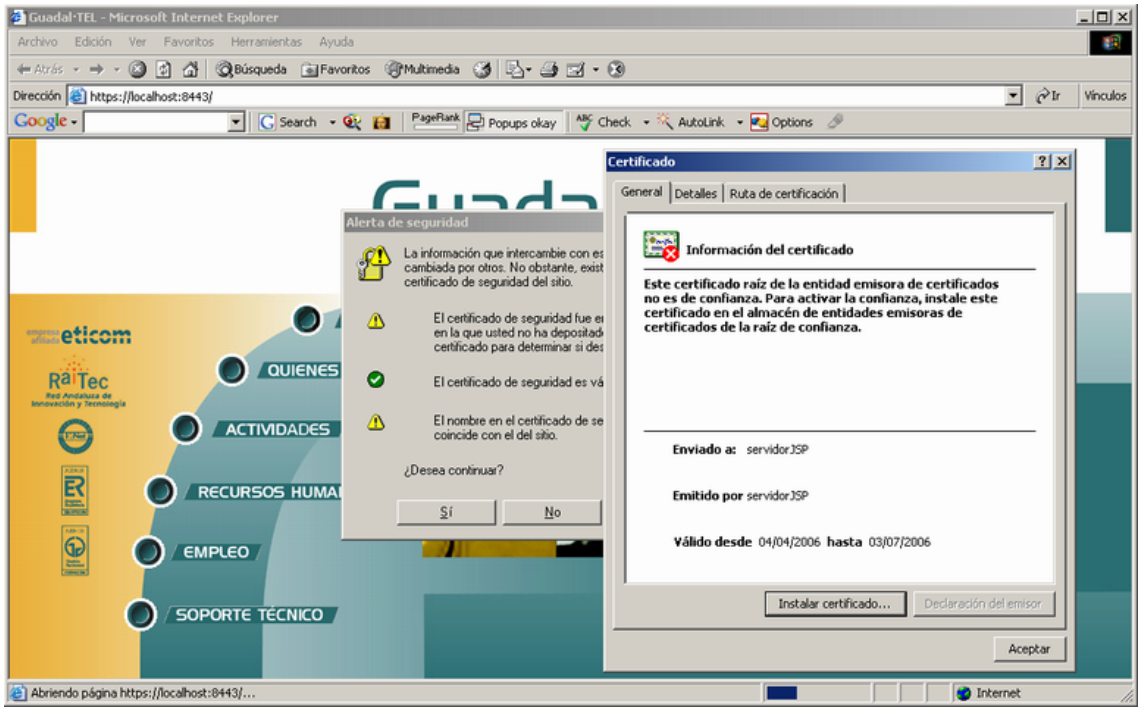
Tras cambiarlo reiniciaremos el Tomcat para que tome los cambios y al arrancar nos deberá salir el siguiente mensaje que nos confirma que se ha habilitado el SSL correctamente.



7. Log de arranque del Tomcat.

Para comprobar con nuestro navegador el correcto funcionamiento del mismo pondremos en la barra de direcciones:

https://servidorJSP:8443/ (en el ejemplo localhost)



8. Acceso a Tomcat por HTTPS.

Tras esto ya tenemos el servidor de aplicaciones funcionando bajo HTTPS.

2.2.3 Configuración de Port@firmas

Tras haber configurado el servidor de aplicaciones bajo HTTPS, deberemos repetir los pasos descritos en el punto 2.1 del presente documento pero teniendo en cuenta que la dirección ahora de nuestra aplicación será https://servidorJSP:8443/pfirma

2.3 Resolviendo SSL desde clientes Web Service

Al haber securizado las comunicaciones de HTTP a HTTPS, los clientes web service, es decir, las aplicaciones clientes que envían peticiones o consultan el estados de las mismas deben que adaptarse a este tipo de comunicación.

El proceso muy sencillo, lo único que hay que hacer es sacar la clave pública del certificado del servidor y cargarlo en el almacén de certificados del JDK sobre el que se está ejecutando, generalmente dentro **"%JAVA_HOME%/jre/lib/security**", el fichero "**cacerts**".

La ruta y clave del almacén de certificados también se puede definir por código por si queremos indicar uno específico.

```
System.setProperty("javax.net.ssl.trustStore",keystorePath);
System.setProperty("javax.net.ssl.trustStorePassword",keystorePassword);
```

Hay que tener cuidado con cambiar estos parámetros al tratarse de parámetros globales de la JVM, por tanto se ven afectadas todas las aplicaciones que funcionan bajo la misma máquina virtual. Lo más recomendable es usar un solo almacén de certificados para todas las aplicaciones y que en él se carguen todos los certificados necesarios.

2.4 Problemas conocidos bajo entornos SSL

A continuación vamos a enumerar una serie de problemas conocidos, así como las soluciones o caminos alternativos para resolver los mismos.

2.4.1 Cliente Web Service: No Trusted Certificated Found

Esta excepción salta cuando la máquina de java no es capaz de validar el certificado del servidor, es decir, no lo encuentra dentro de su almacén de certificados aceptados.

Este error se produce en los siguientes casos:

- La JDK es de la versión 1.4.2_02 o 1.4.2_03.
    - Causa: Estas versiones tienen un “bug” interno que se corrigió en versiones posteriores.
    - Solución: Cambiar a una versión más reciente, recomendado usar 1.4.2_07 o superior.
- La JDK es correcta pero sigue saltando esta excepción.
    - Causa: Realmente no encuentra el certificado en el almacén. El certificado no se encuentra dentro del fichero “cacerts” o la variable de sistema apunta a otro almacén que no lo contiene.
    - Solución: Comprobar que está leyendo correctamente la ruta del almacén y que este a su vez es correcto. Para ello activar como parámetro de arrancar de la máquina de java “**–Djavax.net.debug=all**”, parametro que saca toda la traza de la comunicación SSL.

2.4.2 Open Office: Una llamada a la api no ha finalizado correctamente

Siempre es altamente recomendable que los documentos binarios que se envíen a Port@firmas para que firmen sean formatos no editables por el usuario, es decir, imágenes o bien documenos PDF´s. En el caso de enviar documentos Open Office directamente hay que tener cuidado de haber indicado correctamente el tipo mime de Open Office para que Port@firmas pueda abrir la aplicación correspondiente para visualizarlo.

Open Office soporta directamente protocolo HTTP. Es decir, no descarga el fichero en local y luego lo abre si no que directamente trata de abrirlo en línea para no tener que descargarlo completamente para poder visualizarlo.

Esto en entornos SSL provoca el error anteriormente comentado.

- Causa: El problema radica en que si bien soporta HTTP, no soporta HTTPS ni JAAS, por lo cual no puede abrir estos ficheros por llegar cifrados o bien tener el acceso denegado.

- Solución: La solución es sencilla, solo hay que cambiar el valor de la constante SERVER_WEB y definir las descargas en modo HTTP.

23	CONNECTION	Mantener cuando pueda la conexión abierta	Keep-Alive
24	SERVER_WEB	Dirección URL del servidor IAS	http://localhost:8080/pfirma/
25	DESCARGA_WEB	Procedimiento de descarga WEB_CACHE	descarga/web/

3 JAAS: Java Authentication and Authorization Service

En este punto vamos a desarrollar los mecanismos no tanto ya de asegurar el contenido de las informaciones, sino, que solo pueda acceder a las mismas los usuarios autorizados.

Vamos a abordar entonces los dos frentes desde los cuales se puede atacar a la aplicación, osea, el portal web de cara al usuario accede y firma, y el interfaz webservice que presta los servicios de recepción y consulta de peticiónes de firma a terceras aplicaciones.

3.1 Autenticación de usuarios

Port@firmas a priori, no restringe el acceso a los usuarios, de tal forma que todo usuario que entre se da de alta automáticamente. En este alta o en posteriores accesos se guarda la dirección IP desde la que accede dicho usuario si funciona la resolución inversa de direcciones para poder hacer un seguimiento de los accesos del mismo.

Si trabajamos a nivel de intranet esta es una buena política dado que no hay un conjunto de posibles usuarios muy amplio y un usuario solo puede firmar lo que le llega o bien dar de alta nuevas peticiones que quedan a su vez asociadas al mismo. (esta última opción se puede deshabilitar de la barra de herramientas).

En caso de publicarlo a internet sería recomendable establecer algún mecanismo de acceso basado en usuario y clave, que mediante un servicio de autenticación *SingleSignOn* permita solo a determinados usuarios acceder a los recursos publicados en el servidor web y este a su vez, redirigidos a los servidores de aplicaciones.

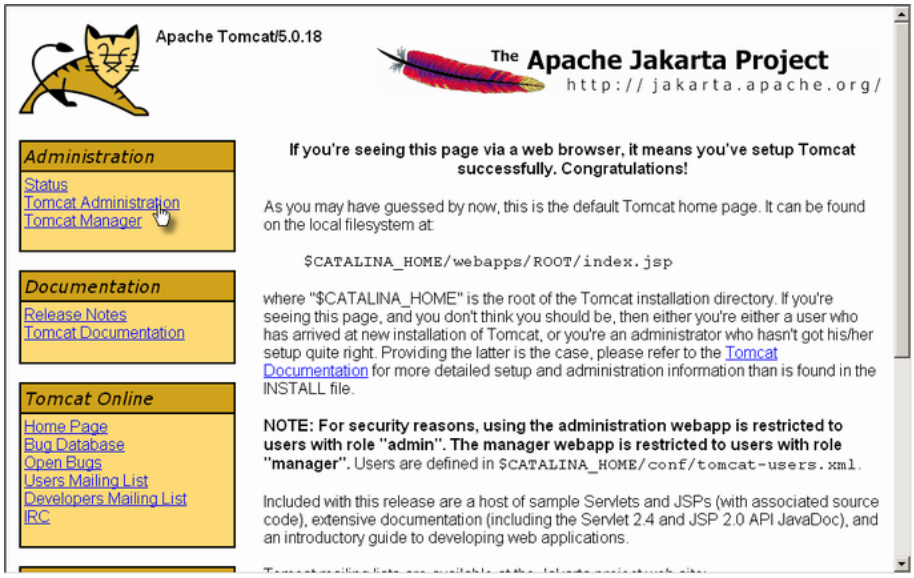
Si este mecanismo de seguridad se establece en el servidor Web, deberá ser el administrador de dicho servidor el que defina la política del mismo para todo el conjunto de las aplicaciones, si por lo contrario se relega este definición de forma individual para cada una de las aplicaciones, vamos a ver a continuación como podemos configurar esta para el servidor de aplicaciones Tomcat de una forma sencilla.

3.1.1 Creación de Rol

Para establecer la seguridad debemos primero acceder a la herramienta de administración del Tomcat. Para ello escribiremos en la barra de direcciones de nuestro navegador la siguiente dirección:

http://servidorJSP:8080/

Tras esto nos saldrá la página principal del Tomcat tal y como se muestra la imagen.

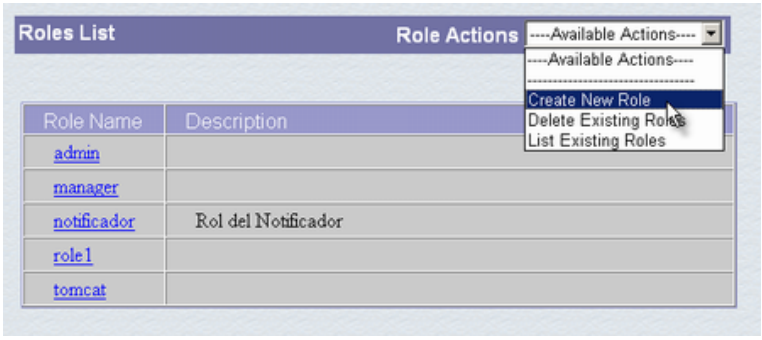


9. Página principal de Tomcat.

Entramos en la herramienta de administración, y tras introducir el usuario y clave del mismo, nos sale el siguiente árbol de opciones. Pulsamos sobre “Roles” y nos aparece la siguiente pantalla.

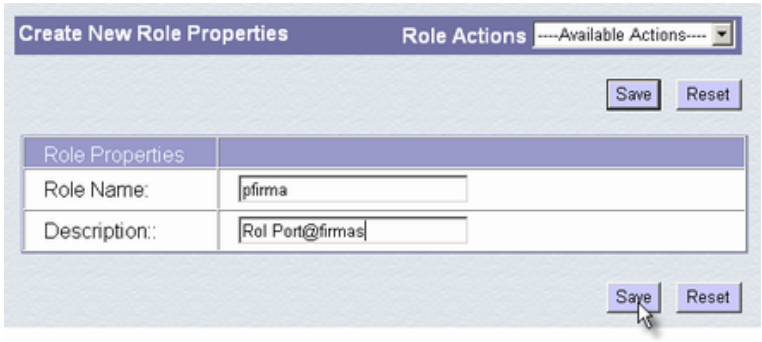
10. Lista de roles definidos en Tomcat.

Pulsamos sobre “Role Actions”, y escogemos “Crear New Role”.



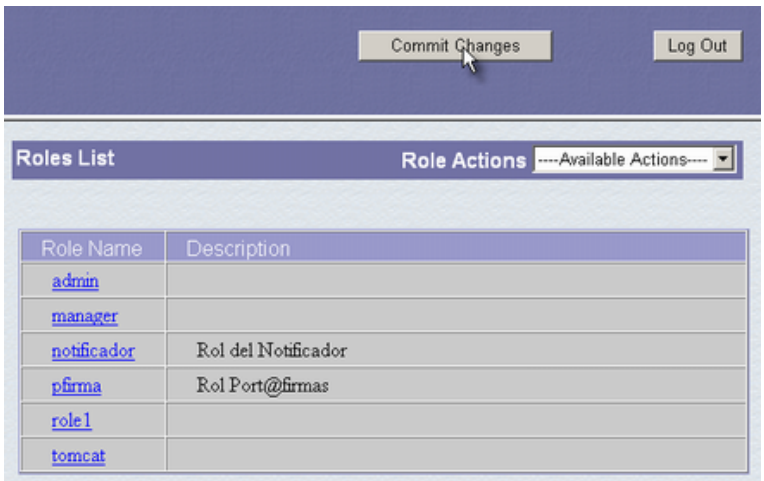
11. Crear un nuevo rol.

Tras esto nos sale un formulario donde indicamos el nombre del rol y una descripción del mismo, lo rellenamos en este caso como indica la imagen.



12. Alta de un nuevo rol.

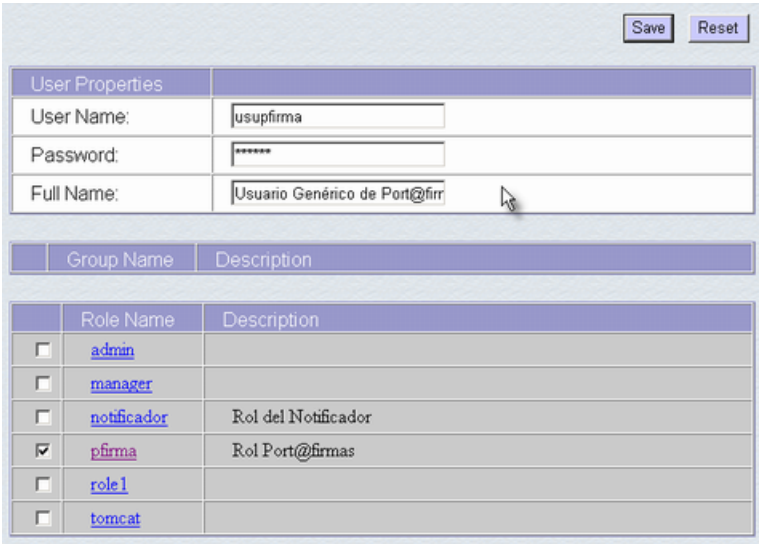
Tras guardar los cambios, los sale la lista de roles existentes, para llevar a efecto estos cambios debemos hacer “Commit Changes”, tal y como se muestra en la imagen.



13. Llevar a efecto los cambios.

3.1.2 Creación de Usuario

El siguiente paso sería crear un usuario, el proceso es similar al de creación de Rol, lo importante radica en la pantalla donde se especifican los datos del mismo le incluyamos el rol que acabamos de crear tal y como se muestra en la imagen.



14. Creación de usuario.

Al igual que antes una vez creado deberemos darle a “Commit Changes” para que se lleven a efecto los cambios.

Desde estas pantallas podemos quitarle el rol, volver añadírselo, cambiar la clave o bien dar de baja el usuario.

3.1.3 Restringir el acceso al portal web Port@firmas

Ahora lo único que queda es indicar a Tomcat, que solo los usuarios con el rol creado tienen acceso a la aplicación, para ello deberemos incluir las siguientes líneas dentro del **web.xml** que hay dentro de la ruta “**/pfirma/WEB-INF/**”

15. Restricción de acceso de seguridad.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Portal Web Port@firmas</web-resource-name>
    <url-pattern>*.jsp</url-pattern>
    <url-pattern>*.jar</url-pattern>
    <url-pattern>*.cab</url-pattern>
    <url-pattern>*.js</url-pattern>
    <url-pattern>*.htm</url-pattern>
    <url-pattern>/servicio/*</url-pattern>
    <url-pattern>/images/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>pfirma</role-name>
  </auth-constraint>
</security-constraint>

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

<security-role>
  <description>Rol de portal Web Port@firmas</description>
  <role-name>pfirma</role-name>
</security-role>
```

Con esto hemos declarado un tipo de autenticación BASICA, es importante que se haga a través de una capa segura en SSL para que los datos del usuario y la clave viajen de forma encriptada.

3.2 Autenticación de aplicaciones

Ya hemos visto como podemos restringir el acceso a los usuarios del portal web de Port@firmas, ahora vamos a ver el caso de pedir autenticación a las aplicaciones que atacan a los servicios web.

3.2.1 Restringiendo el acceso al servicio web de Port@firmas

El procedimiento es idéntico al caso anterior. Vamos a crear un nuevo rol tal y como se describe en el punto 3.1.1 en este mismo documento, en este caso lo vamos a llamar “pfirmaws”, y vamos a crear un usuario a su vez para poder probarlo.

Los siguientes cambios que deberemos hacer serán en el fichero web.xml de la aplicación, en el cual incluiremos las siguientes líneas.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Web Service Port@firmas</web-resource-name>
    <url-pattern>/services/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>pfirmaws</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
```

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

  <security-role>
    <description>Servicio Web Port@firmas</description>
    <role-name>pfirmaws</role-name>
  </security-role>
```

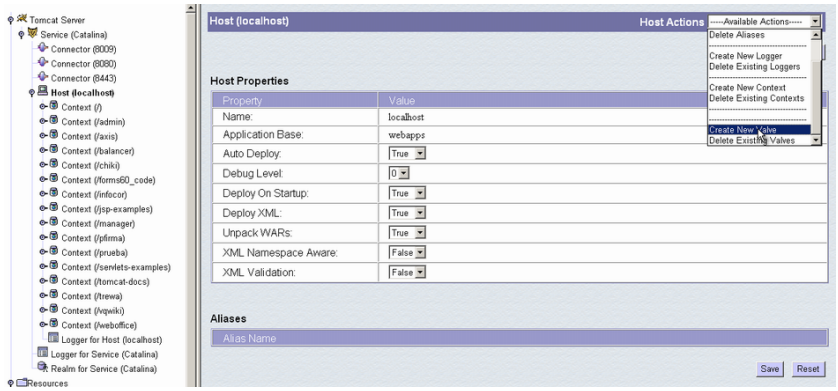
Si queremos tener habilitado ambas autenticaciones, usuario y aplicaciones, el código a añadir total en el web.xml sería:

```
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Portal Web Port@firmas</web-resource-name>
      <url-pattern>*.jsp</url-pattern>
      <url-pattern>*.jar</url-pattern>
      <url-pattern>*.cab</url-pattern>
      <url-pattern>*.js</url-pattern>
      <url-pattern>*.htm</url-pattern>
      <url-pattern>/servicio/*</url-pattern>
      <url-pattern>/images/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>pfirma</role-name>
    </auth-constraint>
  </security-constraint>

  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Web Service Port@firmas</web-resource-name>
      <url-pattern>/services/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>pfirmaws</role-name>
    </auth-constraint>
  </security-constraint>

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

  <security-role>
    <description>Rol de portal Web Port@firmas</description>
    <role-name>pfirma</role-name>
  </security-role>

  <security-role>
    <description>Servicio Web Port@firmas</description>
    <role-name>pfirmaws</role-name>

  </security-role>
```

3.3 Habilitando el SingleSignOn en Tomcat

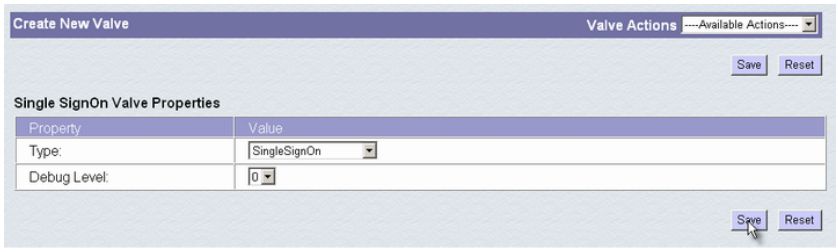
Para habilitar la posibilidad de que un usuario solo se tenga que autenticar una sola vez para varias aplicaciones deberemos activar una de las “válvulas” existentes en Tomcat.

Para ello entraremos de nuevo en la herramienta de administración de Tomcat para activar la válvula de SingleSignOn, pulsaremos en el árbol sobre “Host”, y en el “Host Actions” de la derecha de la pantalla pulsaremos “Create New Valve”.



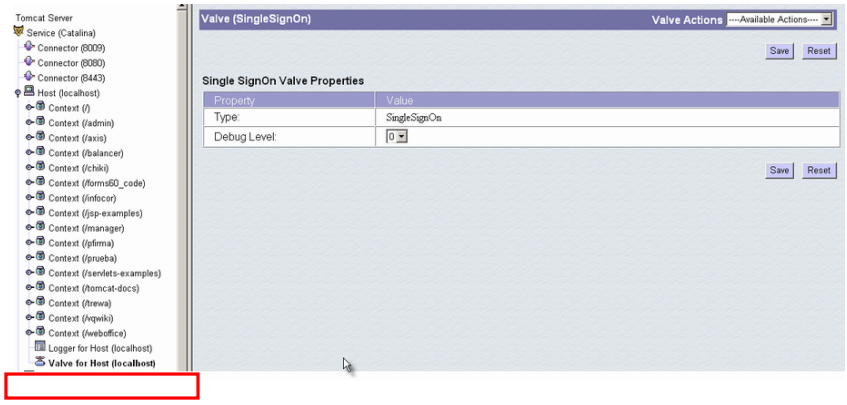
16. Creación de una válvula.

Escogemos una de tipo “SingleSignOn” y le damos al botón “save”.



17. Válvula SingleSignOn.

Le damos a “Commit Changes” para llevar a efecto los cambios y comprobamos que nos aparece ya la misma en el árbol del “Host”.



18. Arbol de objetos del Host.

Con esto habremos conseguido que un usuario que tiene varios roles para varias aplicaciones solo tendrá que autenticarse una sola vez, dejándole acceder a aquellas aplicaciones que le corresponda por los roles que posee.

3.4 Resolviendo JAAS desde clientes Web Service

Al introducir una autenticación a las aplicaciones que accedan a los servicios web de Port@firmas hemos incluido un nuevo nivel de seguridad para las aplicaciones clientes que atacan dichos servicios, por lo cual estas mismas tendrán que identificarse a la hora de invocarlos para que el sistema los acepte como válidos.

El proceso es sencillo, solo hay que incluir los datos de usuario y clave que hallamos definido para esa aplicación en la clase “adaptadora” (Stub) del cliente web service generado con Apache Axis.

```
PfServicioWSServiceLocator locator = new PfServicioWSServiceLocator();
servicio = locator.getPfServicioWS(new URL(urlPfirma));
org.apache.axis.client.Stub s = (Stub) servicio;
s.setUsername(httpUser);
s.setPassword(httpPassword);
```

Tras esto ya habremos configurado el cliente Web Service para que realice la autenticación HTTP con los datos proporcionados.

4 Configuraciones de seguridad Port@firmas

Una vez visto los mecanismos de seguridad que podemos definir sobre Port@firmas vamos a ver cual sería el conjunto de las mismas recomendables para los distintos entornos.

Port@firmas emplea la conexión como testigo de correcto acceso en la aplicación, de tal forma que si se intenta acceder a cualquier otro recurso sin haber pasado por el proceso de autenticación, este lo rechaza al ser nula la misma. Una vez el usuario cierra la aplicación esta se elimina junto al resto de objetos de sesión.

Por lo tanto hay que vigilar que no se pueda acceder masivamente para provocar un número elevado de conexiones.

En el siguiente cuadro se indica cuales son las configuraciones de seguridad recomendadas para cada tipo de entorno.

	PORTAL WEB SSL	PORTAL WEB JAAS	WEB SERVICE SSL	WEB SERVICE JAAS
Intranet aislada (<100 usuarios)	No	No	No	No
Intranet No Aislada (<100 usuarios)	Sí	No	Sí	No
Internet (solo Portal Web)	Sí	Sí	-	-
Internet	Sí	Sí	Sí	Sí

Esta tabla es solo orientativa dado que será el administrador del sistema el que decida cual nivel de seguridad imponer a su sistema.

5 Historia de versiones

VERSION	FECHA	AUTOR	DESCRIPCIÓN
1.1.1	04/04/2006	FJCV	Primera versión del documento.
1.2.0	23/06/2006	FJCV	Revisión, no hay cambios.