# Consigna

## Justificación del caso

Consigna es una adaptación realizada por la Universidad del País Vasco a partir de Quixplorer para el intercambio de ficheros de forma sencilla. No requiere (ni ofrece la opción) autenticación al usuario. Cualquiera puede subir ficheros, con la restricción de que si esto se hace desde una dirección IP no incluida en un fichero de configuración (que típicamente son las direcciones de la institución) sólo se podrán descargar desde direcciones pertenecientes a los rangos declarados en ese fichero.

El sentido de ese control es que el sistema pueda servir para intercambiar ficheros no sólo dentro de la institución, sino permitir también que usuarios externos depositen ficheros para ser descargados por usuarios internos.

Esta forma de control es muy efectiva precisamente por su sencillez, el interfaz es tan simple que este servicio es muy utilizado y agradecido por los usuarios. El problema es que se dan casos tan paradójicos como que un usuario de nuestra institución puede subir un fichero y ni él mismo puede bajarlo luego. Por otro lado, en estos tiempos en que la gente dispone habitualmente de accesos de banda ancha en casa y los utiliza para trabajar, el hecho de que los ficheros que suba no puedan ser descargados por otro usuario desde su casa no parece muy razonable.

Una posible solución es que se solicite de forma voluntaria que el usuario se autentique al sistema, y que en ese caso los ficheros que suba puedan ser descargados desde cualquier lugar. Si se produce un abuso de cualquier tipo, será posible rastrear el problema hasta la persona que subió el fichero.

Al tratarse de un servicio muy usado para el intercambio de información entre personas que trabajan en temas afines (típicamente PDI de universidades), parece un buen candidato para integrarlo en una federación de identidad. De esta forma, no sólo proporcionamos las ventajas del acceso autentificado a nuestros usuarios, sino a cualquier otro cuya institución esté integrada en una federación.

Podemos unir a eso el hecho de que el código fuente de Consigna no es demasiado complejo y está bien estructurado, lo que permite su fácil modificación y de esta forma la experimentación con tecnologías de autenticación, SSO y federación en un servicio real.

## Implementación

Al tratarse de una prueba de concepto, vamos a probar distintas formas de autenticación y diferentes 'interfaces', esto es, cómo se le pide al usuario que se autentifique, si se dan facilidades a los usuarios locales (no les salga una lista de instituciones cuando casi todos los usuarios van a ser locales), etc. También debemos experimentar con las opciones de logout.

Inicialmente vamos a hacer las pruebas con PAPI y Shibboleth, pero podemos contemplar integrar CAS, Sun Access Manager u otros. Algunos de estos sistemas sirven como SSO, otros como sistema de federación y otros pueden desempeñar ambas funciones.

En primer lugar, al tratarse de un servicio donde la autenticación es opcional no podemos controlar el acceso al mismo de la forma habitual, exigiendo ésta para poder acceder a la aplicación. Por tanto, es la propia aplicación la que debe iniciar el proceso de

establecimiento de sesión en el sistema de autenticación en caso de que el usuario decida hacerlo. En el caso de PAPI, esto se puede hacer usando el phpPoA. Para Shibboleth usaremos las Lazy Sessions.

## Servidor Web

El phpPoA de PAPI no necesita configurar nada en el servidor Web, sólo invocar a las funciones del phpPoA desde la aplicación en el momento adecuado. Su implementación sólo necesita un GPoA externo. Para usar las Lazy Sessions de Shibboleth, aparte de que la aplicación haga algo en el momento adecuado, sí necesitamos que ésta esté bajo el control de un SP, es decir, el servidor Web se debe configurar para que ésta sea controlada por el módulo mod_shib, sin exigir el establecimiento previo de sesión:

```
<Location /consigna>
   ~AuthType shibboleth
   ~ShibRequireSession Off
   require shibboleth
</Location>
```

Esa diferencia se debe fundamentalmente a que el protocolo PAPI es bastante más sencillo y puede ser implementado en PHP (caso del phpPoA) aunque no sea completamente, por lo cual necesita un GPoA externo.

## Interfaz

La aplicación debe funcionar normalmente, el interfaz de usuario no cambiará por el hecho de haberse autenticado o no. Este hecho influirá en que al subir ficheros, en la base de datos quedará registrado que el usuario estaba autenticado, a efectos de tomar decisiones en el momento de la descarga. El único cambio necesario en el interfaz es que hay que presentarle la opción de autenticarse, y una vez hecho esto debe aparecer siempre el nombre de usuario así como la opción de logout.

El mostrar en las páginas el nombre de usuario y un enlace de logout es trivial, se pueden poner en cualquier sitio.

Para proporcionar autenticación, no se puede presentar unos campos de usuario/clave, pues estos deberá introducirlos en el IdP que le corresponda. Tenemos aquí varias opciones de autenticación:

- Contra el SSO local (PAPI, CAS, Access Manager, ...)
- Contra el IdP de Shibboleth local
- Contra un servicio WAYF que permite al usuario seleccionar su institución de origen.

Podemos proporcionar las tres opciones, pero los usuarios externos sólo pueden usar la tercera. Los locales podrían usar las tres, pero podemos hacer las siguientes consideraciones:

- Habría que proporcionar varios enlaces, lo cual es confuso para el usuario.
- Si el SSO local protege el IdP local, parece más razonable usar la segunda opción que la primera: en ambos casos van a llegar al SSO pero la aplicación sólo debe comunicarse con un IdP, no con el SSO directamente, lo cual simplifica las modificaciones que hay que realizar a ésta. **Ésta es una de las grandes ventajas de usar un sistema de SSO en la institución, y utilizarlo para proteger el IdP**.

Por tanto la mejor opción, por uniformidad y simplicidad parece la tercera. Pero tiene un inconveniente: si partimos de la suposición de que la mayoría de los usuarios van a ser locales, obligarles a seleccionar continuamente su propia institución de una lista es engorroso para ellos. Dos opciones pueden ser:

- Disponer dos enlaces llamados 'Usuario local' y 'Usuario remoto', que dirigen al IdP local y al WAYF, respectivamente.
- Proporcionar un único enlace para autenticación que dirige al servicio WAYF, pero en éste se presente preseleccionada la opción más probable, que puede ser o siempre la institución local o que según la dirección IP del cliente se intente averiguar la institución del usuario (en caso de duda, preseleccionar la institución local).

Tanto la autenticación como el WAYF puede ser una pequeña incomodidad para el usuario. Sin embargo en un entorno donde se use ampliamente el SSO puede que sólo deba introducir su clave una vez, la primera vez que accede a cualquier servicio, y en el WAYF se puede hacer que recuerde la selección de la institución origen, de forma que otros días no tenga que buscarla en la lista.

## Modificaciones a la aplicación

La aplicación debe comprobar al inicio (en el caso de Consigna, esto es sencillo porque el fichero index.php siempre se invoca para todo) si el usuario está autenticado o no. En las aplicaciones normales esto se hace mediante el uso de sesiones. Consigna no las usa, no las necesita. En nuestro caso, ya usemos PAPI, Shibboleth o lo que sea, podemos comprobar si se presentan las cookies o cabeceras que estos sistemas proporcionan, y si son válidas. En este caso, Shibboleth nos lo pone más sencillo: nos basta con comprobar que se nos presentan ciertas cabeceras, sin tener que comprobar su validez, porque no olvidemos que el mod*shib controla el acceso a nuestra aplicación y él se encarga de que nos lleguen correctamente. En el caso de PAPI, tendríamos que comprobar que está presente la cookie Lcook y también que es válida, lo cual esta fuera de nuestras posibilidades. Podemos poner una llamada a check*Access() al principio, pero esto provocaría una redirección al AS en caso de no tener sesión, lo cual no nos interesa porque la autenticación es voluntaria.

Por tanto si usamos PAPI, no podemos averiguar de forma sencilla si nos hemos autenticado previamente, lo cual solucionamos (de forma no muy satisfactoria) usando sesiones en la aplicación. Como variables de sesión usamos el método de autenticación seleccionado (se le asigna valor cuando estamos seguros de que se ha establecido sesión por alguno de los métodos: PAPI, Shib, Shib+WAYF) y la identificación del usuario tal como la presenta el sistema de autenticación.

Cuando el usuario selecciona uno de los enlaces que se le proporcionan para autenticarse, se activa la variable de tipo de autenticación y se recarga la aplicación. Ésta, al inicio comprueba si está definida esa variable y hace lo siguiente:

- PAPI: Se invoca a check_Access() de phpPoA, el cual nos llevará al AS y de vuelta a la aplicación. En los siguientes accesos esa misma función detecta que estamos autenticados y nos deja pasar.
- Shibboleth contra nuestro IdP: Comprobamos que el valor de alguna de las cabeceras que ha debido poner mod_shib no está vacío y si es así se continua la aplicación. En caso contrario redirigimos al servicio SSO del IdP con los parámetros GET correspondientes a nuestra aplicación.

- Shibboleth con WAYF: Como el anterior pero en este caso redirigimos al servicio WAYF (en nuestro caso usamos el software de SWITCHaai).

En todos los casos, que se nos pida o no usuario/clave dependerá de si nos hemos autenticado previamente ante el sistema de SSO para algún otro servicio, naturalmente.

Las únicas modificaciones adicionales consisten en guardar registro en la base de datos de Consigna, cuando se suben ficheros, de la identidad del usuario, y la comprobación cuando se descargan ficheros de si fueron subidos por un usuario autenticado, como paso previo al chequeo de IP.

En total todas las modificaciones apenas suponen unas pocas docenas de líneas, lo cual nos permite ser bastante optimistas en cuanto a las posibilidades de adaptar otras aplicaciones para un uso federado.

## Logout

En aplicaciones federadas existen dos tipos de logout: de la aplicación y del IdP (mejor dicho, del SSO). Un logout de la aplicación hace que para ésta ya no estemos autenticados y no tengamos acceso a los beneficios que esto proporciona. Pero en cuanto pinchemos en el enlace de autenticación volveremos a estarlo, sin que se nos pida de nuevo usuario/clave, gracias al sistema de SSO. Este tipo de logout se implementa:

- PAPI: Poniendo una cookie Lcook vacía
- Shibboleth: Redirigiendo a la URL de Logout del SP configurado en el fichero shibboleth.xml

El Logout del SSO es un asunto que hay que investigar.

## WAYF

Shibboleth no incluye un servicio WAYF tal que presente al usuario una lista de IdP's para que seleccione el que le corresponde (a diferencia de PAPI, que sí lo implementa). Sin embargo existen implementaciones de dicho servicio que podemos usar. En nuestro caso hemos probado la que han realizado en SWITCH, que funciona bastante bien, está programada en PHP y es relativamente sencilla de configurar e integrar en el fichero shibboleth.xml de configuración del SP.

## Atributos

Como dijimos al principio, el sentido de federar Consigna es guardar opcionalmente un registro de la identidad del usuario que deposita ficheros, por tanto necesitamos obtener del IdP algún atributo de identificación. En el caso de Shibboleth nos basta con el REMOTE_USER, y en el de PAPI, el PAPIuid incluyendo el dominio o al menos el AS.

En realidad la identidad del usuario sólo se utiliza para asegurarnos de que se trata de alguien autorizado y para rastrear en caso de algún problema, por lo que el atributo usado podría ser algún identificador opaco que su institución de origen pudiera resolver en caso de necesidad a un usuario real.

### SP First o IdP First

El caso de uso de Consigna, tal como se ha presentado, es un escenario SP First, en el que la autenticación no se requiere para usarlo, sino que solo es opcional. El servicio requiere conocimiento del mecanismo que se ha utilizado para autenticarse, por tanto, la autenticación debe iniciarse desde el propio servicio, no con un paso previo por el IdP.

## Notas

Si este servicio se llega a implantar de forma federada, cabría la posibilidad de ofrecerlo desde un único sitio a toda la comunidad, en vez de montarlo en cada universidad. En este caso habría que estudiar cuál debería ser su funcionamiento. Si fuera para uso interno de la federación, se podría obligar a la autenticación previa, pero si queremos continuar ofreciendo la posibilidad de que usuarios externos depositen ficheros para los internos, ésa no es una opción. Una posibilidad es que los usuarios autenticados puedan descargarse cualquier fichero, haya sido subido por usuarios internos o externos, autenticados o no. Y los ficheros subidos por usuarios externos no autenticados puedan ser bajados sólo desde las IPs de la federación (aparte de usuarios autenticados, claro).

Asunto para debate...