Conclusión y líneas de continuación

La federación de sistemas es un tema apasionante y es posible que represente el futuro inmediato de las redes de comunicación computerizadas. Puede apostarse a que los avances que en este campo se consigan afectarán de forma rápida a nuestra vida cotidiana, debido a la importante presencia que Internet posee ya en la sociedad actual.

En esta memoria se han plasmado los objetivos que nos marcamos al inicio de la misma:

- Se ha propuesto un modelo de federación que cumple las condiciones para convertirse en un sistema real.
- Se ha analizado SAML, un nuevo estándar para el intercambio de información de seguridad, así como una implementación práctica de dicho estándar, Shibboleth.
- Se han presentado otras iniciativas para la gestión de identidad federada.
- Basándonos en la iniciativa Shibboleth, se ha implementado un sistema Java que representa una aproximación simplificada al modelo presentado inicialmente. A excepción del sistema operativo Windows, todos los demás elementos utilizados son gratuitos y de código abierto.
- Se han analizado las mejoras que pueden realizarse tanto en el modelo inicial como en el sistema implementado para seguir avanzando hacia una federación real.

Hemos podido comprobar que implementar una federación de sistemas es un objetivo muy ambicioso. El diseño de la arquitectura del sistema presenta innumerables posibilidades y variantes que exigen responder a gran cantidad de interrogantes, y tomar decisiones que pueden repercutir de forma decisiva en la idoneidad de la implementación práctica. De la misma forma, la programación del sistema diseñado implica emplear código de considerable calado. Finalmente, poner en producción el sistema final y configurarlo para un funcionamiento óptimo tampoco es una tarea trivial.

Este Proyecto pretende tan sólo ser un paso más en el camino que lleve a dicha implementación final. Esperamos que esta memoria pueda servir como punto de partida para otros alumnos y/o investigadores en este mismo ámbito. Con ese propósito se tratan seguidamente las posibles líneas de continuación del trabajo realizado en este Proyecto.

El sistema que se ha implementado utilizando los bloques de código de Shibboleth constituye la estructura básica de una federación de servidores. Sin embargo, existen aún características del mismo que no reflejan por completo el modelo inicial de federación que se propuso, mientras que otras, al tratarse de un sistema en pruebas, no son adecuadas para un sistema real en producción.

Por estos motivos, aún restan una serie de mejoras en el camino hacia una implementación realmente operativa, la mayoría de las cuales requieren un análisis en profundidad previo a su realización. A continuación se detallan las más importantes:

- En el sistema actual, todo el código se ha instalado en un mismo ordenador, simulando la existencia de dos servidores mediante la adecuada configuración del archivo de "hosts" de Windows y la asignación de puertos de acceso distintos a los diferentes bloques que componen cada sistema. En un sistema real habría que instalar el Service Provider y el Identity Provider en **servidores separados**, tanto el código Java como los archivos de configuración. La solución más sencilla, como ya se explicó en 10.3, consistiría en instalar todo el código Java allí donde se necesite alguno de sus bloques, desactivando aquellos que no vayan a usarse.
- Asignar **direcciones IP reales** a los dos servidores empleados, sustituyendo así los nombres ficticios sp.example.org e idp.example.org que, en nuestro caso, equivalían a la dirección del "localhost", 127.0.0.1.
- El sistema se ha implementado sobre Windows XP. En producción sería mejor utilizar un **sistema operativo** más adecuado para servidores, como, por ejemplo, Linux o Windows Server. El primero cuenta con la ventaja de ser gratuito y de código abierto, características que lo hacen idóneo para la investigación universitaria.

- Instalar **Apache** para que funcione en conjunción con Tomcat. Para ello sería necesario también instalar un conector JK. Son válidos tanto mod_jk como mod_jk2.
- Utilizar certificados SSL firmados por alguna **autoridad reconocida** por todos los miembros de la federación. No tiene porqué tratarse de certificados adquiridos comercialmente, basta con que todos los navegadores implicados en las transacciones incluyan a dicha autoridad de certificación en su base de datos. Cada servidor miembro de la federación debería contar con su propio certificado asociado.
- El Servicio Single Sign-On emplea la **Autenticación Básica de Tomcat** para identificar a los usuarios. En un sistema real sería necesario emplear otro método más seguro. Una posibilidad sería definir un filtro de servlet asociado a la ruta del Single Sign-On Service. Este filtro se programaría para solicitar al usuario su identidad y contraseña federadas y contrastarlas con la tabla (ver punto siguiente) que posee el agente de autenticación.
- Es necesario construir la **tabla de identidades y contraseñas federadas** que debe contener el agente de autenticación. En nuestro caso, el archivo de usuarios de la Autenticación Básica de Tomcat cumple este papel, pero si sustituimos la Autenticación Básica por otro sistema más seguro para realizar la labor del Servicio Single Sign-On (tal como recomendábamos en el párrafo anterior), entonces necesitaremos sustituir también la forma en que almacenamos los usuarios y sus contraseñas. Esto puede realizarse utilizando, por ejemplo, una base de datos MySQL. En caso de que no se requiera procesado de atributos y no sea necesaria, por tanto, la presencia del agente de credenciales, cada entrada de la tabla contendría también la dirección de la Attribute Authority del sistema de origen del usuario.
- Implementar el **agente de credenciales**. En primer lugar, habría que estudiar cómo sería posible integrarlo, de forma que las llamadas del Attribute Requester pasen por él y no vayan dirigidas directamente a la Attribute Authority, como ocurre en la implementación actual. El agente debería poseer la tabla con las identidades federadas y sus correspondientes sistemas de procedencia. Las respuestas de la Attribute Authority también deberían ser filtradas por el agente,

que realizaría el procesado y adaptación de los atributos. Para la integración del agente se propone estudiar la misma opción que se sugirió antes para otras cuestiones: incluir todo el código Java dentro del agente o al menos los bloques correspondientes al Proveedor de Identidad original de Shibboleth. Contaríamos así con un bloque de código que aceptaría peticiones de atributos desde el Service Provider (Attribute Authority) y otro cuya función sería realizar solicitudes de los mismos al servidor de procedencia del usuario (Attribute Requester). La implementación del agente de credenciales no es trivial, siendo la adaptación de atributos entre diferentes sistemas lo que presenta una mayor dificultad conceptual.

- Configurar el agente de autenticación para que, una vez que haya autenticado satisfactoriamente al usuario, use la **identidad federada como identificador**, en vez de un handle. Configurar así el agente de autenticación implica que todos los bloques que participen en la transacción usarían también dicho identificador para el usuario.

- Construir una **fuente real de atributos**, ya sea una base de datos, un directorio LDAP, etc. y configurar el Identity Provider para utilizar dicha fuente, en vez del sistema automático de respuesta que empleamos ahora.

- Definir **más recursos** a proteger. Usando una aplicación web por recurso (ahora sólo una, la aplicación "secure"), podríamos establecer condiciones de acceso diferentes para cada uno de ellos. En este caso, "recurso" significaría una carpeta (la que indica la ruta de su aplicación web) y sus correspondientes subcarpetas.