El sistema Java implementado

Capítulo 7

Descripción del sistema

Contenido

7.1 Introducción

En la tercera parte de esta memoria mostramos el sistema que se ha implementado usando como base código Java extraído de la iniciativa Shibboleth. Se trabajará en un entorno de pruebas que nos permitirá construir un sistema simplificado para aproximarnos al modelo de federación mostrado al principio de esta memoria. Los elementos que vamos a usar han aparecido ya en capítulos anteriores: un usuario que hace uso de un navegador web, un Proveedor de Identidad y unos recursos electrónicos situados en un Proveedor de Servicios, cuyo acceso queremos controlar. En nuestro

caso existirán dos archivos con extensión “htm”, uno de los cuales será de libre acceso, mientras que para acceder al otro será necesario proporcionar las credenciales adecuadas. En principio, el Proveedor de Identidad reúne las funcionalidades del agente de autentificación y del agente de credenciales que se presentaba en el capítulo inicial de esta memoria aunque, más adelante, se discutirá la manera de poder separar ambos agentes.

El funcionamiento del sistema es, a grandes rasgos, el siguiente:

- El usuario intenta acceder a uno de los dos archivos .htm que se encuentran en el Proveedor de Servicios.
- Si el archivo está protegido, se redirige la petición al Proveedor de Identidad para realizar un proceso de autentificación del usuario.
- El usuario retorna desde el Proveedor de Identidad con las credenciales adecuadas para acceder al recurso deseado.

En este capítulo realizaremos una descripción general del sistema. En los capítulos siguientes profundizaremos más en su estudio. En especial el capítulo 8 concretará muchos detalles que en éste sólo se perfilan.

7.2 Componentes del sistema

El sistema está programado íntegramente en Java, usando las librerías J2SE 5.0 como soporte. No se instala Apache, utilizamos únicamente Tomcat 5.5 como servidor web. El entorno de trabajo elegido fue Eclipse y el sistema operativo, Windows XP.

La estructura básica de la implementación se basa en la que apareció en el capítulo 5, dedicado a Shibboleth. Dicha estructura se muestra en la figura 21:

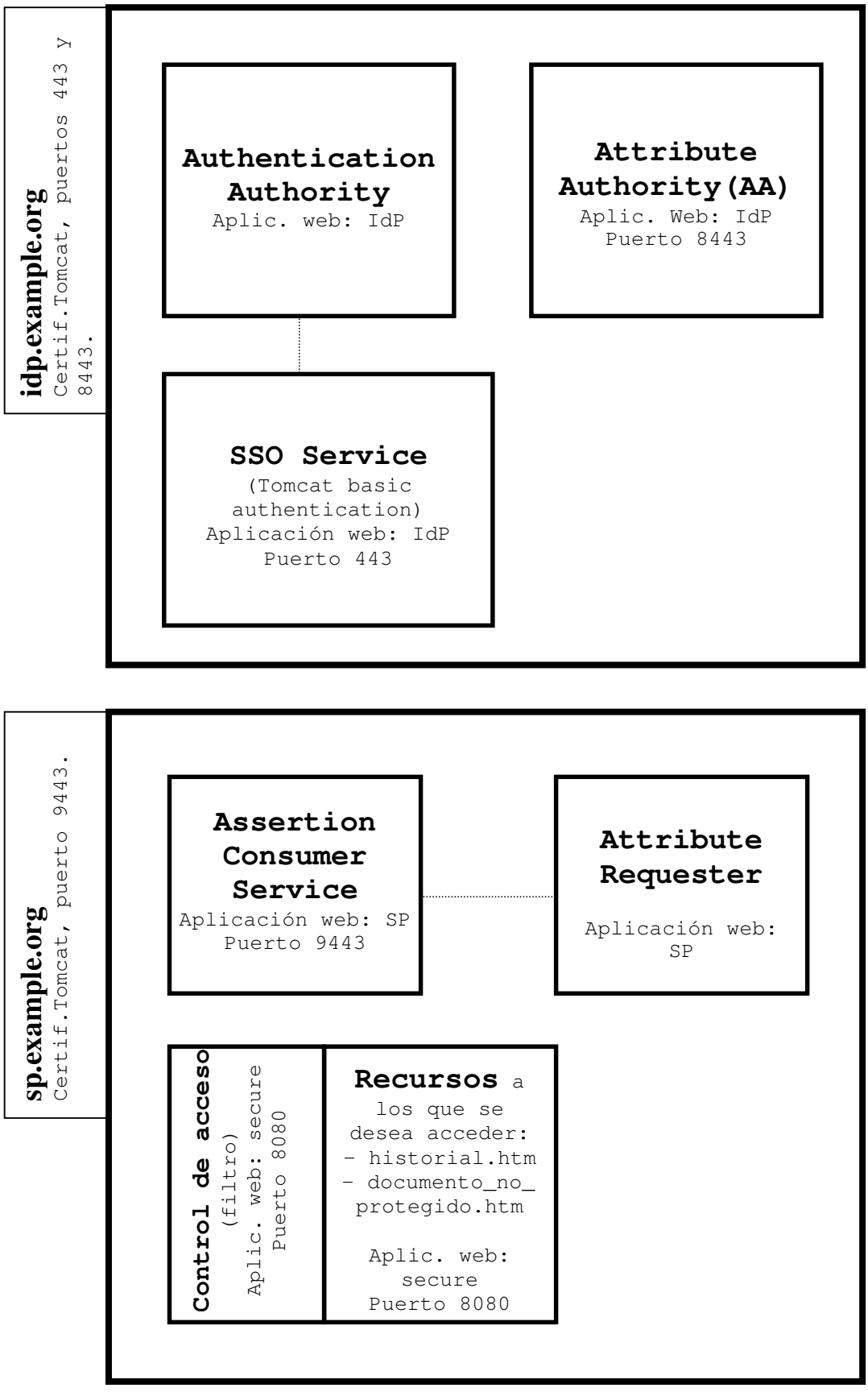

Figura 21:Elementos del sistema implementado

Disponemos de un Proveedor de Identidad o Identity Provider (IdP) y de un Proveedor de Servicios o Service Provider (SP). Comparando con el esquema de la federación presentado al principio de esta memoria, el Proveedor de Identidad aúna las funciones del agente de autenticación y del agente de credenciales. El Proveedor de Servicios desempeña el papel de uno de los sistemas genéricos que aparecían en nuestra federación modelo. En él residen los recursos que deseamos proteger.

En un sistema que estuviese en producción, Proveedor de Identidad y Proveedor de Servicios tendrían cada uno una dirección IP real. Nuestro caso, por ser un sistema en pruebas, es un tanto diferente. Una vez que Tomcat está activo, éste responderá a las peticiones del navegador dirigidas al bucle local o "localhost", cuya dirección es siempre 127.0.0.1. Los nombres de dominio asignados al Proveedor de Identidad y al de Servicios, idp.example.org y sp.example.org respectivamente, son ficticios, no correspondiéndose con ninguna dirección IP real. Podemos modificar el archivo de "hosts" de Windows y añadir dos líneas, una para el IdP y otra para el SP, para asignar a ambos nombres de dominio la dirección IP 127.0.0.1. De esta forma, siempre que escribamos en el navegador "http://sp.example.org" o "http://idp.example.org", estaremos realmente enviando una petición al bucle local, petición que será atendida por Tomcat. Así logramos simular localmente la existencia de dos servidores.

Existen tres bloques funcionales en el Proveedor de Servicios:

- Dos archivos HTM que representan los recursos a los que queremos acceder, junto con su correspondiente control de acceso.
- El Assertion Consumer Service, que se encarga de procesar las aserciones SAML de autentificación que le llegan desde el Proveedor de Identidad.
- El Attribute Requester, que construye y emite las peticiones de aserciones SAML de atributos, dirigidas al Proveedor de Identidad.

Otros tantos podemos encontrar en el Proveedor de Identidad:

- El Single Sign-On Service, encargado de autentificar al usuario.

- La Authentication Authority, que crea aserciones SAML de autentificación.
- La Attribute Authority, que se encarga de responder a peticiones de aserciones de atributos.

Asignaremos unos determinados puertos de acceso a algunos de estos bloques. Usaremos dichos puertos siempre que realicemos una llamada para ejecutarlos. La asignación de puertos es la siguiente:

- Recursos y control de acceso: puerto 8080.
- Assertion Consumer Service: puerto 9443.
- SSO Service: puerto 443.
- Attribute Authority: puerto 8443.

No asignamos ningún puerto ni a la Authentication Authority del Proveedor de Identidad ni al Attribute Requester del Proveedor de Servicios, ya que éstos no lo necesitan. Cuando veamos la descripción detallada del sistema en tiempo de ejecución, podremos darnos cuenta de que, cada vez que estos dos bloques entren en acción, lo harán por llamadas realizadas desde bloques de código Java situados dentro de su propio servidor.

El hecho de asignar puertos distintos para cada bloque nos permite gestionar mejor las características de seguridad del sistema. Esto es gracias a que Tomcat nos da la posibilidad de asociar un certificado de servidor distinto a cada puerto utilizado. En nuestro caso, asociaremos a los puertos 443 y 8443, un certificado de seguridad expedido a nombre del servidor “idp.example.org”, mientras que el puerto 9443 tendrá asociado otro certificado expedido a nombre de “sp.example.org”. Cuando un bloque se comunique con otro se usarán los certificados como identificación, como si realmente ambos se encontraran en servidores distintos. El puerto 8080 no necesita ningún certificado de seguridad, al no usarse para el intercambio de información SAML. La asignación de certificados a los puertos se realiza modificando convenientemente el archivo server.xml de Tomcat.

7.3 Distribución de los bloques funcionales

Los bloques que acabamos de describir se construyen utilizando servlets y filtros Java, incluidos en tres aplicaciones web que se despliegan en Tomcat. Dos de dichas aplicaciones, "secure" y "shibboleth-sp", conforman el Proveedor de Servicios, mientras que la restante,"shibboleth-idp", constituye el Proveedor de identidad. Presentamos a continuación sus características principales:

- **Aplicación web "secure"**

  Representa los recursos a acceder y la parte del sistema relacionada directamente con el control de acceso. En la figura 21 son los dos bloques con borde contiguo que aparecen en la parte inferior. Se accede a la aplicación a través del puerto 8080 de Tomcat. El archivo que el contenedor de aplicaciones despliega tiene por nombre "secure.war". Esta aplicación atenderá, por tanto, todas las peticiones del navegador que contengan

  `http://sp.example.org:8080/secure`

  En su ruta principal alberga dos archivos HTM. Uno de ellos, documento_no_protegido.htm, es de libre acceso, mientras que historial.htm requiere autentificación. Para establecer esta diferencia de requisitos de seguridad incluimos un filtro Java, el cual se encarga del control de acceso. Disponemos de dos medios, dentro del descriptor de despliegue de la aplicación web, para especificar de forma granular qué archivos queremos que el filtro controle y cuáles queremos que sean libremente accesibles.

  Por una parte, usando la directiva de Tomcat `<filter-mapping>`, configuramos la aplicación para que el filtro se ejecute sólo en caso de que intentemos acceder a archivos con extensión "htm".

Por otra, en el descriptor de despliegue incluimos un parámetro de inicialización, llamado "requireId", y que le indica al filtro qué sólo debe controlar el acceso a archivos cuyo nombre incluya la palabra "historial".

Los parámetros de inicialización que aparecen en el descriptor de despliegue de esta aplicación son:

- **providerId**
  Se usa como identificador único del Proveedor de Servicios. Su valor es
  `https://sp.example.org/shibboleth`

- **wayfURL**
  Es la dirección del Servidor "Where are you from?" (WAYF), utilizado para obtener del usuario el nombre del Proveedor de Identidad en el que desea autentificarse (véase capítulo 5 para mayor información). En nuestro caso tenemos tan sólo un Proveedor de Identidad, por lo que el valor de este parámetro será la propia dirección de dicho Proveedor de Identidad único, al que serán redirigidos todos los usuarios que pretendan acceder a los recursos del Proveedor de Servicios. En concreto, le asignaremos como valor la dirección del Servicio Single-On del Proveedor de Identidad, es decir

  `https://idp.example.org:443/shibboleth-idp/SSO`

- **shireURL**
  Indica la localización del Assertion Consumer Service en el Proveedor de Servicios. En nuestro caso,

`https://sp.example.org:9443/shibboleth-sp/shibboleth.sso/SAML/POST`

  Podemos observar que se trata de una subruta dentro de la aplicación web "shibboleth-sp", de la que hablamos más abajo. El valor de este parámetro lo adjuntaremos en la petición de autentificación que el Proveedor de Servicios le enviará al Proveedor de Identidad, cuando un

usuario intente acceder a un recurso protegido. De esta forma, el IdP sabrá a dónde tiene que mandar su respuesta, es decir, su aserción de autentificación.

- **requireId**
  Se trata de un parámetro cuyo valor será leído por el filtro de control de acceso y que le indicará a éste si debe realmente intervenir en el intento de acceso que se está produciendo o dejar que se produzca libremente. En concreto, se configura para que el filtro actúe en caso de que se intente acceder a archivos que incluyan la palabra "historial".

En esta aplicación web no se despliega ningún servlet.

- **Aplicación web "shibboleth-sp"**
  Esta aplicación contiene el resto del Service Provider, es decir, el Assertion Consumer Service y el Attribute Requester. Para desplegar esta aplicación la habremos empaquetado previamente en el archivo "shibboleth-sp.war", lo que implica que la aplicación responderá a llamadas del tipo

```
https://sp.example.org:9443/shibboleth-sp
```

La aplicación incluye un servlet, llamado "AssertionConsumer" y que se mapea a la clase Java principal del bloque de código del mismo nombre, el Assertion Consumer Service. La ruta que este servlet manejará se define como

```
/Shibboleth.sso/SAML/POST
```

Esto implica que una petición como la siguiente:

```
https://sp.example.org:9443/shibboleth-sp/Shibboleth.sso/SAML/POST
```

pondría en marcha la "maquinaria" Java del Assertion Consumer Service (obsérvese la correspondencia con el parámetro "shireURL" que presentamos más arriba).

El Attribute Requester entrará en acción siempre a través de llamadas internas desde el Assertion Consumer Service y, por tanto, no estará asociado a ningún puerto en particular, ni definiremos ningún servlet que esté mapeado a ninguna de sus clases Java.

En el descriptor de despliegue de esta aplicación web aparecen otros servlets y filtros para tareas administrativas (para producir logs, por ejemplo), que nosotros no utilizaremos.

- **Aplicación web "shibboleth-idp"**

  Esta aplicación la constituyen los tres bloques del Identity Provider de la figura 21. Tomcat se encarga de desplegar el archivo "shibboleth-idp.war" para poner esta aplicación a disposición del navegador. Se configura un servlet, llamado "IdP", al cual se asocian las dos subrutas siguientes:

  - **/SSO**

    Las llamadas a esta ruta se asocian al puerto 443 (puerto estándar SSL) y sirven para activar el Servicio Single Sign-On del Proveedor de Identidad. Además, se incluyen líneas en el descriptor de despliegue para que se active la Autenticación Básica de Tomcat, cada vez que se reciba una llamada. Esto quiere decir que, si escribimos en el navegador la línea siguiente:

    ```
    https://idp.example.org:443/shibboleth-idp/SSO
    ```

    obtendremos, en primer lugar, la máscara de autentificación de Tomcat, donde introduciremos un nombre de usuario y una contraseña, y, si la información que proporcionamos es correcta, se ejecutará el código Java del Servicio Single Sign-On.

- **/AA**
  Esta subruta se asocia al puerto 8443 y proporciona acceso a la Attribute Authority del Proveedor de Identidad.

Hay que puntualizar, para ser rigurosos, que aparece otra subruta más ("/Artifact") en el descriptor de despliegue, que nosotros no usaremos en ningún momento.

La aplicación web "shibboleth-idp" no despliega ningún filtro de servlet

7.4 Interacción usuario – sistema

A continuación describiremos cómo se produce la interacción del sistema implementado con el usuario, el cual utiliza un navegador web. La figura 22 ilustra dicha interacción:

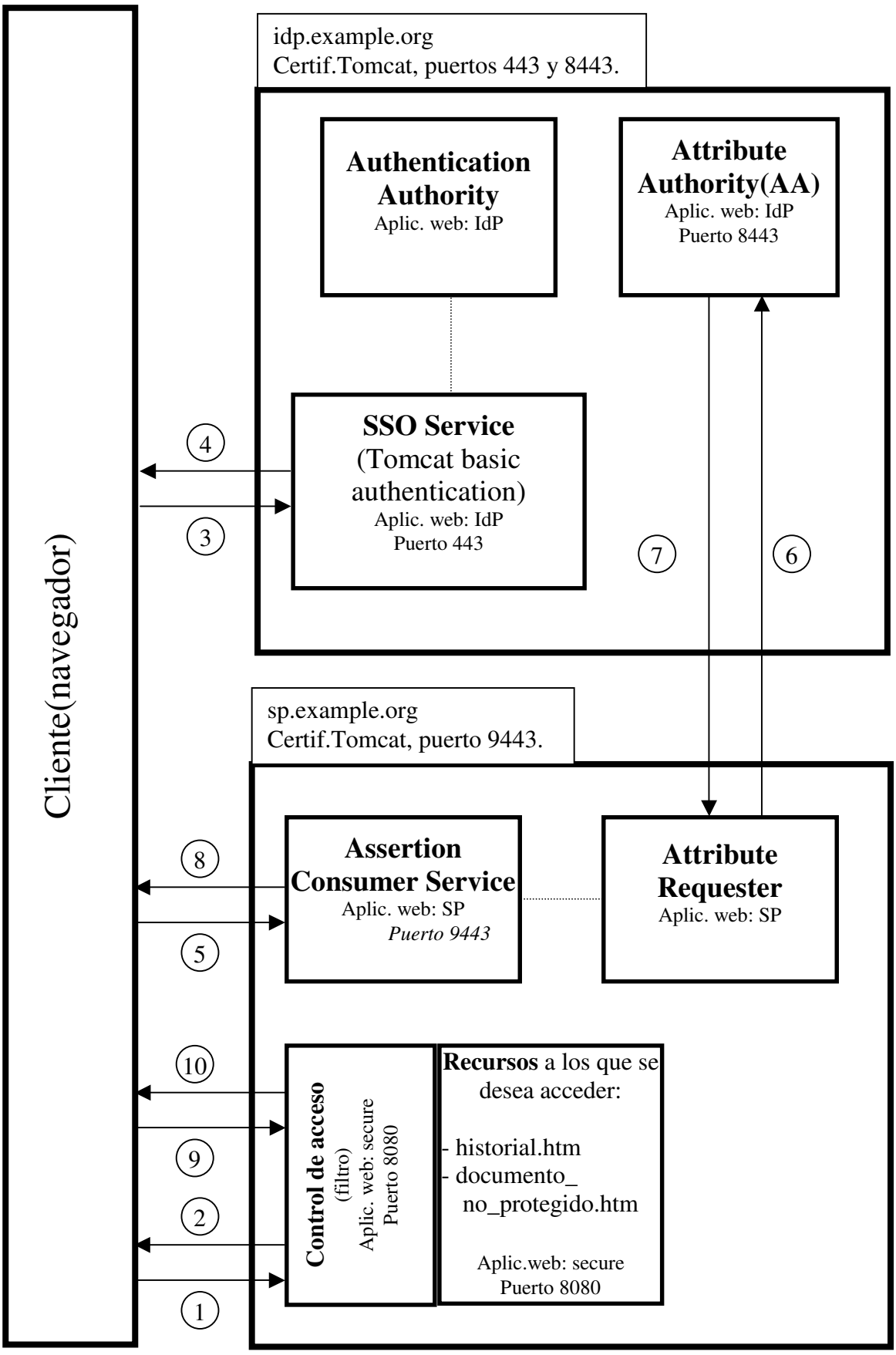

Figura 22:Interacción usuario – sistema

Resumen de la interacción

Los pasos que aparecen en la figura 22 pueden resumirse de la siguiente manera:

1. El usuario solicita un archivo del Proveedor de Servicios utilizando su navegador web.

2. Si el acceso al documento requiere autentificación, el filtro de acceso redirige al navegador al Proveedor de Identidad.

3. El navegador, obedeciendo la orden del filtro, envía la petición de autentificación al IdP, en concreto al Servicio Single Sign-On.

4. El Servicio Single Sign-On responde con una aserción SAML, obtenida de la Authentication Authority del Proveedor de Identidad.

5. El navegador envía la aserción, incluida en un formulario HTML, al Assertion Consumer Service del Proveedor de Servicios.

6. El Assertion Consumer Service crea una sesión para el usuario. El Attribute Requester solicita los atributos del usuario a la Attribute Authority del Service Provider.

7. La Attribute Authority responde con una aserción SAML de atributos.

8. El Assertion Consumer redirige el navegador al archivo deseado por el usuario.

9. El navegador efectúa la redirección y solicita, al igual que en el paso 1, el archivo al Proveedor de Servicios.

10. Al existir una sesión de seguridad para el usuario, el archivo es entregado a su navegador.

Análisis detallado de la interacción

A continuación desarrollamos cada uno de los pasos enumerados anteriormente:

1. El usuario intenta acceder a un archivo alojado en el Proveedor de Servicios, concretamente dentro de la aplicación web "secure". Para ello escribe la dirección correspondiente en su navegador web tal como aparece en la figura 23:

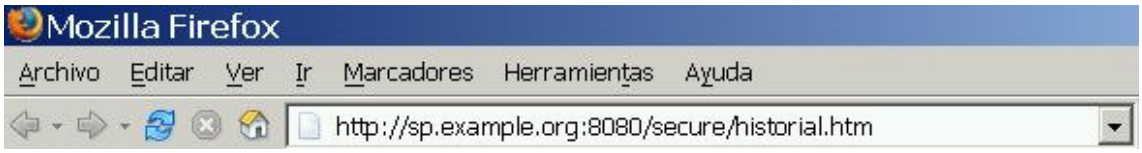

Figura 23: Intento de acceso al archivo protegido

- Tomcat pasa el control a "secure".

- Al tener el archivo extensión "htm", Tomcat pone en marcha la ejecución del filtro Java de control de acceso.

- Dentro de su código, el filtro lee el valor de requireId (uno de los parámetros de inicialización de la aplicación web "secure") y comprueba si el nombre del archivo al que desea acceder el usuario concuerda con lo que le indica dicho parámetro. Al incluir el nombre del archivo la palabra "historial", estamos en la situación en que se requiere control de acceso. Si no fuera así, y hubiéramos solicitado, por ejemplo, el archivo **documento_no_protegido.htm**, el filtro terminaría su ejecución y devolvería el control a Tomcat, el cual entregaría el archivo al navegador del usuario, finalizando la interacción con el mismo. En ese caso, obtendríamos la pantalla mostrada en la figura 24 y la secuencia de pasos terminaría aquí:

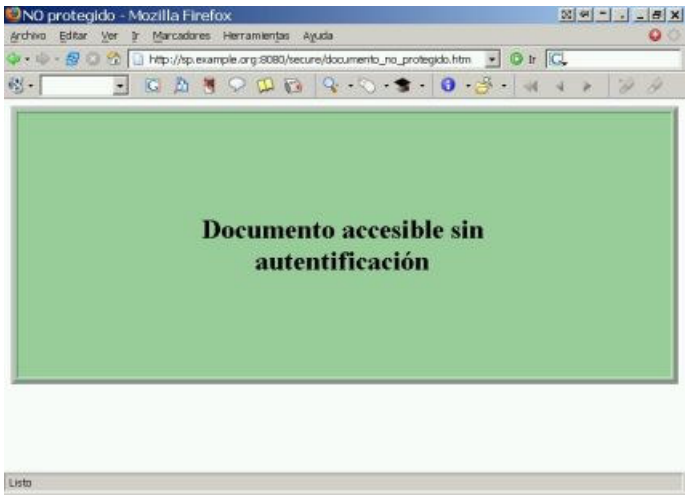

Figura 24: El archivo documento_no_protegido.htm

- Al requerir el archivo deseado medidas de control, el filtro continúa su ejecución. La siguiente comprobación que realiza es si el usuario dispone ya de una sesión de seguridad en el Proveedor de Servicios. Si es así, la interacción saltaría al paso 10. En caso contrario, seguiríamos por el paso 2.

2. El filtro Java ya ha confirmado que el archivo deseado cumple las condiciones para requerir control de acceso y que el usuario no dispone de una sesión de seguridad en el Proveedor de Servicios. Se pone en marcha, por lo tanto, el proceso de autentificación y autorización del usuario. El primer paso será conseguir del Proveedor de Identidad una aserción SAML de autentificación. Para ello, el filtro lee el valor del parámetro de inicialización de la aplicación "secure" denominado "wayfURL" y redirige el navegador a la dirección dada por dicho parámetro. Como se comentó en apartados anteriores, nuestro sistema no utiliza ningún servidor WAYF, así que el valor del parámetro wayfURL es directamente la dirección del Servicio Single Sign-On del Proveedor de Identidad.

3. El navegador envía la petición de autentificación SAML al servicio Single Sign-On del Identity Provider. La dirección completa a la que se dirige, que presentamos en cuatro líneas de texto para mayor claridad, es la que aparece en la figura 25:

```
      https://idp.example.org:443/shibboleth-idp/SSO?
shire=https://sp.example.org:9443/shibboleth-sp/Shibboleth.sso/SAML/POST&
      target=http://sp.example.org:8080/secure/historial.htm&
      providerId=https://sp.example.org/shibboleth
```

Figura 25: Petición de autenticación

Se puede observar que hemos incluido tres parámetros en la llamada:

- La dirección del Assertion Consumer Service del SP, que se toma del parámetro de inicialización “shireURL”, de la aplicación “secure”:

shire=https://sp.example.org:9443/shibboleth-sp/Shibboleth.sso/SAML/POST

- La dirección del recurso deseado:

target=http://sp.example.org:8080/secure/historial.htm

- La identificación única del proveedor de servicio, también tomada de los datos de “secure”:

providerId=https://sp.example.org/shibboleth

Al haber solicitado una conexión segura al servidor idp.example.org (obsérvese que hemos usado “https://” en vez de “http://”), éste intentará identificarse ante nosotros, enviándonos su certificado de seguridad. En nuestro sistema usamos certificados de prueba, que obtenemos gratuitamente al ser generados por la propia aplicación Java, en vez de adquirirlos comercialmente a alguna autoridad certificadora. El navegador, por tanto, no reconoce (al menos, en primera instancia) el certificado que nos envía idp.example.org porque la autoridad certificadora no consta en su base de datos. Como consecuencia obtenemos el cuadro de advertencia de la figura 26.

Figura 26:El navegador no reconoce la autoridad certificadora

Podemos elegir aceptar temporalmente el certificado, con lo que, en la próxima interacción con el sistema nos encontraremos de nuevo con este mensaje o bien aceptarlo permanentemente, con lo cual el navegador incluirá el certificado en su base de datos y no volverá a presentarnos esta advertencia (si elegimos la tercera opción, "no aceptar este certificado", la ejecución se detendrá).

Una vez que hemos elegido aceptar el certificado (sea temporal o permanentemente), entrará en acción la aplicación web "shibboleth-idp". En concreto, estamos accediendo a la subruta "/SSO", la cual está configurada en el descriptor de despliegue para requerir la **Autenticación Básica de Tomcat**. Por tanto:

- Tomcat fuerza al navegador a presentar la máscara de autentificación, para introducir un nombre de usuario y una contraseña (figura 27).

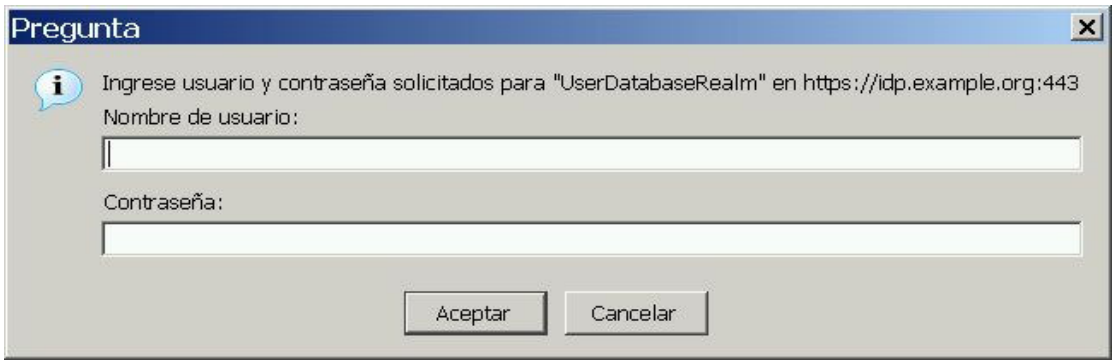

Figura 27: Máscara de Autenticación Básica de Tomcat

- Introducimos "tomcat" como nombre de usuario y la misma palabra como contraseña. Los datos introducidos se comparan con los existentes en el archivo de configuración de la Autenticación Básica de Tomcat. Al coincidir, podemos seguir adelante.
- El Servicio Single Sign-On obtiene una afirmación SAML de autentificación de la Authentication Authority.

4. El Servicio Single Sign-On responde a la petición del navegador (la que aparecía al inicio del punto 3) con un documento HTML que contiene un formulario con campos ocultos. El valor de uno de esos campos es la respuesta SAML ("SAML Response") que incluye la afirmación de autentificación generada por la Authentication Authority. En la figura 28 se muestra el código del formulario HTML. Por razones de claridad, no se muestra explícitamente el valor de la Respuesta SAML.

```
<form id="shibboleth"  action =
  "https://sp.example.org:9443/shibboleth-
  sp/Shibboleth.sso/SAML/POST" method="post">

  <input type="hidden" name="TARGET" value=
  "http://sp.example.org:8080/secure/historial.htm" />

  <input type="hidden" name="SAMLResponse" value="response" />
  <noscript>
    <input type="submit" value="Continue" />
  </noscript>
</form>
```

Figura 28: Formulario HTML con Respuesta SAML

Puede observarse que se devuelve la dirección del recurso deseado (campo **target** del formulario) y que el formulario tiene como dirección de destino la del Assertion Consumer del Proveedor de Servicios (véase el valor de **action**).

La respuesta del Servicio Single Sign-On incluye también una línea de código Javascript que fuerza al navegador a reenviar automáticamente el formulario HTML a la dirección contenida en "action":

```
<body onload="document.forms[0].submit()">
```

La línea que aparece dentro del elemento `<noscript>` en la figura 28 permite que la redirección pueda tener lugar incluso cuando el navegador del que dispone el usuario no es compatible con Javascript. En ese caso, será necesario pulsar sobre el botón "Continue".

Un mensaje informando sobre la redirección es mostrado por el navegador durante la ejecución de la misma (ver figura 29).

Figura 29: Redirección desde el Servicio Single Sign-On

5. El navegador envía el formulario del Single Sign-On al Assertion Consumer, usando POST como método del protocolo HTTP. Esto será automático o tras la pulsación, por parte del usuario, del botón "Continue" de dicho formulario (ver paso anterior).

En concreto, la dirección a la que nos dirigimos, y que ya aparecía en el formulario mostrado anteriormente, es la siguiente:

```
https://sp.example.org:9443/shibboleth-sp/Shibboleth.sso/SAML/POST
```

Nos encontramos entonces con una situación análoga a la descrita en el paso 3: el Proveedor de Servicios, que actúa como servidor en esta transacción, se identifica mediante su certificado pero al cliente, es decir, al navegador, dicho certificado no le resulta confiable (una descripción más detallada puede encontrarse en el paso 3). Suponemos que aceptamos el certificado y continuamos.

Se pone en marcha la aplicación web "shibboleth-sp" y , en concreto el Assertion Consumer, al estar mapeado a la subruta que hemos llamado.

El Assertion Consumer analiza el formulario HTML que recibe, valida la firma de la respuesta SAML que en él se incluye, crea una sesión y pasa el control al Attribute Requester (esto es sólo una llamada Java que no implica al navegador).

6. El Attribute Requester envía una SAML Request a la Attribute Authority del Identity Provider.

7. El control pasa entonces de nuevo a la aplicación "shibboleth-idp", en concreto a la Attribute Authority. Ésta procesa la petición y responde a ella con las credenciales del usuario. La **Attribute Release Policy (ARP)** del Proveedor de Identidad determina qué atributos se entregan en la respuesta y cuáles se consideran confidenciales y no serán revelados. El Proveedor de Identidad tiene configurado, asimismo, cómo debe gestionar la identidad del usuario, lo cual determina el elemento empleado como identificador del sujeto en la aserción SAML que se incluye en la respuesta. Seguidamente, el Attribute Requester recibe las credenciales y aplica la **Attribute Acceptance Policy (AAP)**, es decir, descarta aquellos atributos que el Proveedor de Servicios no desee, por no ser significativos en relación con la decisión de acceso o por cualquier otra razón. Hay más información sobre gestión de identidad y de atributos en los apartados 8.6, 8.7 y 10.4.

8. El Attribute Requester devuelve el control al Assertion Consumer, el cual redirige el navegador a la dirección del recurso a acceder.

9. El navegador solicita, al igual que en el punto 1, el acceso al recurso deseado, añadiendo a la llamada, en este caso, la identificación de la sesión iniciada:

```
http://sp.example.org:8080/secure/historial.htm?ShibbolethSessionId=3FDPZKGEaQ097gQZ
```

10. Al existir ya un contexto de seguridad el documento solicitado es enviado al navegador (figura 30).

Figura 30: Recurso solicitado