ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

PAPOID: PAPI OpenID Server

Realizado por
Teresa Matamoros Casas

Dirigido Por
DIEGO R. LOPEZ Y MANUEL VALENCIA

Departamento
DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

Sevilla, Junio de 2008

Índice general

Índice de figuras

Capítulo 1

Introducción

El **Servidor de Identidad PAPI OpenID (PAPOID)**[1] es una aplicación web que ofrece un punto de conexión entre los servicios de identidad de las instituciones que utilizan PAPI y aquellos sitios web que utilizan el protocolo OpenID Authentication para la identificación de sus usuarios.

PAPOID nace dentro del **Servicio de Identidad de RedIRIS**[25] (SIR). El SIR es un servicio de RedIRIS[2] que ofrece un gateway de conexión entre los servicios de identidad de las institucyrtesiones afiliadas y proveedores de servicio, a nivel nacional e internacional.

El SIR se basa en tecnologías de federación de identidades, de manera que:

- Los *usuarios* se identifican en los servidores locales de la institución, utilizando el procedimiento de identificación definido por la misma y sin que las credenciales sean expuestas fuera del dominio local.
- Los *administradores* de los servicios de identidad de las instituciones tienen control total sobre los procedimientos de identificación y los atributos asociados a cada usuario.
- Cada *institución* aplica de manera autónoma los mecanismos de control que considere necesarios para ofrecer a sus usuarios la posibilidad de decidir acerca de la información personal susceptibe de ser transmitida.

[1] PAPI (Point of Access to Providers of Information) es un sistema desarrollado por RedIRIS que provee un control de acceso a recursos de información electrónica restringidos. Intenta mantener la autenticación como un aspecto local de la organización a la que pertenezca el usuario, mientras que da a los proveedores de información total control sobre los recursos que ofertan.

[2] RedIRIS es la red académica y de investigación nacional, patrocinada por el Plan Nacional de I+D+I y que está gestionada por Red.es.

- *RedIRIS* proporciona una conexión segura, fiable y conforme a estándares entre las instituciones y los proveedores de servicio. Los proveedores de servicio aplican de manera autónoma los mecanismos de control de acceso a los recursos que tienen bajo su control. Es importante tener en cuenta que los proveedores de servicio pueden ser tanto entidades ajenas al entorno académico (comerciales, gubernamentales, etc.) como dentro del mismo, ya pertenezcan a RedIRIS u a otra red académica nacional (NREN).

La versión actual del SIR utiliza el protocolo de federación PAPI v.1 y soporta los siguientes protocolos de salida:

- PAPI v.1[20]
- Shibboleth 1.3[1]
- eduGAIN[3][26] con perfil SAML 1.1[3]
- OpenID (versión 1.1 y 2.0) [11, 12]

De esta manera, y gracias a PAPOID, todos aquellos usuarios que pertenezca a una federación que utilice PAPI dispondrán de una identidad de OpenID que podrá utilizar para autenticarse en todos aquellos sitios web que soporten el protocolo *OpenID Authentication* versión 1.1 y 2.0.

En este servidor se ha considerado además la posibilidad del intercambio de perfiles del usuario entre los puntos finales, haciendo uso de la extensión al protocolo de OpenID *Simple Registration Extension 1.0[13]* (SREG). Así se ofrece al usuario la posibilidad de decidir qué información es la que quiere que sea transmitida al sitio web que utiliza OpenID.

En este documento, y en primer lugar, se analiza la importancia de la identidad digital a día de hoy, pues es fundamental para entender la necesidad de tener protocolos de autenticación y autorización seguros.

En el capítulo titulado “Infraestructuras federadas de autenticación y autorización” se describe el escenario en el que tiene sentido la utilización de este servidor PAPI OpenID.

[3] eduGAIN es el nombre de una Infraestructura de Autenticación y Autorización (IAA) construida dentro de GEANT2.

Posteriormente se entra de lleno en la descripción de los protocolos PAPI y OpenID (capítulos 4 y 5) puesto que son los pilares básicos en los que se apoya PAPOID. Y una vez que el lector haya adquirido los conocimientos básicos necesarios para comprender este proyecto se procede a la descripción del mismo. Tras estos apartados, se recogen las conclusiones a las que se ha llegado tras la realización de este proyecto y, en último lugar, se ve un ejemplo de uso de este servidor.

Capítulo 2

Identidad Digital

Una parte cada vez más importante de nuestra vida cotidiana gira en torno a Internet, por lo que necesitamos diseñar nuevas vías para reinventar nuestras interacciones sociales del mundo real, de modo que funcionen en el mundo virtual. Las tecnologías que permiten el tratamiento de la identidad están llamadas a desempeñar un papel clave en este proceso.

Es fundamental hacer frente a los principales problemas que han surgido relacionados con el impacto de las nuevas tecnologías en el tratamiento de la identidad, a fin de crear un entorno en el que reine la confianza de cara a la sociedad de la información.

Las personas pueden necesitar establecer su identidad, con diversos grados de certeza, en una amplia variedad de circunstancias. Los distintos tipos de casos y los mecanismos de identificación relevantes no siempre están claramente definidos y muchos suponen circunstancias híbridas, tomadas del mundo real y del mundo electrónico. Pero normalmente al final de la escala se encuentra la autenticación: ¿Es Alicia realmente Alicia? Seguida de la autorización: Éste es Juan:,¿tiene permiso para ver este documento?

La cuestión de la identidad no es nueva. Durante mucho tiempo, las empresas han venido desarrollando una gran variedad de sistemas en este campo, desde contraseñas y bases de datos de clientes a cookies que se alojan de manera transparente en nuestro ordenador y envían o señalan datos sobre nosotros, y posiblemente sobre nuestras actividades en línea, a los sitios web que visitamos.

Ahora bien, las organizaciones no son las únicas que concentran esfuerzos para ser reconocibles en Internet. Tener una identidad online es también una necesidad individual. Pertenecer a la sociedad

de la información equivale a ser alguien dentro de ella. Se forma parte con sólo participar de alguna manera en Internet, ya sea escribiendo un comentario, publicando una fotografía o siendo mencionado por otros. Pero ser alguien con nombre propio requiere un poco más de dedicación.

No sólo es importante ser alguien en Internet, sino también la propia privacidad individual es un tema cada vez más importante en el contexto de la sociedad de la información. El tratamiento de la identidad ofrece un medio que permite a las personas controlar la naturaleza y cantidad de información personal que se facilita sobre ellas. En particular, para conseguir la privacidad, las personas pueden utilizar pseudónimos y determinar el grado de relación entre las diferentes apariciones de sus datos. Mediante el uso de pseudónimos seguros y autenticados se puede conseguir la responsabilidad de un individuo sobre sus actos, sin facilitar datos personales. Así pues, los sistemas de tratamiento de la identidad que refuerzan la privacidad permiten a los usuarios un control sobre los datos asociados a su identidad digital mucho mejor que antes. Tales sistemas son necesarios en todas las comunicaciones informatizadas, y aún más con la llegada de las nuevas tecnologías, como las comunicaciones móviles UMTS o la informática ubicua.

En definitiva, podría decirse que en el mundo actual estamos habituados a convivir con los conceptos tradicionales de identidad, donde las personas actúan de acuerdo con sus funciones y, generalmente, son capaces de resolver los conflictos relacionados con las mismas. Tienen una comprensión intuitiva de en quién puede confiarse, dependiendo de la situación, de su función y de los demás participantes en el proceso de comunicación. Ahora este problema se traslada a la sociedad de la información, y es aquí donde surge la necesidad de definir una *"identidad digital"*.

Capítulo 3

Infraestructuras federadas de autenticación y autorización

A día de hoy, las organizaciones han tenido la necesidad de proteger el acceso a determinados recursos, donde sus usuarios tenían que autenticarse previamente si querían acceder a ellos. La **autenticación** es el proceso mediante el cual el usuario se identifica en su organización y, en el caso de que todo haya ido correctamente, obtiene unas credenciales con la información de su identidad.

Otro de los elementos que forman parte de la seguridad en la infraestructura de la organización es el de permitir acceder al recurso dependiende de qué rol o atributos tenga el usuario. Este proceso recibe el nombre de **autorización** y permite una granularidad más fina en la seguridad del entorno, ya que no todos los usuario podrán acceder a los mismos recursos.

Para aquellas organizaciones que querían o necesitaban acceder a recursos protegidos, tanto de su entorno como de otras entidades, se les planteaba el despliegue mostrado en la figura 3.1. Como puede verse, cada vez que el usuario quiere acceder a una aplicación web tiene, por lo general, que realizar la autenticación, como podría ser introducir un par “nombre de usuario/contraseña”, en cada una de ellas.

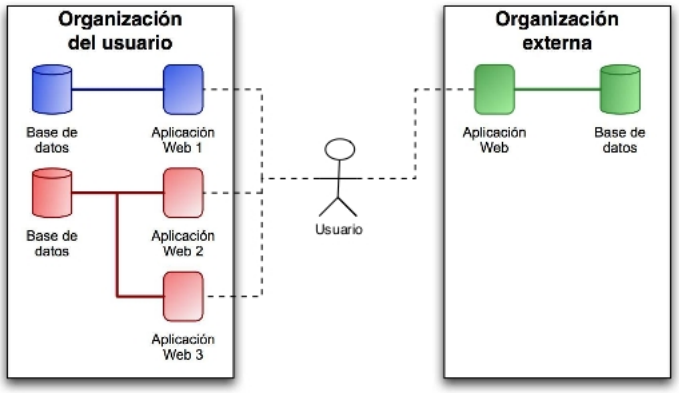

Figura 3.1: Usuario ante una infraestructura no federada

Este escenario conlleva una serie de desventajas, siendo la raíz de ellas que el usuario necesita una identificación diferente, aunque sean los mismos valores en cada una de ellas, por cada aplicación web. Pero este no sólo es un problema para el usuario, ya que su organizacion, y en general cada organización que necesite autenticarlo, tiene que mantener en diferentes sistemas de almacenamiento múltiples entradas de identificacion de un mismo usuario.

Los principales factores negativos de este escenario son:

- *Aumento de los costes de tiempo de registro y administración de los usuarios*, ya que todas las organizaciones tienen que gestionar cuentas de identidad para ellos, en la mayoría de los casos sin posibilidad de automatizar el proceso.
- *Aumento de los costes de recursos físicos*, puesto que hay un aumento exponencial del uso de la red y de los sistemas hardware que se necesitan. Ya no tenemos la información de los usuarios de nuestra organización sino que debemos almacenar el de todas las organizaciones.
- Posibilidad de que ocurran *inconsistencias en los sistemas que almacenan los datos* de un usuario en las distintas organizaciones.
- Los usuarios deben recordar *múltiples pares "nombre de de usuario"/"contraseña"*. Esto provoca que la seguridad del sistema tenga una fuerte dependencia en ellos.

La Identidad Federada es una de las soluciones propuestas para abordar estos problemas asociados a la gestión de la identidad digital. El valor añadido respecto a otras soluciones es la gestión de identidad interdependiente entre compañías, lo que se denomina *Federated Identity Managemen*t.

Mediante soluciones de Identidad Federada los individuos pueden emplear la misma identificación personal (típicamente usuario y contraseña) para identificarse en redes de diferentes departamentos o incluso empresas. De este modo, las empresas comparten información sin compartir tecnologías de directorio, seguridad y autenticación, como requieren otras soluciones (metadirectorio, Single Sign On, etc.). Para su funcionamiento es necesaria la utilización de estándares que definan mecanismos que permiten a las empresas compartir información entre dominios. El modelo es aplicable a un grupo de empresas o a una gran empresa con numerosas delegaciones y se basa en el "círculo de confianza" de estas, un concepto que identifica que un determinado usuario es conocido en una comunidad determinada y tiene acceso a unos servicios específicos.

En definitiva, las infraestructuras federadas tienen como objetivo delegar la identificación de cada usuario a la organización a la que pertenece, en la cual se realizará el proceso de autenticación una sola vez. Ahora cada vez que el usuario quiera acceder a un recurso protegido, se limitará a presentar las credenciales obtenidas al autenticarse.

En este nuevo escenario federado identificamos dos componentes principales:

- *Poveedor de identidad*: realiza la autenticación del usuario y emite sus credenciales. Éstas no sólo contienen información sobre si la autenticación ha sido correcta, sino que también incluirá una serie de atributos asociados a él, como por ejemplo si es un alumno o un profesor. El objetivo es reducir al máximo la información más sensible del usuario, como su nombre y apellidos, y basar los derechos de acceso en qué tipo de usuario es.
- *Proveedor de servicio*: comprueba las credenciales del usuario, y en el caso de que no sean válidas o suficientes, o bien el usuario no las haya obtenido previamente, niega el acceso al recurso protegido.

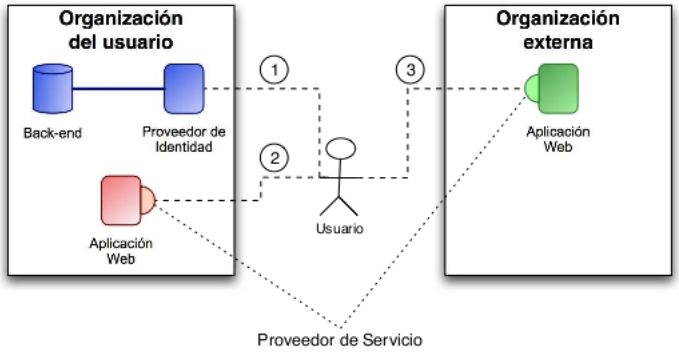

Figura 3.2: Usuario ante una infraestructura federada

La figura 3.2 refleja cómo quedaría el mismo ejemplo de la figura 3.1, pero tras la implantación de una federación. Ahora el usuario que quiere acceder a un recurso protegido en una federación, en un primer momento (*paso 1 en la figura 3.2*) realiza el proceso de autenticación en el proveedor de identidad (IdP) de su organización, así obtiene sus credenciales, que generalmente están cifradas de manera que sólo el proveedor de identidad y los proveedores de servicio pueden leerlas. En el momento que quiera acceder a una aplicación web de su organización *(paso 2 en la figura 3.2*) no tendra que volver a identificarse, sino que la misma peticion envía también sus credenciales. Exactamente igual ocurre cuando accede a la apliación web (*paso 3 en la figura 3.2*) de la organización externa que ha instalado un proveedor de servicio para proteger su recurso.

Las ventajas que ofrece este escenario federado son:

- Las organizaciones no tienen que registrar los datos de los usuarios de otras organizaciones. Además de no aumentar los costes de tiempo en gestionar los usuarios de otras entidades, se reduce la complejidad de las políticas de seguridad y no interfiere en el cumplimiento de las leyes de protección de datos.
- Los usuarios tienen que recordar un solo par de “nombre de usuario/contraseña”. Esto supone una mayor comodidad para los mismos, pero en detrimento de la seguridad, pues si un atacante consiguiera este par el número de recursos que se verían comprometidos sería mayor que en el caso en que se cada uno de ellos hubiese estado protegido por su propio par de “nombre de usuario/contraseña”.

- Los recursos son ofrecidos a todos los usuarios de las organizaciones, sin necesidad de informar a aquellos de la existencia de cada una de las entidades.

Existen diversos software para desplegar IAA (Infraestructura de Autenticación y Autorización). Hasta la publicación de SAML[3] por parte de OASIS en noviembre de 2002, no existía ningún tipo de estandarización a la hora de desarrollar este tipo de programas, lo que ha derivado en una amplia gama de protocolos y de componentes.

Aunque son muchos los componentes software que permiten instalar una IAA, los más extendidos son:

- **PAPI**[2]. Desarrollado por RedIRIS, la primera versión apareció en el año 2000. Define tres componentes principales, que se ven más detalladamente en el capítulo dedicado a PAPI:
  - *Servidor de Autenticación* (AS).
  - *Punto de Acceso* (PoA).
  - *Grupo de Puntos de Acceso* (GPoA).
- **Shibboleth**[1]. Tras la publicación de SAML 1.0 por parte de OASIS, Internet2 comenzó a desarrollar este software como implementación de dicho estándar. Se identifican dos componentes principales:
  - *Proveedor de Identidad* (IdP): Gestiona las credenciales del usuario y emite aserciones SAML que contienen sentencias de autenticación, a través del servicio de Single Sign-On (SSO) y de atributos enviados por la Autoridad de Atributos (AA). Una característica especial de este componente es que necesita de un sistema externo que se encargue de la autenticación de los usuarios. Una vez que el servicio de SSO comprueba que el usuario se ha autenticado correctamente, entonces emite la sentencia para indicar que la autenticación ha sido correcta.
  - *Proveedor de Servicio* (SP): Realiza el control de acceso a una aplicación web, redirigiendo al usuario a un servicio Where Are You From (WAYF) donde podrá elegir cuál es su IdP en el caso de que no tenga unas credenciales válidas. El SP podrá realizar a su vez una petición directa al servicio AA del IdP del usuario para solicitar sus atributos y comprobar si cumplen la política de acceso al sitio.

- **A-Select**[5]. El sistema open source de autenticación A-Select es una iniciativa de la Red Académica Nacional de Holanda en el cual se ofrece un servicio de autenticación para aplicaciones Web. Este producto contiene los siguientes componentes:

  - *A-Select Aware Application*: Es aquella aplicación web que implementa el control de acceso al servicio siguiendo el modelo de confianza de A- Select, comunicándose bien con un A-Select Server o con un A-Select Agent.
  - *A-Select Agent*: Es un proceso ligero que corre en la misma máquina que el componente anterior y se encarga de gestionar todo el protocolo de comunicaciones hacia el A-Select Server. Este componente ofrece una API a las aplicaciones web para que puedan utilizar el modelo de confianza de este producto software.
  - *A-Select Server*: Gestiona el proceso de autenticación del usuario, almacenando en una base de datos los usuarios del sistema y el mecanismo que deben seguir para ser autenticados. De esta forma, no realiza la autenticación sino que redirige al usuario al A-Select Authentication Service Provider.
  - *A-Select Authentication Service Provider* (AuthSP): Es un servidor que realiza la autenticación del usuario, mostrándole un interfaz web que realmente comprueba sus datos.

- **Liberty**[7] . Es el sistema más exitoso dentro del marco empresarial a la hora de desplegar infraestructuras de autenticación y autorización. El proyecto es gestionado por Liberty Alliance y basa su éxito en ser los primeros en utilizar SAML 2.0. A través de esta iniciativa se han desarrollado hasta la fecha los siguientes proyectos:

  - *Liberty Federation* (LF): Sistema para gestionar la autenticación, la seguridad y la privacidad de una infraestructura, es decir, permite un control de acceso a recursos y el proceso de autenticación de los usuarios obteniendo un entorno de Single Sign-On sobre una red o dominio. Este sistema tiene una arquitectura típica en IAA y sus componentes son:
    - *Proveedor de Identidad* (IdP): Gestiona la información que hay alrededor de la identificación de un usuario o componente, emitiendo a su vez las aserciones de autenticación y de atributos a otros proveedores.
    - *Proveedor de Servicio* (SP): Este proveedor gestiona el control de acceso a un recurso protegido.
  - *Liberty Web Services* (LWS): Framework para la gestión de servicios web integrados dentro de un entorno Liberty Federation, ofreciendo además un conjunto de aplicaciones

protegidas de geolocalización, agenda, calendario y servicio de mensajería entre otros. En este sistema se identifican dos roles completamente diferentes:

- *Web Service Consumer* (WSC): Representa a quien ha realizado la petición a un servicio web.
- *Web Service Provider* (WSP): Representa a la entidad que proporciona un servicio web al entorno .

Este framework ofrece una serie de componentes, los cuales utilizarán tanto WSCs como WSPs:

- *People Service* (PS): Servicio que permite compartir información social con otros servicios.
- *Interaction Service* (IS): Servicio que gestiona las preferencias de los usuarios a la hora de compartir información u ofrecer atributos no enviados por un IdP.
- *Discovery Service* (DS): Servicio que facilita el registro de otros servicios, ofreciendo a su vez un directorio de servicios web para cada usuario.
- *Single Sign-On Service* (SSO): Proporciona un servicio que permite obtener aserciones de autenticación.

- **Sun Access Manager**[8]. Producto originario de Sun Microsystems, que tiene a su vez una edición de código abierto llamado Open Single Sign-On. Su objetivo es ofrecer una solución completa que ofrezca autenticación y autorización en entornos Web basados en Java, federados y basados en servicios web.

  Aunque tiene una administración centralizada, Sun Access Manager se compone de los siguientes elementos:

  - *Authentication Service*: Gestiona la identificación del usuario.
  - *Policy Service*: Este servicio evalúa una política de acceso a un recurso protegido con respecto al usuario que intenta acceder a él.
  - *User Session Management*: Una sesión de usuario es el intervalo entre el momento en el que el usuario accede a una aplicación protegida y el momento en el que sale de ella. Durante cada sesión, este servicio mantiene y gestiona información sobre las interacciones que realiza el usuario con dicha aplicación protegida.
  - *SAML Service*: Este servicio se encarga de emitir información, tanto de identificación como de atributos, del usuario usando mensajes SAML.

  - *Identity Federation Service*: Este servicio unifica las diferentes identificaciones que tiene para las aplicaciones y ofrece una aplicación global federada.
  - *Logging Service*: Mantiene información sobre las acciones que realiza el usuario en el entorno protegido por el Sun Access Manager.

- **OpenID**[11, 12]. Orientado a aplicaciones web, es un sistema de Single Sign-On descentralizado donde no existe ningún servidor central que se encargue de gestionar la autenticación del usuario. Consta de dos elementos principales:

  - *OpenID provider*.
  - *Relying party*.

  La descripción del protocolo OpenID se ve de una manera más detallada en el capítulo 5.

- **OAuth**[10]. Recientemente un grupo de expertos dentro del mundo open source ha publicado la especificación 1.0 de OAuth, siendo un protocolo de protección de recursos orientado a la delegación, es decir, una aplicación web, Consumer, quiere obtener un conjunto de datos protegido por otra aplicación web, Service Provider. El objetivo es permitir el acceso a unos datos sin que se compartan los datos de los usuarios. Pese a ser una tecnología muy reciente y sin ninguna implementación de referencia, al estar respaldada por muchas empresas que mantienen populares portales web 2.0, como Twitter o Google, se prevé un fuerte impacto en el sector tecnológico de la autenticación y autorización.

Capítulo 4

PAPI

4.1. Introducción

PAPI es un sistema para facilitar el acceso, a través de Internet, a recursos de información cuyo acceso está restringido a usuarios autorizados. Los mecanismos de autenticación empleados para identificar a los usuarios se han diseñado para ser lo más flexibles posible, permitiendo que cada organización emplee un esquema de autenticación propio, manteniendo así los datos dentro de su propio ámbito, a la vez que los proveedores de información disponen de datos suficientes para realizar estadísticas.

Los mecanismos de control de acceso son transparentes para el usuario y compatibles con los navegadores comunmente empleados en cualquier sistema operativo. Dado que PAPI emplea procedimientos estándar HTTP, su uso para proveer servicios de identidad digital y control de acceso no requiere de ningún hardware o software específico, garantizando a los usuarios un acceso ubicuo a cualquier recurso de información al que tengan derecho.

PAPI consta de tres elementos independientes: el *Servidor de Autenticación* (AS), el *Punto de Acceso* (PoA) y el *Grupo de Puntos de Acceso* (GPoA). Esta estructura hace que el sistema tenga una gran flexibilidad y permite su integración en diferentes entornos operativos. No se requiere ningún tipo de correspondencia entre un determinado AS y un determinado PoA: un PoA es capaz de manejar peticiones desde cualquier número de ASes y dirigirlas hacia cualquier número de servidores web.

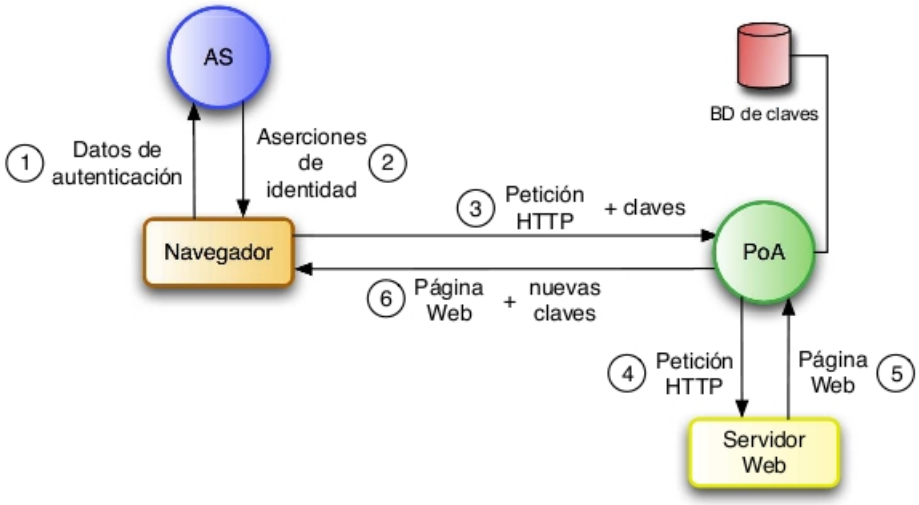

Figura 4.1: Arquitectura de PAPI

Una importante propiedad de este sistema es que ofrece una compatibilidad absoluta con cualquier otro sistema de control que se emplee, dado que no impone requisitos de ningún tipo a otros procedimientos adicionales que se empleen con este fin. En otras palabras, el control de acceso PAPI es completamente ortogonal a mecanismos como la protección por medio de contraseñas, filtros basados en direcciones IP, control de acceso por medio de *Transport Layer Security* (TLS), etc.

4.2. Authentication Service

El propósito del servicio de autenticación (AS) de PAPI es ofrecer a los usuarios un punto único de autenticación y proporcionarles (de manera completamente transparente) todas las claves temporales que les permitirán acceder a los recursos para los que están autorizados. Aunque este servidor no impone ningún tipo de restricción en la manera en que el usuario se identifica, generalmente el proceso de autenticación se realizará en base a su par nombre de usuario y contraseña, comprobando esta información con un sistema de directorio o gestor de datos, también llamados back-end. PAPI ofrece en el AS diferentes conectores de back-end, como por ejemplo LDAP, base de datos, servidores IMAP y servidores POP3. Además, puede mantener en alguno de estos sistemas una lista de puntos de acceso (PoA) a los que el usuario puede acceder.

4.3. Point of Access

El PoA realiza el control de acceso efectivo para un conjunto de localizaciones web dentro de un determinado servidor. El proveedor de información (o el operador de los servidores web) tiene la responsabilidad de gestionar el punto de acceso de acuerdo con su política. Un PoA puede ser adaptado a cualquier servidor web, con independencia de la plataforma sobre la que está implementado. Es más, un servidor web puede contener más de un PoA, y, además, un PoA puede controlar el acceso a más de un servidor web.

4.4. Group Point of Access

Podemos combinar los diversos PoAs de manera jerárquica en grupos controlados por un PoA de grupo (un GPoA), a través del que se validan los intentos iniciales de acceso. De esta manera, el navegador del usuario debe únicamente cargar las claves temporales para los GPoAs en la raíz de la jerarquía.

4.5. Protocolo

El protocolo empleado por PAPI puede considerarse dividido en dos fases: autenticación y control de acceso. La fase de autenticación comienza cada vez que un usuario accede al servidor de autenticación para obtener las claves temporales válidas. Durante el periodo que dura la validez de estas claves, este usuario no necesita volver a pasar por esta fase. El usuario debe reiniciar la fase de autenticación antes de la expiración de sus claves únicamente en ciertos casos específicos:

- Cuando las claves temporales son eliminadas del navegador. En la implementación actual, esto ocurre cuando las cookies son borradas explícitamente o cuando el fichero de cookies se ve dañado.
- Cuando ocurre una corrupción de las claves temporales.
- Cuando las claves temporales se copian a otro navegador y son empleadas por otro usuario.
- Cuando la clave simétrica principal (la K1 descrita más adelante) del punto de acceso ha sido cambiada.

4.5.1. Fase de autenticación

Esta etapa se inicia en el Servidor de Autenticación, donde el usuario es autenticado y acaba, una vez que todo se ha llevado a cabo de manera exitosa, cuando un conjunto de claves temporales (dos por cada uno de los (G)PoA para los que el usuario está autorizado) es almacenado por el navegador del usuario.

El usuario accede al Servidor de Autenticación por medio de un navegador Web y proporciona los datos que el AS requiere para reconocerlo como un usuario válido. Qué datos son estos y su validación es una cuestión local al AS. Si la autenticación es correcta se aplica la política que la organización utilice para proporcionar acceso a los recursos disponibles a través de PAPI, generando una lista de URLs que se corresponden con los puntos de acceso que deben ser contactados para descargar las claves temporales. Esta lista de URLs se envía al navegador del usuario, integrada en una página que da cuenta del resultado positivo del proceso de autenticación. La política de acceso de la organización puede basarse en atributos tales como el tipo de usuario, su relación con la organización, su rango dentro de ella, el número de accesos que la organización puede hacer a un determinado recurso, etc.

Cada una de los URLs mencionados anteriormente incluyen una referencia al procedimiento que el PoA correspondiente emplea para generar las claves temporales, junto con tres parámetros que se pasan empleando el método GET o HTTP: el identificador del AS, un código de petición y una aserción cifrada acerca del usuario. Esta aserción contiene:

- Una cadena de texto que describe al usuario y sus derechos. Una descripción típica del usuario incluye referencias al usuario y a los grupos a los que pertenezca, anonimizados si es necesario.
- Duración que se solicita para la clave temporal principal que se va a generar.
- Una marca de tiempo, para evitar ataques por medio de repeticiones de los datos.

Dado que el navegador del usuario recibe estos URLs incorporados en una página HTML, contacta de manera completamente transparente los puntos de acceso que están en la lista. Cada uno de ellos verifica la integridad y la marca de tiempo de la petición, empleando para ello la clave pública del servidor de autenticación. En este punto se verifican también las reglas de control de acceso aplicables a los datos que se reciben acerca del usuario. Si todo es correcto, se generan dos claves temporales: la clave temporal principal Hcook y la clave temporal secundaria Lcook. La clave temporal principal, junto con una cadena generada de manera aleatoria, se incluye en un registro de claves temporales

mantenido por el punto de acceso. Ambas claves temporales se envían al navegador del usuario, codificadas como cookies HTTP.

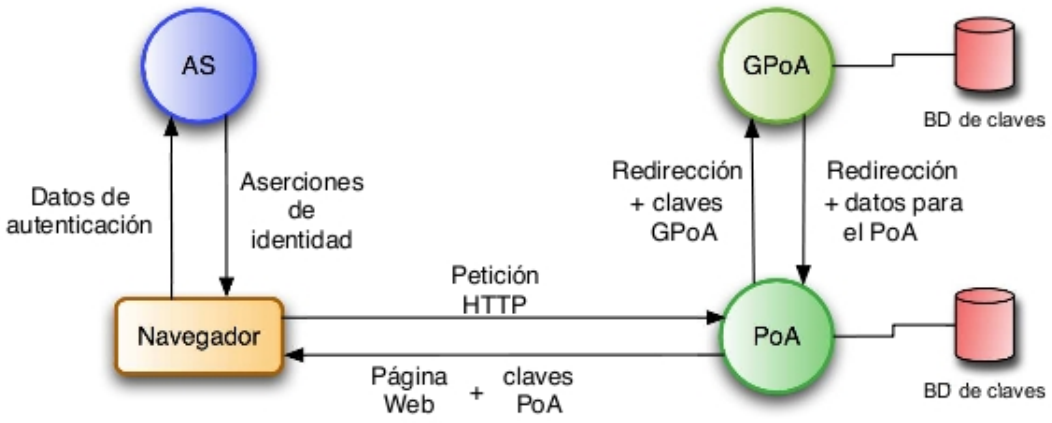

Figura 4.2: Interacción entre PoA y GPoA

Además de ser contactado directamente por un AS, un PoA puede también ser contactado por otros PoAs cuando éstos se hayan incluido en un grupo controlado por un PoA de grupo (GPoA). Un GPoA recibe peticiones de sus PoAs subordinados por medio de redirecciones HTTP, como muestra la figura 4.2. Si los procedimientos de control de acceso devuelven un resultado válido para el GPoA, éste construye un URL similar al descrito para el AS (empleando un código de petición distinto) y redirige el navegador de vuelta al PoA subordinado que contactó inicialmente al GPoA. En función de los datos que recibe por medio de esta redirección, el PoA subordinado decide si genera su par de claves temporales.

4.5.2. Claves temporales

El uso de las claves temporales diferentes pretende un balance entre seguridad, con el objetivo de evitar accesos no autorizados por medio de la duplicación de las claves, y la eficiencia del protocolo, intentando evitar la repetición excesiva de cálculos criptográficos largos.

Hcook es la clave primaria, criptográficamente más fuerte y que contiene más datos. Contiene un conjunto de campos con información de control de acceso, codificados con una clave simétrica (K1) que es exclusiva del PoA. Los campos de control de acceso son:

- Un código de usuario, derivado de la aserción recibida desde el AS.
- La localización a la que da acceso esta clave.
- El momento en que la clave expira.
- Un bloque generado de manera aleatoria.

Las entradas en el registro de claves temporales del PoA mantienen las claves temporales primarias que son válidas. Este registro se usa para evitar los accesos ilegales por medio de duplicación de cookies: cada vez que el PoA comprueba la validez de una Hcook recibida, compara el bloque aleatorio que contiene con el último que se generó para ella y que se almacenó en el registro.

Lcook es la clave temporal secundaria. Se emplea en la fase de control de acceso para realizar comprobaciones más rápidas y consiste también en un conjunto de campos de control de acceso, codificados con otra clave simétrica (K2), que también es exclusiva del PoA, aunque más corta de K1. Los campos que contiene son:

- La localización a la que da acceso esta clave.
- El momento en que la clave fue generada.

El tiempo de vida típico de una clave secundaria es de unos pocos segundos, mientras que las claves primarias son recalculadas en intervalos del orden de varios minutos. Una página primaria puede considerarse como una clave de página, mientras que la secundaria pretende evitar retardos adicionales cuando se están cargando los elementos que componen la página.

4.5.3. Fase de control de acceso

En esta fase el punto de acceso se encarga de verificar las claves temporales asociadas con la localización que el navegador ha solicitado. El uso de dos claves temporales permite emplear la clave secundaria (cuya validez es muy corta) para reducir la complejidad de los cálculos criptográficos necesarios en cada decisión de acceso, incrementando así la capacidad de respuesta del sistema. Dado que la Lcook tienen un periodo de validez muy corto, cuando expira se lleva a cabo una verificación

larga: se decodifica Hcook, se genera un nuevo par de claves temporales y se actualiza el registro de claves temporales. Dado que las claves están almacenadas como cookies HTTP, cada vez que se intenta acceder una localización controlada, el navegador las envía automáticamente, sin necesidad de ninguna intervención por parte del usuario.

Cuando un PoA recibe una petición de acceso a una localización protegida, lleva a cabo los siguientes pasos:

1. Busca las claves temporales almacenadas en las cookies identificadas como Lcook y Hcook. Si no se han recibido las cookies, el PoA intenta redirigir la petición hacia su correspondiente GPoA o AS (si está definido). En otro caso, la petición es rechazada.

2. El punto de acceso verifica la clave secundaria: decodifica Lcook usando K2 y comprueba si son correctos la localización y la marca de tiempo. Si esta comprobación tiene éxito, la petición es aceptada.

3. Si el tiempo de validez de Lcook ya ha expirado es necesario proceder a la verificación de la clave primaria: Hcook ha de ser decodificada usando K1. La petición es rechazada si alguna de las siguientes condiciones no se satisface:

   - El periodo de acceso no ha expirado.
   - La localización codificada en la clave es válida.
   - El bloque aleatorio codificado en la clave coincide con el que se encuentra almacenado en el registro de claves temporales.

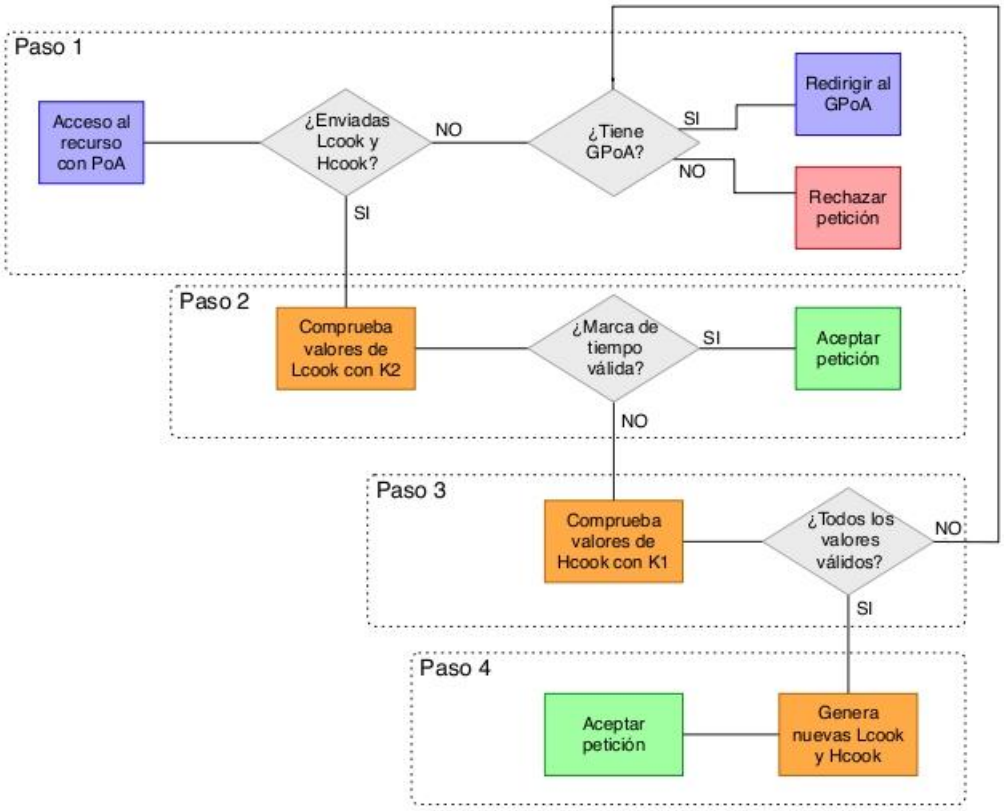

Figura 4.3: Procesamiento de una petición en un PoA

4. Si la clave primaria es correcta, una nueva clave (conteniendo un nuevo bloque aleatorio) es generada y almacenada en el registro de claves temporales. Asimismo, una nueva clave secundaria es generada, y los dos nuevos valores son enviados codificados como cookies hacia el navegador del usuario junto con los datos de la petición original. A partir de este momento, si se reciben nuevas peticiones con el valor antiguo de Hcook, el valor de su bloque aleatorio no coincidirá con el almacenamiento en el registro, por lo que el PoA asume que se ha producido una duplicación de cookies y rechazará las peticiones.

Capítulo 5

OpenID

5.1. Introducción

OpenID es un sistema de autenticación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una URL (o un XRI en la versión actual) y puede ser verificado por cualquier servidor que soporte el protocolo. OpenID es un estándar abierto, libre y descentralizado que permite a los usuarios controlar la cantidad de información que proveen.

En los sitios que soporten OpenID, los usuarios no tienen que crearse una nueva cuenta de usuario para obtener acceso. En su lugar, solo necesitan disponer de un identificador creado en un servidor que verifique OpenID, llamado proveedor de identidad o IdP. Dado que OpenID es descentralizado, cualquier sitio web puede utilizar OpenID para autenticar a sus usuarios, esto es, OpenID no necesita que una autoridad central confirme la identidad digital de los usuarios.

A diferencia de arquitecturas Single Sign-On, OpenID no especifica el mecanismo de autenticación. Por lo tanto, la seguridad de una conexión OpenID depende de la confianza que tenga el cliente OpenID en el proveedor de identidad. Si no existe confianza en el proveedor, la autenticación no será adecuada para servicios bancarios o transacciones de comercio electrónico; sin embargo, el proveedor de identidad puede usar autenticación fuerte pudiendo ser usada para dichos fines.

OpenID está ganando fuerza debido a que algunas empresas importantes como Google, Verisign, IBM, AOL, Orange o Yahoo han apostado por la adopción del estándar.

5.1.1. Historia

El protocolo de autenticación original fue desarrollado en 2005 por Brad Fitzpatric y pronto se implementó el soporte para OpenID en LiveJournal y DeadJournal, ganando rápidamente la atención de la comunidad de identificación digital. Seguidamente empezaron los primeros contactos entre los desarrolladores de OpenID y los desarrolladores de software empresarial de la compañía NetMesh, que condujeron a la colaboración en la interoperabilidad entre OpenID y el similar protocolo de identidad Light-Weight (LID). El resultado directo de la colaboración fue el protocolo Discovery Yadis. A este proyecto Yadis se unieron los desarrolladores de XRI/i-names, contribuyendo con su formato Secuencia de Descriptor de Recursos Extensible (XRDS) para su utilización en el protocolo.

Posteriormente, desarrolladores en Sxip Identity empezaron a tratar con la comunidad OpenID/Yadis después de anunciar la versión 2 de su Protocolo de Identidad Extensible Simple (SXIP) a identidades basadas en URLs como LID y OpenID.

Otros de los avances a destacar en el camino de desarrollo de OpenID han sido el desarrollo de la Extensión de Registro Simple (Simple Registration Extension) de OpenID para un primer intercambio de perfiles, los primeros estándares para la extensión del Intercambio de Atributos (Attribute Exchange, AX) debidos a SXIP, junto con los trabajos para dar un soporte XRI completo en OpenID.

5.2. Ventajas del uso de OpenID

A continuación se exponen algunas de las principales ventajas que el uso de OpenID conlleva:

1. *Descentralización*: Es descentralizado, lo que implica que no todos los datos de usuarios están almacenados en un sólo lugar o por una entidad o compañía. En un sistema de infraestructuras federadas, esto sería equivalente a afirmar que la identificación del usuario podría llevarse a cabo de manera local en la organización a la que pertenezca, de manera que la información del individuo es un asunto local a la organización y gestionada por la misma.

2. *Seguridad*: Es más fácil mantener un sólo usuario y contraseña en lugar de 20. Un usuario podría cambiar los hosts de OpenID de manera periódica, con el objetivo de hacer que todo sea más seguro. Se presenta, pues, la misma ventaja que ya se vio en el escenario de las infraestructuras federadas de autenticación y autorización, con lo que no podemos tampoco obviar el inconveniente que este sistema presenta si un atacante consiguiera dicho par de usuario/contraseña.

3. *Mercado*: Incrementa el número de usuarios que se registran en una página por primera vez, pues evita el trámite de tener que rellenar un formulario con todos sus datos, un formulario que en muchas ocasiones puede llevar al usuario a abandonar el proceso de registro.

4. *Facilidad de instalación*: Un sitio web puede convertirse en un proveedor de identidades de OpenID fácilmente. Es muy simple y es lo que lo hace tan atractivo para muchos usuarios.

5. *Manejar identidades*: En lugar de tener que recordar múltiples usuarios y contraseñas, sólo hay una.

6. *Ahorra tiempo* cuando se desean probar nuevos sitios web.

7. Es fácil mantener múltiples identidades.

5.3. Inconvenientes del uso de OpenID

Ahora bien, no todo son ventajas en el mundo de OpenID, también tiene una serie de problemas muy importantes, entre los que podemos destacar los siguientes:

1. *Perfiles de usuarios*: Pérdida de anonimato. Actualmente, cada sitio en el que se tiene una cuenta sabe sobre el usuario lo que éste quiera contarle. Con OpenID, incluso aunque puedan mantenerse múltiples identidades, se están relacionando de manera inherente muchos servicios y en consecuencia perdiendo cierta parte del anonimato del usuario.

2. *Seguridad*: Como ya se ha comentado anteriormente, si el par "nombre de usuario/contraseña" cayeran en manos de un potencial adversario, el número de sitios web en los que la identidad digital del usuario se vería comprometida sería mucho mayor.
   En las especificaciones de OpenID se contemplan distintos aspectos de seguridad, como los referentes al establecimiento de claves, firma de mensajes y mecanismos de verificación, así como con el uso de canales protegidos TLS/SSL; sin embargo, el empleo de estos sistemas de seguridad es completamente opcional. Este aspecto supone una de las principales diferencias

que se establecen con otros estándares usados en infraestructuras federadas, sea éste por ejemplo el caso de SAML, donde se establecen robustos sistemas de seguridad[28].
Estos aspectos de seguridad en OpenID se están tratando de manera activa[29], pero las soluciones propuestas aún están en fases iniciales de desarrollo y cada OpenID tiene la responsabilidad de elegir su propia solución. Hasta que esta situación no se resuelva, no es muy adecuado utilizar este protocolo en comunicaciones que necesiten una alta privacidad, como bancos o sistemas sanitarios.

3. *Usabilidad*: Para el usuario medio, OpenID es aún demasiado confuso para crear una cuenta y usarla. Simplemente el hecho de encontrar una página en la que poder crear su propia identidad de OpenID puede ser difícil; y el proceso de utilizarlo en un página que afirme aceptar este tipo de identidad es tosco y difícil. Por ejemplo, ¿cómo elige un nuevo usuario a su proveedor de OpenID?
Además el proceso de autenticación puede ser confuso, pues implica que debemos dirigirnos a otro sitio web.

En la actualidad hay mucha gente que dispone de cuentas OpenID pero ni siquiera es consciente de ello, como puede ser el caso de los usuarios de Yahoo. OpenID es un buen paso hacia la resolución de algunos de los problemas clave de la identidad digital a través de un estándar abierto que no trata de resolver todos los problemas a la vez, sino que, en su lugar, trata de resolver y manejar los distintos problemas y necesidades conforme van surgiendo.

5.4. Protocolo OpenID Authentication 2.0

El objetivo de esta sección es explicar de manera más detallada en qué consiste el protocolo OpenID Authenticacion versión 2.0.

5.4.1. Terminología

En lo que sigue se detalla el glosario de términos utilizado en OpenID:

- *End User*: Usuario final o persona que quiere acceder con su identidad a un sitio web.
- *Identifier*: Identificador que puede ser una URL, http o https, o un XRI.
- *User-agent*: Navegador web del usuario final que implementa HTTP/1.1.

- *Identity provider u OpenID provider*: OP. Servidor de OpenID en el que un Relying Party confía.
- *Relying Party o Consumer*: RP. Aplicación web que quiere probar que un usuario controla un identificador.
- *OP Endpoint URL*: URL que acepta mensajes del protocolo de OpenID Authentication.
- *OP Identifier*: Identificador para el OpenID Provider.
- *User-Supplied identifier*: Identificador proporcionado por el usuario final al Relying Party, o seleccionado por el usuario en el OpenID Provider. Durante la fase de iniciación del protocolo, el usuario final puede entrar o bien su propio identificador o el del OpenID Provider. Si se usa un identificador de OP, entonces el OP puede ayudar al usuario a seleccionar el identificador que quiere compartir con el Relying Party.
- *Claimed Identifier*: Identificador que el usuario afirma poseer, el propósito final de este protocolo es verificar esta afirmación
- *OP-Local Identifier*: Identificador para un usuario final que es local para un OP concreto, por ello no tiene por qué estar necesariamente bajo el control del usuario.

5.4.2. Resumen del protocolo

1. El usuario final inicia la autenticación presentando un identificador al Relying Party a través del navegador.
2. Después de normalizar el identificador proporcionado por el usuario, el Relying Party aplica el protocolo Discovery sobre el mismo y establece la URL del OpenID Provider que el End User usa para la autenticación. Debe señalarse que el identificador proporcionado puede ser el de un OP, lo que permite la selección del Claimed Identifier en el OP o que el protocolo pueda continuar sin un Claimed Identifier si se está haciendo cualquier otra cosa a través de una extensión.
3. (opcional) El Relying Party y el OP establecen una asociación, es decir, un secreto compartido establecido usando Diffie-Hellman Key Exchange. El OP usa una asociación para firmar los

siguientes mensajes y el Relying Party para verificarlos; así se elimina la necesidad de sucesivas peticiones directas[1] para verificar la firma después de cada autenticación petición/respuesta.

4. El Relying Party redirige al navegador hasta el OP con una petición de autenticación OpenID.

5. El OP establece si el usuario está autorizado para llevar a cabo una autenticación con OpenID y desea hacerlo. La manera en la que el usuario se autentica en el OP y cualquier tipo de política entorno a este aspecto quedan fuera del alcance del protocolo.

6. El OP redirige el navegador hasta el Relying Party con una aserción en la que o bien se afirma o bien se niega la autenticación.

7. El Relying Party verifica la información recibida desde el OP, incluyendo la comprobación de la URL de vuelta, verificando la información obtenida con Discovery y comprobando el nonce[2] y la firma. Para la comprobación de la firma se usa la clave compartida que se estableció durante la asociación o bien se envía una petición directa al OP.

[1] **Comunicación Directa**: Es inicializada por el Relying Party hasta la URL de un OP. Se utiliza para establecer asociaciones y verificar aserciones de autenticación. El mensaje debe ser codificado usando el cuerpo de una petición POST. Todas las comunicaciones directas son HTTP POSTs.

Comunicación Indirecta: En la comunicación indirecta, los mensajes se pasan a través del navegador del usuario. Esta comunicación puede ser inicializada por el Relying Party o el OP. Este tipo de comunicación se usa en peticiones y respuestas de autenticación (checkid_setup y checkid_immediate). Hay dos métodos para la comunicación indirecta: redirecciones HTTP y formularios HTML.

[2] El nonce (Number ONCE) es un número arbitrario que es generado con propósitos de seguridad, tales como un vector de inicialización. Es usado sólo una vez durante una sesión de seguridad.

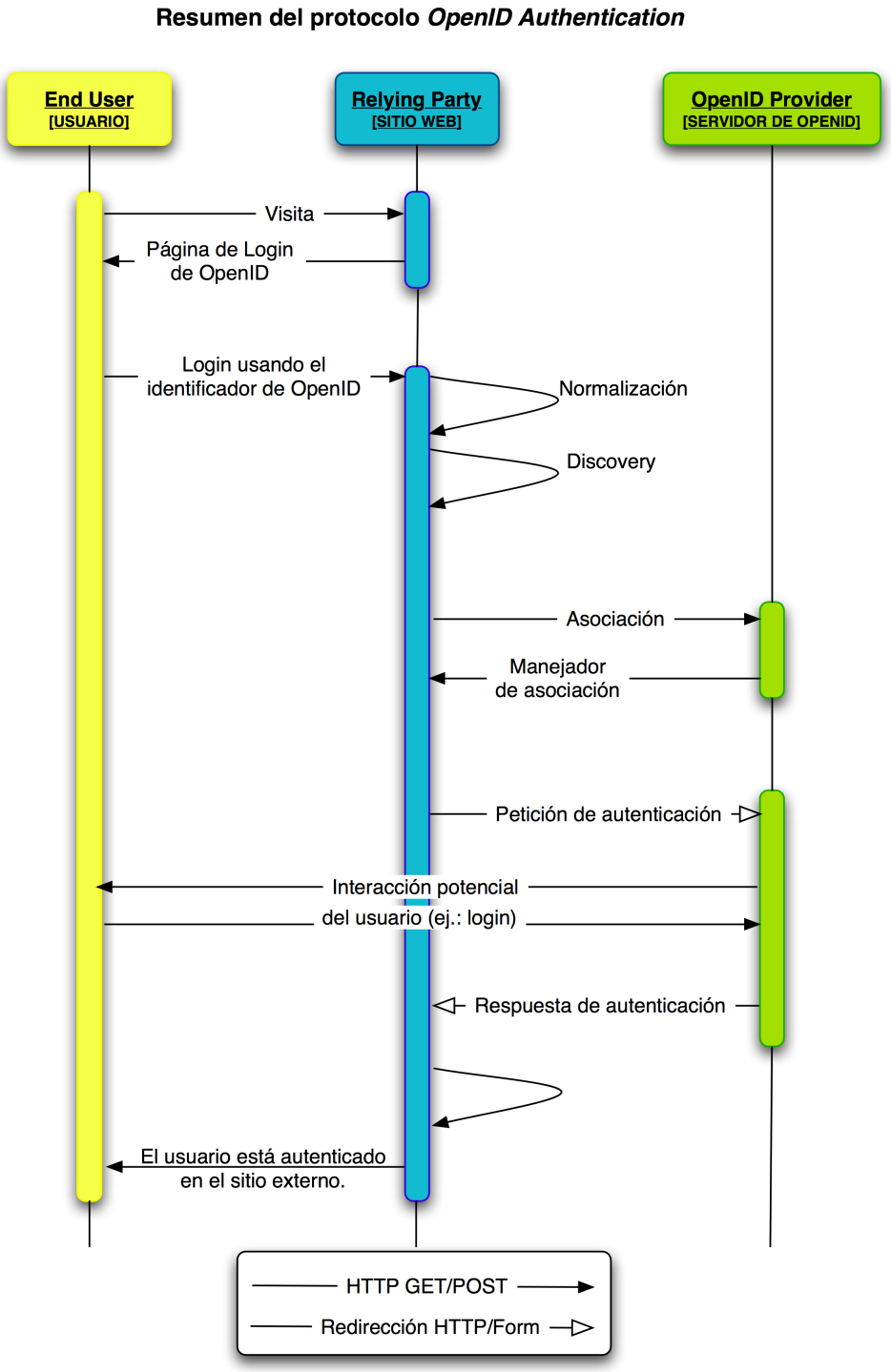

Figura 5.1: Resumen del protocolo OpenID Authentication

Para facilitar la comprensión del protocolo por parte del lector, además de detallar cómo funciona en las siguientes secciones, se adjuntan dos ejemplos del mismo, con los mensajes que se intercambian y el contenido de los mismos, en el apartado 5.4.8.

5.4.3. Identificadores OpenID: URL, XRI

5.4.3.1. Transformar un documento HTML en un identificador

Para que un Relying Party sepa quién es el OpenID Provider de un Identificador, el usuario debe añadir algunas etiquetas en la sección HEAD del documento HTML localizado en su URL asociada. No es necesario que el host del documento HTML sea también el Identity Provider del usuario; la URL identificadora y el proveedor de identidad pueden ser servicios totalmente desacoplados.

Para usar “http://example.com” como el identificador del usuario y “http://openid.example.com” como su proveedor de identidad, la siguiente etiqueta *link* debe ser añadida en la sección HEAD del documento HTML devuelto cuando se busca en la URL del usuario.

```
1 <html>
2    <head>
3         <link rel="openid2.provider" href="http://openid.example.com/">
4    </head>
5    <body></body>
6 </html>
```

Figura 5.2: Ejemplo de un documento HTML usado en un OpenID Identifier

5.4.3.2. Delegar autenticación

Si el host del usuario no dispone de un servidor de OpenID, o el usuario final desea utilizar otro alojado en un host diferente, se necesitará delegar la autenticación. Por ejemplo, si desea utilizar su sitio web “http://www.example.com” como su identificador, pero no desean ejecutar su propio servidor de OpenID.

Si tienen una cuenta con LiveJournal (por ejemplo, usuario ’usuarioEjemplo’), y sabe que Livejournal cuenta con un OpenID Provider y que éste afirmará que el usuario posee dicho identificador, se podrá delegar la autenticación al proveedor de identidad de LiveJournal. Así, para usar

www.example.org como su identificador, pero que el Relying Party realmente esté verificando "http://usuarioEjemplo.livejournal.com" con el Identity Provider localizado en "http://www.livejournal.com/openid/server.bml", hay que añadir las siguientes etiquetas en la sección HEAD del documento HTML devuelto cuando se busca en la URL proporcionada:

```
<html>
   <head>
        <link rel="openid2.provider"
                        href="http://www.livejournal.com/openid/server.bml">
        <link rel="openid2.local_id"
                        href="http://exampleuser.livejournal.com/">
   </head>
   <body></body>
</html>
```

Figura 5.3: Ejemplo de un documento HTML usado para delegar la autenticación

5.4.3.3. Selección de identificador

Si lo que se desea es que es que la selección del identificador de usuario tenga lugar en el OP, entonces la estructura del documento HTML debe ser la que sigue:

```
<html>
   <head>
         <link rel="openid2.provider"
                         href="http://openid.example.com/">
        <link rel="openid2.local_id"
                         href="http://specs.openid.net/auth/2.0/identifier_select">
   </head>
   <body></body>
</html>
```

Figura 5.4: Ejemplo de un documento HTML usado para la selección de un OpenID Identifier

5.4.3.4. XRDS

Supóngase ahora el caso en el que una persona utiliza un identificador XRI. Si dicho usuario dispone de un identificador que sea "http://www.example.com", pero realmente tiene a un Relying party verificando el identificador "http://exampleuser.livejournal.com/" con el OpenID Provider

"http://www.livejournal.com/openid/server.bml", el siguiente trozo de código xml tiene que estar presente en el elemento XRD final del archivo XRDS obtenido al aplicar Discovery a "http://www.example.com/":

```
<Service xmlns="xri://$xrd*($v*2.0)">
  <Type>http://specs.openid.net/auth/2.0/signon</Type>
  <URI>http://www.livejournal.com/openid/server.bml</URI>
  <LocalID>http://exampleuser.livejournal.com/</LocalID>
</Service>
```

Figura 5.5: Ejemplo de un documento XRDS usado en un OpenID Identifier

5.4.4. Smart mode vs Dumb mode

De acuerdo con las capacidades del Relying Party, el funcionamiento del protocolo OpenID Authentication puede adaptarse dando lugar a dos modos de operación distintos. En concreto, soporta los modos *smart mode* y *dumb mode*. En aquellos casos en los que el Relying Party no pueda (o desee) establecer un secreto compartido con el Provider se usará el modo *stateless* o *dumb mode*, donde todo el trabajo de generación de las firmas de los mensajes y comprobación de las mismas recae en el servidor de identidad. Por el contrario, si el Relying Party puede almacenar información del estado, entonces usará el modo *statefull* o *smart mode*, con el que hará un poco más de trabajo al principio para establecer una asociación con el Provider pero después disminuirá el número de aserciones que tienen que intercambiar con éste.

5.4.5. Modos

A la hora de describir los distintos modos, si no se especifica lo contrario, los parámetros intercambiados serán obligatorios.

5.4.5.1. associate

- Descripción: Establece un secreto compartido entre el Relying Party y el OpenID Provider.
- Método HTTP: POST.
- Flujo: Relying Party ->OpenID Provider ->Relying Party

Parámetros de la petición

- *openid.ns*
  Este valor debe estar presente para que la petición sea válida en OpenID Authentication 2.0.

  Si este valor no está presente o su valor es "http://openid.net/signon/1.1" or "http://openid.net/signon/1.0", entonces este mensaje debería ser interpretado usando el modo compatible con OpenID Authentication 1.1.
  Valor: "http://specs.openid.net/auth/2.0”

- *openid.mode*
  Valor: “associate”

- *openid.assoc_type*
  Tipo de asociación preferido. El tipo de asociación define el algoritmo que se va a utilizar para firmar las siguientes peticiones.

  Valor: Un tipo de asociación válido[3].

- *openid.session_type*
  El tipo preferido para la sesión de asociación. Define el método usado para cifrar la clave MAC intercambiada.

  Valor: Un tipo de asociación de sesión válido[4].

  Nota: A menos que se use cifrado en la capa de transporte, “no-encryption” no debe usarse.

- *openid.dh_modulus*
  Parámetro necesario para configurar el intercambio de clave con el cifrado Diffie-Hellman.

  Valor: base64(btwoc(p))

[3] OpenID Authentication define dos tipos válidos de asociación: “HMAC-SHA1" y "HMAC-SHA256".
[4] OpenID Authentication define tres tipos válidos de asociación de sessión: "no-encryption", "DH-SHA1" y "DH-SHA256".

- *openid.dh_gen*
  Parámetro necesario para configurar el intercambio de clave con el cifrado Diffie-Hellman.

  Valor:base64(btwoc(g))
  Por defecto: g=2

- *openid.dh_consumer_public*
  Parámetro necesario para configurar el intercambio de clave con el cifrado Diffie-Hellman.

  Valor:base64(btwoc(g ^ xa mod p))
  Por defecto: g=2

Parámetros de la respuesta

Formato de la respuesta: pares clave-valor[5].

- *ns*
  Valor: "http://specs.openid.net/auth/2.0”

- *assoc_handle*
  El manejador de asociación se usa como una clave para hacer referencia a esta asociación en los sucesivos mensajes.

  Valor: una cadena de 255 caracteres o menos.

- *assoc_type*
  Valor del parámetro openid.assoc_type de la petición.

- *session_type*
  Valor del parámetro openid.session_type de la petición

[5]Un mensaje en formato clave-valor es una secuencia de líneas. Cada línea empieza con una clave, seguida por dos puntos, y el valor asociado a la clave. La línea se termina con una fin de línea, "\n".

- *expires_in*
  El tiempo de vida, en segundos, de esta asociación. El Relying Party no debe usar esta asociación después de que este tiempo haya pasado.

Si la clave intercambio de clave se hace utilizando “no-encryption” entonces el siguiente campo aparecerá en la respuesta:

- *mac_key*
  La clave MAC (secreto compartido) para esta asociación.

Si el intercambio de la clave utiliza Diffie-Hellman entonces aparecerán los siguientes campos:

- *dh_server_pub*
  Valor: base64(btwoc(g ^ xb mod p))
  Descripción: La clave pública Diffie-Hellman del OP.

- *enc_mac_key*
  Valor: base64(H(btwoc(g ^ (xa * xb) mod p)) XOR MAC key)
  Descripción: La clave MAC (secreto compartido), cifrado con el valor del secreto Diffie-Hellman.

Parámetros de la respuesta sin éxito

Si el OP no soporta un tipo de sesión o un tipo de asociación, debe responder con un mensaje directo indicando que la petición de asociación falló. Si hay otro tipo de sesión o asociación disponible, el OP debería incluir esta información en la respuesta.

- *ns*
  Valor: "http://specs.openid.net/auth/2.0”

- *error*
  Valor: Un mensaje en lenguaje humano indicando por qué falló la asociación.

- *error_code*
  Valor: “unsuported_type”

- *session_type*
  Valor: (opcional) Un tipo de sesión válido que el OP soporte

- *assoc_type*
  Valor: (opcional) Un tipo de asociación soportado por el OP.

5.4.5.2. check_immediate y checkid_setup

- Descripción: pregunta a un OP si un usuario posee un determinado Claimed Identifier, recibiendo una respuesta inmediata de afirmación o negación.
- Método HTTP: GET o POST.
- Flujo: RP ->User-Agent ->IdP ->User-Agent ->RP

Parámetros de la petición

- *openid.ns*
  Valor: "http://specs.openid.net/auth/2.0”
- *openid.mode*
  Valor: “checkid_immediate” o “checkid_setup”
  Si el Relying Party desea que el usuario pueda interactuar con el OP, “checkid_setup” debería ser usado.

- *openid.claimed_id*
  Valor: (opcional) El Claimed Identifier.
  “openid.claimed_id” y “openid.identity” deberán estar ambos presentes o ambos ausentes. Si ningún valor está presente, la aserción no es relativa a un proceso autenticación y contendrá otra información, haciendo uso de xtensiones.

- *openid.identity*
  Valor: (opcional) El OP-Local Identifier.
  Si un OP-Local Identifier diferente no es especificado, el identificador que se reclama como propio se debe usar como valor de este campo también.
  Nota: Si este campo toma el valor especial "http://specs.openid.net/auth/2.0/identifier_select" entonces el OP debería elegir un identificador que pertenezca al usuario. Este parámetro puede que no aparezca si la petición no está relacionada con un identificador.

- *openid.assoc_handle*
  Valor: (opcional) Un manejador para una asociación entre el Relying Party y el OP que debería ser usado para firmar la respuesta.
  Nota: Si no se envía ningún manejador de asociación, la comunicación utiliza el dumb mode.

- *openid.return_to*
  Valor: (opcional) URL a la que el OP debería redirigir al navegador con la respuesta, indicando el estado de la petición.
  Nota: Si este valor no se envía en la petición significa que el RP no desea que el usuario vuelva a ser redirigido.
  Este campo puede ser utilizado por el RP como un mecanismo para adjuntar información sobre la petición en la propia respuesta.

- *openid.realm*[6]
  Valor: (opcional) Patrón URL que en el que OP debería pedir al usuario que confiara. Este campo es obligatorio si no se especifica un “openid.return_to”.
  Por defecto: URL de return_to .

[6]Un “realm” es un patrón que representa la parte del espacio de la URL para el que la petición de OpenID Authenticacion es válida.

Parámetros de la respuesta

Cuando una petición de autenticación llega a través del navegador en una comunicación indirecta, el OP debería determinar si un usuario final autorizado desea completar la autenticación. Si un usuario final desea terminar la autenticación, el OP debería enviar una aserción positiva al RP.

Los métodos para identificar a los usuarios finales autorizados y obtener la aprobación para devolver la aserción OpenID Authentication quedan fuera del alcance de la especificación.

Si el Relying Party solicitó que la selección del identificador fuera dirigida por el OP, indicando que el valor de “openid.identity” es “http://specs.openid.net/auth/2.0/identifier_select" y hay identificadores para los que el usuario está autorizado, el OP debería permitir al usuario elegir qué identificador quiere usar.

Si el Relying Party proporcionó un manejador de asociación con la petición de autenticación, el OP debe buscar una asociación ligada a ese manejador. Si no la encuentra o ha expirado, el OP debería enviar el parámetro “openid.invalidate_handle” como parte de la respuesta con el valor del parámetro “openid.assoc_handle” de la petición, y proceder como si no se le hubiera especificado ningún manejador.

Si no se especifica ningún manejador, el OP debería usar una asociación privada para firmar la respuesta. El OP debe almacenar esta asociación y responder a peticiones posteriores de comprobación de firma con peticiones directas (verificando directamente con el OP).

Los Relying Parties deberían aceptar y verificar aserciones sobre identificadores para los que no se ha solicitado una autenticación previa. Los OP deberían usar asociaciones privadas para firmar aserciones positivas no solicitadas.

Si el valor de “openid.return_to” no aparece en la petición, esto significa que el Relying Party no desea recibir una aserción de autenticación desde el OP. Este modo puede ser útil cuando se estén usando extensiones para transferir datos desde el RP al OP.

Las aserciones positivas son respuestas indirectas con los siguientes campos:

- *openid.ns*
  Valor: "http://specs.openid.net/auth/2.0”

- *openid.mode*
  Valor: “id_res”

- *openid.op_endpoint*
  La URL del OP.

- *openid.claimed_id*
  Valor: (opcional) El Claimed Identifier.
  “openid.claimed_id” y “openid.identity” deberán estar ambos presentes o ambos ausentes. Si ningún valor está presente, la aserción no es relativa a autenticación y contendrá otra información, haciendo uso de extensiones.

- *openid.identity*
  Valor: (opcional) El OP-Local Identifier.
  Si un OP-Local Identifier diferente no es especificado, el identificador que se reclama como propio se debe usar como valor de este campo también.
  Nota: Si este campo tomó el valor especial "http://specs.openid.net/auth/2.0/identifier_select" en la petición, entonces en la respuesta su valor será el identificador elegido por el usuario a través del OP.

- *openid.assoc_handle*
  Valor: El manejador para la asociación que fue usada para firmar la aserción.

- *openid.invalidate_handle*
  Valor: (opcional) Si el Relying Party ha enviado un manejador de asociación inválido con la petición, debería incluirse aquí.

- *openid.return_to*
  Valor: Copia literal de la URL del parámetro return_to enviado en la petición.

- *openid.response_nonce*
  Valor: Una cadena de 255 caracteres o menos, que debe ser única para esta respuesta satisfactoria de autenticación.

- *openid.signed*
  Valor: lista separada por comas de los campos firmado.

- *openid.sig*
  Valor: firma codificada en base 64.

Parámetros de la respuesta sin éxito

Si el OP no es capaz de identificar al usuario final o bien éste no aprueba, o no puede aprobar, la petición de autenticación, el OP debería mandar una aserción al RP como una respuesta indirecta.

Cuando se recibe una aserción negativa como respuesta a una petición en modo “checkid_immediate”, los RP deberían construir una petición de autenticación nueva usando el modo “checkid_setup”.

Si la petición fue de tipo inmediata, no se le da al usuario la oportunidad de que interactúe con páginas en el OP para que éste le pueda proporcionar credenciales de identifiación o aprobar la petición. Una aserción negativa a esta petición toma la siguiente forma:

- *openid.ns*
  Valor: "http://specs.openid.net/auth/2.0”

- *openid.mode*
  Valor: "setup_needed"

Si la petición no era de tipo inmediata, puesto que el OP puede mostrar páginas al usuario y pedirle credenciales, una respuesta negativa a una petición de este tipo es definitiva. Toma la siguiente forma:

- *openid.ns*
  Valor: "http://specs.openid.net/auth/2.0”

- *openid.mode*
  Valor: "cancel"

5.4.5.3. check_authentication

Si el Relying Party ha almacenado una asociación con el manejador especificado en la aserción, debe comprobar la firma de la respuesta él mismo. Si no tiene una asociación almacenada, debe pedir al OP que verifique la firma con una petición directa, entonces el OP usaría la asociación privada que fue generada cuando se construyó la respuesta positiva.

- Descripción: pregunta a un OP si una respuesta a una petición de autenticación ha sido emitida por él, es decir, es válida.
- Método HTTP: POST.
- Flujo: Relying Party ->Identity Provider ->Relying Party

Parámetros de la petición

- o*penid.mode*
  Valor: "check_authentication"

- Copias exactas de todos los campos recibidos en la respuesta de autenticación, excepto el openid.mode.
  Para verificar la firma el OP sólo debe usar claves privadas y no debe usar asociaciones que tengan claves compartidas. Si la petición de verificación contiene un manejador de una asociación compartida, significa que el RP ya no conoce el secreto compartido, o que una entidad distinta del RP (ej.: un atacante) ha establecido esta asociación con el OP.
  Para prevenir ataques de suplantación de identidad, el OP no debe emitir más de una verificación para cada petición de respuesta de autenticación que ya haya sido enviada. Una respuesta de autenticación y su correspondiente petición de verificación pueden identificarse por sus valores de “openid.response_nonce”.

Parámetros de la respuesta

La respuesta se envía como pares clave-valor en el cuerpo de una respuesta HTTP. Los parámetros que se envían son:

- *ns*
  Valor: "http://specs.openid.net/auth/2.0"

- *is_valid*
  Valor: "true" o "false"; afirma si la firma de la petición de verificación es válida.

- *invalidate_handle*
  Valor: (opcional) El valor "invalidate_handle" enviado en la petición de verificación, si el OP confirma que es inválido.
  Descripción: Si está presente en una respuesta de verificación con "is_valid" a "true", el Relying Party debería borrar la correspondiente asociación y no debería enviar más peticiones con este manejador.
  Este procedimiento para invalidar asociaciones es necesario para prevenir que un atacante pueda invalidar una asociación añadiendo el parámetro "invalidate_handle" a una respuesta de autenticación.

5.4.6. Extensiones

Una extensión a OpenID Authentication es un protocolo que intercambia información adicional en las peticiones y respuestas.

Las extensiones de OpenID se identifican por un Type URI. El Type URI puede usarse como el valor en un elemento <xrd:Type>de un elemento <xrd:Service>en un documento XRDS asociado con un Claimed Identifier.

El Type URI también se puede usar para asociar pares clave-valor en mensajes con una extensión. Para asociar claves y valores en un mensaje con una extensión, la clave debe estar asociada con un Type URI. Para asociar claves con un Type URI, se establece un alias añadiendo una clave prefijada con "openid.ns" y terminada con el texto del alias cuyo valor es el Type URI. Una vez que un alias ha sido establecido, todas las parejas en el mensaje cuyas claves empiecen por "openid." seguido por el

```
<?xml version="1.0" encoding="UTF-8"?>
<xrds:XRDS
   xmlns:xrds="xri://$xrds"
   xmlns:openid="http://openid.net/xmlns/1.0"
   xmlns="xri://$xrd*($v*2.0)">
   <XRD>
       <Service priority="0">
         <Type>http://openid.net/signon/1.0</Type>
         <Type>http://openid.net/sreg/1.0</Type>
         <URI>http://www.myopenid.com/server</URI>
       </Service>
   </XRD>
</xrds:XRDS>
```

Figura 5.6: Ejemplo de un documento XRDS con Type URI para la extensión Simple Registration Extension

alias, y seguido por un punto o por el final de la clave se asocian con la extensión.

Véase el siguiente ejemplo en el que se reflejan algunos de los atributos que pueden aparecen en un mensaje del protocolo OpenID que hace uso de la extensión Attribute Exchange 1.0:

```
//Algunos atributos de una petición OpenID que usa la extensión AX
/*Type URI*/
openid.ns.ax=http://openid.net/srv/ax/1.0
openid.ax.mode=fetch_request

/*Type URIs*/
openid.ax.type.fname=http://example.com/schema/fullname
openid.ax.type.gender=http://example.com/schema/gender
openid.ax.type.fav_dog=http://example.com/schema/favourite_dog openid.ax.type.fav_movie=
     ...http://example.com/schema/favourite_movie
/*End of type URIs*/

openid.ax.count.fav_movie=3
openid.ax.required=fname,gender
openid.ax.if_available=fav_dog,fav_movie
```

Figura 5.7: Type URI para asociar pares clave-valor en el protocolo Attribute Exchange 1.0

Puede verse un ejemplo detallado del uso de una extensión de OpenID en el Anexo 8.1 Ejemplo de uso de PAPOID.

5.4.6.1. OpenID Simple Registration Extension 1.0

OpenID Simple Registration es una extensión al protocolo OpenID Authentication que permite un intercambio de perfiles de usuario a un bajo coste. Está diseñado para pasar ocho campos de información que suelen pedirse cuando un usuario quiere registrarse en una nueva cuenta con un servicio web.

Formato de la petición

Los parámetros de la petición que se especifican deberían ser enviados con una petición OpenID en modo checkid_immediate o checkid_setup.

Todos los campos de la petición que se especifican son opcionales, aunque al menos u "openid.sreg.required" u
"openid.sreg.optional" deben aparecer.

- *openid.sreg.required*: Lista separada con comas de nombres de atributos. Si no aparecen en la respuesta, impedirán que el Reying Party pueda completar el registro sin la interacción del usuario. Los posibles campos son los que se especifican en el apartado dedicado al formato de la respuesta, pero sin el prefijo "openid.sreg."

- *openid.sreg.optional*: Lista separada con comas de nombres de atributos que serán usados por el Relying Party. Si no aparecen en la respuesta, no impedirá que se complete el registro. Los posibles campos son los que se especifican en el apartado dedicado al formato de la respuesta, pero sin el prefijo "openid.sreg."

- *openid.sreg.policy_url*: Una URL que provee el Relying Party con información sobre el uso que se le dará a los datos obtenidos haciendo uso de SREG. El OP debería mostrar esta URL al usuario si se le proporciona.

Formato de la respuesta

Los campos que se especifican a continuación deberían incluirse en la respuesta del OP cuando "openid.mode" vale "id_res".

Un Identity Provider puede devolver cualquier subconjunto de los campos que siguen en la respuesta. En concreto:

- *openid.sreg.nickname*: Nick que el usuario quiere usar.
- *openid.sreg.email*: Correo electrónico del usuario.
- *openid.sreg.fullname*: Nombre completo del usuario.
- *openid.sreg.dob*: Fecha de nacimiento del usuario.
- *openid.sreg.gender*: Género del usuario.
- *openid.sreg.postcode*: Código postal del usuario.
- o*penid.sreg.country*: País del usuario.
- *openid.sreg.language*: Idioma del usuario.
- *openid.sreg.timezone*: Zona horaria.

5.4.7. Ejemplo

En este apartado se muestra un ejemplo sencillo del modo de funcionamiento statefull y stateless del protocolo OpenID Authentication 2.0 para comprender mejor cómo funciona el protocolo. El flujo de mensajes aquí expuesto supone que el proceso de autenticación del usuario se lleva a cabo de manera satisfactoria y sin que se produzca ninguna incidencia.

Supongamos que Alicia usa su navegador para dejar un comentario en el blog http://blog.example.com. Este blog exige a los usuarios que se autentiquen previamente con OpenID y Alicia tiene una identidad OpenID, http://alicia.op.example.com, que será la que utilice a tal fin.

El primer paso es rellenar el formulario que muestra el blog:

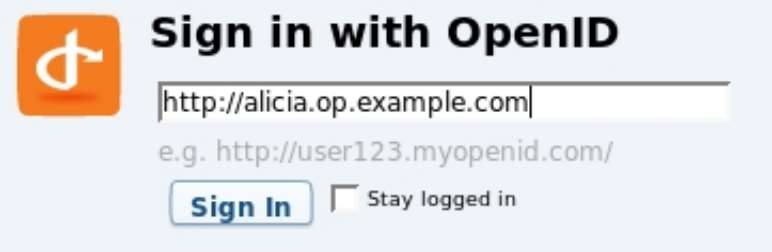

Figura 5.8: Formulario de login con OpenID

Con esta información el blog o Relying Party usa el protocolo Discovery para acceder al siguiente documento HTML:

```
1 <html>
2    <head>
3         <link rel="openid2.provider" href="http://op.example.com/openid-server.cgi"/>
4    </head>
5    <body></body>
6 </html>
```

Figura 5.9: Ejemplo de un documento HTML usado en un OpenID Identifier

5.4.7.1. Stateless o Dumb mode

El intercambio de mensajes que se produce cuando el protocolo OpenID utiliza el modo de funcionamiento Stateless sería el que se describe en esta sección.

Lo primero que hace el blog, o Relying Party, es redirigir el navegador al OpenID Provider con la siguiente petición GET o POST

```
http://op.example.com/openid-server.cgi?
         openid.ns = http://specs.openid.net/auth/2.0&
         openid.mode = checkid_setup&
         openid.identity = http://alicia.op.example.com&
         openid.claimed_id = http://alicia.op.example.com&
         openid.realm = http://blog.example.com&
         openid.return_to = http://blog.example.com/coment.cgi?session_id=Alicia
```

Figura 5.10: Ejemplo de petición OpenID Authentication en el modo checkid_setup

Ahora el OpenID Provider verifica que Alicia es quien dice ser, con un par "usuario/contraseña" por ejemplo, y redirigie el navegador a la URL especificada en el parámetro "openid.return_to" de la petición:

```
http://blog.example.com/coment.cgi?session_id=Alicia&
       openid.ns = http://specs.openid.net/auth/2.0&
       openid.mode = id_res&
       openid.identity = http:/alicia.op.example.com&
       openid.claimed_id = http://alicia.op.example.com&
       openid.op_endpoint = http://blog.example.com&
       openid.return_to = http://blog.example.com/coment.cgi?session_id=Alicia&
       openid.signed = mode, identity, return_to&
       openid.assoc_handle = *opaque handle*&
       openid.sig = *firma HMAC en base 64*&
       openid.response_nonce = *unico para cada respuesta?op_endpoint*
```

Figura 5.11: Ejemplo de respuesta OpenID Authentication al modo checkid_setup

Cuando esta respuesta se recibe en el blog, a priori, éste no tiene manera de comprobar que la firma es correcta, por lo que envía una petición POST directamente al OP.

```
http://op.example.com/openid-server.cgi?
       openid.ns = http://specs.openid.net/auth/2.0&
       openid.mode = check_authentication&
       openid.signed = mode,identity,return_to&
       openid.assoc_handle = *opaque handle*&
       openid.sig = *firma HMAC en base 64*&
       openid.responce_nonce = 2008-03-15T16& resto_campos
```

Figura 5.12: Ejemplo de petición OpenID Authentication en el modo check_authentication

Ahora el OP comprueba que la petición es correcta y le envía la siguiente respuesta HTTP a http://blog.example.com?session_id=Alicia:

```
<html>
     <head></head>
     <body>
          ns: http://specs.openid.net/auth/2.0
          is_valid: true
     </body>
</html>
```

Figura 5.13: Ejemplo de respuesta OpenID Authentication al modo check_authentication

5.4.7.2. Statefull o Smart mode

Si el blog hubiese optado por utilizar el modo Statefull, lo primero que habría hecho es solicitar un secreto compartido al OP, y para ello le envía la siguiente petición POST:

```
http://op.example.com/openid-server.cgi?
      openid.ns = http://specs.openid.net/auth/2.0&
      openid.mode = associate&
      openid.assoc_type = HMAC-SHA256&
      openid.session_type = no-encryption
```

Figura 5.14: Ejemplo de petición OpenID Authentication en el modo associate

El Provider genera una clave compartida para el método de firma HMAC-SHA1 y se la envía al RP sin cifrar.

Si nada falla, el Relying Party envía al OpenID Provider un mensaje checkid_setup o checkid_immediate con el mismo formato que los anteriores. La única diferencia será que en el mensaje se especificará el valor de openid.assoc_handle para pedirle al OP que utilice la clave asociada a ese manejador para firmar la respuesta. Ahora cuando el RP recibe la respuesta no se genera una petición check_authentication porque es el propio Relying Party el que verifica la firma

```
<html>
    <head></head>
    <body>
        ns: http://specs.openid.net/auth/2.0
        assoc_handle: *handle opaco*
        assoc_type: *el mismo del request*
        session_type: *el mismo del request*
        expires_in: *tiempo en segundos*
        mac_key: *clave en texto plano*
    </body>
</html>
```

Figura 5.15: Ejemplo de respuesta OpenID Authentication en el modo associate

Capítulo 6

PAPOID

6.1. Introducción

Una vez explicadas las tecnologías utilizadas en este proyecto, se puede tener una idea más clara de los conceptos que permitirán conocer más a fondo cuál es el objetivo y la estructura del mismo.

El objetivo es desarrollar un perfil para el sistema de autenticación y autorización PAPI que soporte el protocolo OpenID Authentication versión 1.1 y 2.0, junto con la extensión Simple Registration Extension. Así es como surge PAPOID.

En este capítulo se explica cómo funciona PAPOID y cómo se enmarca en el contexto del protocolo de OpenID. Posteriormente se explica de una manera más detallada cuáles son los bloques básicos que componen la aplicación y cómo se conectan entre sí, algo fundamental para comprender mejor el proceso de instalación.

En último lugar, se detallan los requerimientos necesarios para instalar y configurar el Servidor PAPI-OpenID. Además se especifica el flujo básico de operaciones que tiene lugar en PAPOID antes de que éste responda al Relying Party.

6.2. Funcionamiento de PAPOID

La aplicación web PAPOID es un servidor de identidades OpenID, pero con algunas particularidades que lo hacen especial. Desde un punto de vista general, PAPOID se encarga básicamente de responder a las peticiones de autenticación de un usuario por parte de un Relying Party para lo que

será necesario que el usuario se autentique con PAPI. Además, durante este proceso puede que sea necesario rellenar un formulario para completar el proceso de autenticación si se está haciendo uso de la extensión Simple Registration Extensión.

Este sería el funcionamiento de PAPOID en detalle:

1. El usuario dispone de un proveedor de identidad PAPI sobre el que se sustenta PAPOID.
2. El usuario visita el sitio web o Relying Party que utiliza el protocolo OpenID para autenticar a sus usuarios.
3. El Relying Party muestra el formulario de Login de OpenID al usuario.
4. El usuario introduce su identidad de OpenID (en el apartado posterior de instalación y configuración se verá en detalle cómo se construye exactamente esta identidad de OpenID a partir de la cuenta que el usuario tiene en el proveedor de identidad PAPI).
5. El Relying Party aplica *Discovery* sobre esta URL para obtener la dirección del servidor a partir del fichero _index.html, o _ident_select.html si se está utilizando la versión 2.0 del protocolo OpenID con la opción de selección del identificador del usuario en el lado del Provider.
6. De manera opcional, el Relying Party puede establecer una *asociación* con el proveedor de identidad que proporciona PAPOID, para funcionar en modo *smart mode*.
7. El Relying Party envía una *petición de autenticación* al OpenID Porvider.
8. El Provider está protegido con PAPI, por lo que para poder continuar con el proceso de autenticación el usuario tendrá que *identificarse en su proveedor de identidad PAPI* . Este paso se hace la primera vez que se intenta acceder al Provider.
9. Si el protocolo OpenID se está usando son la extensión *Simple Registration Extensio*n con datos requeridos, entonces:
   a) El Provider muestra al usuario un formulario para que introduzca los datos que el Relying Party solicita.
   b) Cuando el usuario confirma los datos del usuario, éstos son procesados en _process_sreg.php y enviados de nuevo al Provider.
10. En este momento, el servidor de identidad ya tienen toda la información que necesita para poder *responder a la petición de autenticación* y así lo hace.

11. En el modo dumb mode, se establece una comunicación directa entre el Relying Party y el Provider para que éste primero compruebe que la respuesta recibida es realmente válida.

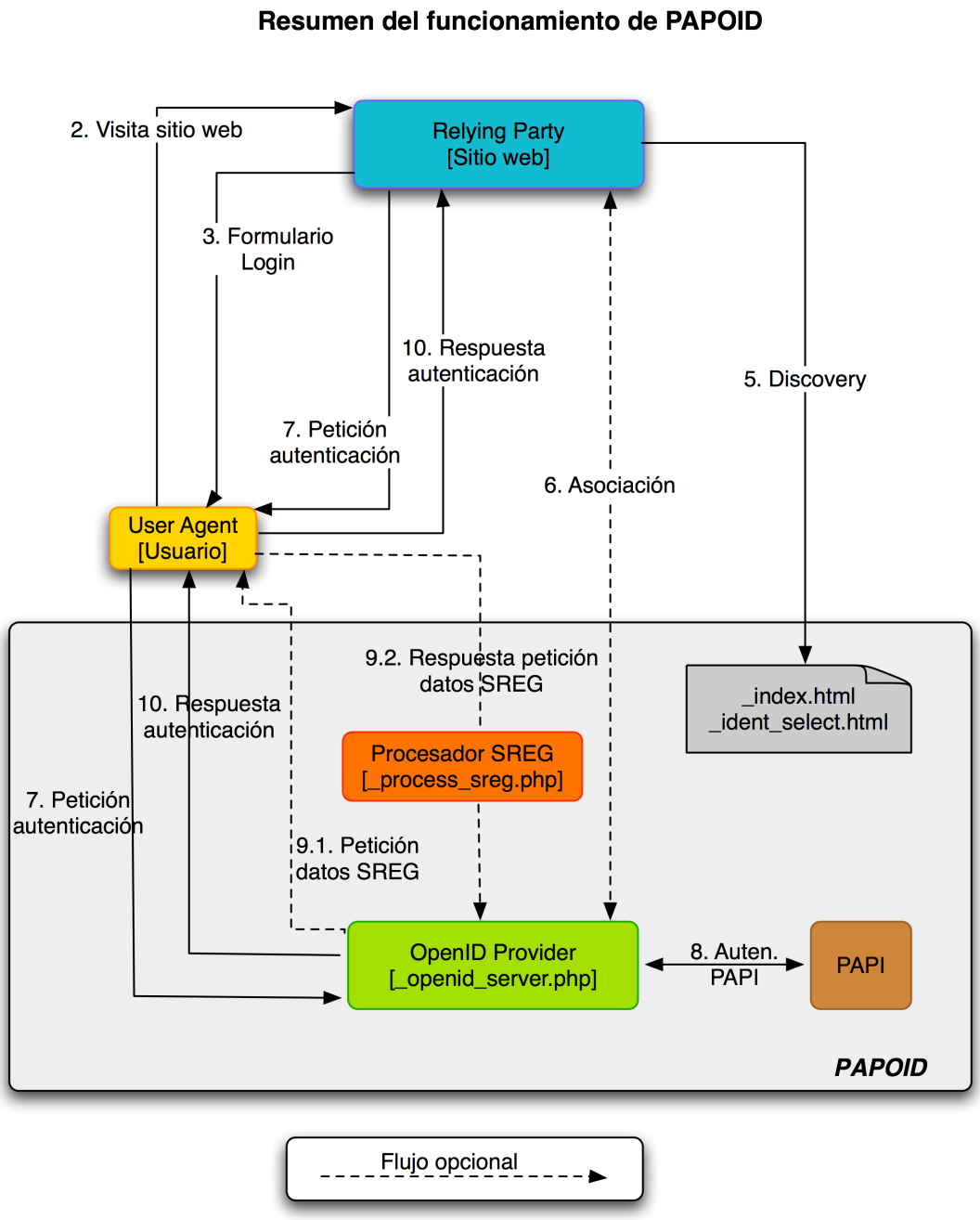

Figura 6.1: Integración de PAPOID en el protocolo OpenID Authenticacion 2.0

6.3. Estructura de PAPOID

PAPOID ha sido diseñado para poder proporcionar la funcionalidad descrita en el apartado anterior de manera que sea lo más transparente posible al usuario.

En la figura 6.2 podemos ver cómo se organiza en grandes bloques el Servidor PAPI-OpenID. Se van a explicar cada uno de ellos más en detalle para tener una visión global de la estructura de la aplicacion:

Servidor de OpenID Es el proveedor de identidades propiamente dicho. Está protegido con PAPI y se encarga de manejar las peticiones de OpenID que recibe y de responderlas. Se corresponde con el archivo *_openid_server.php*.

JanRain OpenID PHP Library La libería PHP OpenID de JanRain permite habilitar la autenticación OpenID en sitios construidos usando PHP. En la actualidad cubre los requisitos de un consumidor de OpenID, de un servidor de OpenID e implementaciones de almacenamiento.

Funciones auxiliares En el archivo *OpenIDServer.php* se recogen algunas funciones auxiliares que son necesarias durante el manejo de peticiones y respuestas por parte del Provider.

Procesador SREG El archivo *_process_sreg.php* es el encargado de tratar los datos del formulario que se muestra al usuario cuando la petición de autenticación usa la extensión Simple Registration Extension.

Archivo de configuración La configuración del servidor PAPOID está recogida en el archivo *papoid.ini*.

Página de error PAPOID está preparado para redireccionar el navegador del usuario a una página de error en caso de que la petición que reciba el Servidor de OpenID no sea una petición del protocolo OpenID válida. Esta página de error se corresponde con el fichero *_NotOpenIDRequest.php*.

Identificadores Los archivos *_ident_select.html* e *_index.html* estarán situados en las URLs a las que se hace fetch con el protocolo Discovery al comienzo del protocolo. Estos archivos contienen las etiquetas HTML necesarias para que el Relying Party pueda localizar al Provider.

- **_ident_select.html** Se utiliza con la versión 2.0 del protocolo OpenID en conjunción con la opción de que el identificador de usuario se seleccione en el Provider.
- **_index.html** Se utiliza con el resto de posibles identificadores de usuario.

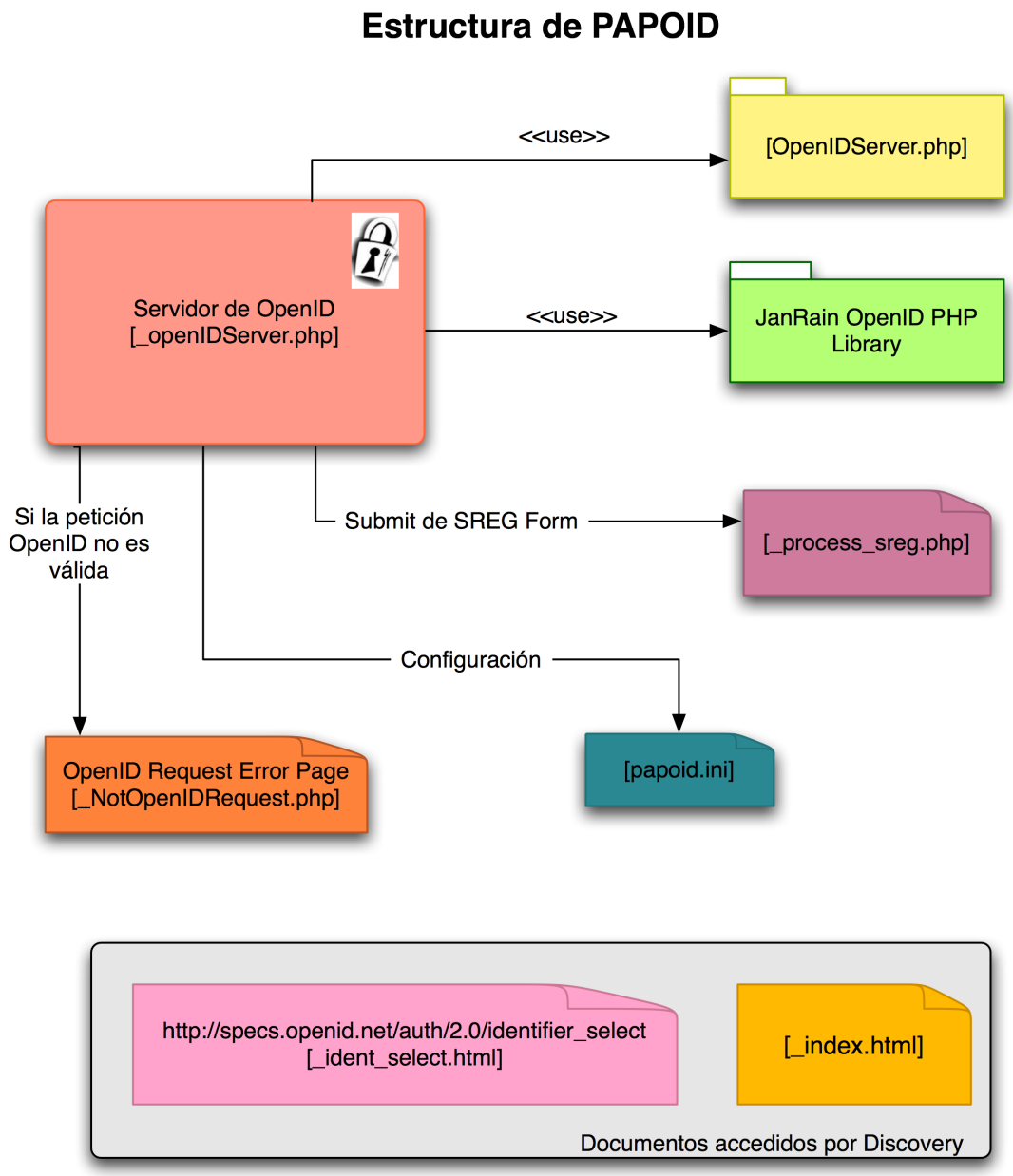

Figura 6.2: Estructura interna de PAPOID

6.4. Requerimientos de la aplicación

Los requerimientos necesarios para utilizar PAPOID son los siguientes:

- Servidor de PHP
- Módulo PHP5
- phpPoA
- Librería JanRain PHP OpenID.

En lo que sigue se describe cada uno de estos requisitos detalladamente y la forma de configurarlos para poder utilizar el Servidor PAPI-OpenID.

Ha de notarse que el entorno aquí propuesto para el despliegue de la aplicación es el proporcionado por el servidor Apache, pero PAPOID podría instalarse en cualquier otro servidor que soporte PHP.

6.4.1. Servidor de PHP

Es necesario disponer de un servidor web que soporte PHP5, como puede ser Apache, Cadium o los servidores de Sun, iPlanet o Netscape, entre otros. En concreto, este proyecto ha sido desarrollado utilizando Apache, por lo que todas las anotaciones y ejemplos que se expongan se referirán a él.

Para instalar un servidor Apache, hay que bajarse la aplicación de la página *http://httpd.apache.org/* y seguir los pasos de instalación que allí se exponen. Un requisito indispensable para que la aplicación funcione es tener instalados los módulos de PHP5.

6.4.2. Módulo PHP5

PHP (acrónimo de 'PHP: Hypertext Pre-processor') es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Se trata de un lenguaje interpretado, esto es, un lenguaje que se ejecuta por medio de un intérprete, en lugar de ser compilado.

Su interpretación y ejecución se da en el servidor web, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al

servidor para que le envíe una página web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, regresando el resultado al servidor, y éste al cliente.

Puede obtener el módulo de PHP5 en *http://php.net/*.

6.4.3. phpPoA

phpPoA implementa las principales características de un punto de acceso PAPI en PHP. Esto implica que no es necesario incluir su configuración en el archivo de configuración del servidor web: sólo hay que llamar a los métodos del PoA desde cualquier página web que deseemos proteger. Al principio de la página hay que incluir el código PHP necesario para llamar a los métodos del PoA que manejan la autenticación. Es muy importante incluir este código al principio de la página para asegurar que no se muestra ningún dato de la misma antes de la correcta autenticación del usuario. Por ejemplo:

```
1 <?php
2     include 'PoA.php';
3     $poa = new autoPoA('admin');
4     $userData = $poa->check_Access();
5 ?>
6 <html>
7     //Web page.
8 </html>
```

Figura 6.3: Página web protegida con PAPI

La configuración de control de acceso para cualquier sitio web protegido con phpPoA está definido en el archivo de configuración phpPoA.ini. En este archivo se pueden establecer filtros, páginas web de error, logs, URLs clave, etc. Puesto que phpPoA implementa un PoA simple, hay que establecer también un GPoA o AS.

El phpPoA tiene dos modos de operación:

1. **Redirección automática**
   En este modo el PoA redirecciona automáticamente al usuario a una página de error cuando la autorización falla o cuando hay algún error del sistema. Esta funcionalidad está implementada en la clase autoPoA. Este es el modo de operación que se ha utilizado en PAPOID.

2. **Autenticación simple**
   En este modo de funcionamiento la página web protegida tiene el control y decide qué hacer cuando el PoA devuelve la validación. Si el usuario está autorizado en cualquier modo, el phpPoA devuelve un código positivo. Esta funcionalidad está implementada en la clase PoA.

Estos componentes proporcionan una manera muy sencilla de proteger un sitio web con un mínimo coste de configuración e instalación. Simplemente incluyendo tres líneas en cualquier página web (e inicializando el archivo de configuración apropiadamente), puede asegurarse que ningún usuario no autorizado accederá a dichos recursos.

Toda la información necesaria para instalar phpPoA puede encontrarse en *http://papi.rediris.es/*.

6.4.4. Librería JanRain PHP OpenID

Esta librería de OpenID para PHP incluye:

- Un servidor de OpenID.
- Un consumidor de OpenID.
- Almacenamientos para MySQL, PostgreSQL, SQLite, y sistemas de archivos para el almacenamiento de datos de OpenID.
- Un ejemplo de servidor y consumidor.

Durante el desarrollo de PAPOID los módulos que se han utilizado principalmente han sido:

- **Auth/OpenID.php**
  Este módulo contiene el núcleo de la funcionalidad usada por la librería.
- **Auth/OpenID/Server.php**
  Un servidor de OpenID debe llevar a cabo tres tareas:

  1. Examinar la petición que llega para determinar su naturaleza y valided.
  2. Tomar una decisión sobre cómo responder a esta petición.
  3. Dar formato a la respuesta de acuerdo al protocolo.

La primera y la última de estas tareas pueden llevarse a cabo con los métodos *Auth_OpenID_Server::decodeRequest()* y *Auth_OpenID_Server::encodeResponse()*.
Si es una petición para autenticar a un usuario (una petición 'checkid_setup' o 'checkid_immediate'), se necesita decidir si se reconoce al usuario en cuestión o no. Se examinan las propiedades del objeto *Auth_OpenID_CheckIDRequest*, y cuando se haya tomado una decisión se responde con *Auth_OpenID_CheckIDRequest::answer()*.
Auth_OpenID_Server contiene toda la lógica y datos necesarios para responder a otros tipos de asociación, relacionados con el establecimiento de asociación entre cliente y servidor o la verificación de la autenticidad de comunicaciones previas. Simplemente hay que pasárselas a *Auth_OpenID_Server::handleRequest()*.
Puesto que las extensiones no varian el modo en que la autenticación OpenID funciona, el código para manejar las extensiones puede estar completamente separado de la clase Auth_OpenID_Request.
Sin embargo, puede desearse enviar los datos que se envíen de vuelta estén firmados. Auth_OpenIDResponse proporciona métodos que permiten añadir datos que al final se firmen.

- **Auth/OpenID/FileStore.php**
  Este archivo proporciona un método de almacenamiento en memoria cache para los servidores y consumidores de OpenID.

- **Auth/OpenID/SReg.php**
  En este archivo se ofrece la representación en objetos y el parseado de respuestas y peticiones que utilizan la extensión Simple Registration Extension.
  Este módulo contiene objetos que representan las peticiones y respuestas de la extensión Simple Registration Extension. El funcionamiento de este módulo sería como sigue:

  1. El Relying Party crea un objeto de petición y lo añade al objeto *Auth_OpenID_AuthRequest* antes de hacer la petición de autenticación al Provider.
  2. El OpenID provider extrae la petición de registro simple de la petición de OpenID usando *Auth_OpenID_SRegRequest::fromOpenIDRequest*, obtiene la aprobación y los datos del usuario, y crea un objeto *Auth_OpenID_SRegResponse* que añade a la respuesta:

```
1 $sreg_req = Auth_OpenID_SRegRequest::fromOpenIDRequest($checkid_request);
2 // [ get the user's approval and data, informing the user that
3 // the fields in sreg_response were requested ]
4 $sreg_resp = Auth_OpenID_SRegResponse::extractResponse($sreg_req, $user_data);
5 $sreg_resp->toMessage($openid_response->fields);
```

Figura 6.4: Manejo de SREG en el Provider con la librería JanRain PHP OpenID

3. El Relying Party usa *Auth_OpenID_SRegResponse::fromSuccessResponse* para obtener los datos de la respuesta de OpenID:

```
1 $sreg_resp = Auth_OpenID_SRegResponse::fromSuccessResponse($success_response);
```

Figura 6.5: Manejo de SREG en el Relying Party con la librería JanRain PHP OpenID

Descarga la librería JanRain PHP OpenID desde el sitio web *http://www.openidenabled.com/php-openid/* y sigue las instrucciones de instalación que vienen con la distribución.

6.5. Instalación y configuración

6.5.1. Apache

Una vez que hemos instalado los requisitos necesarios para configurar PAPOID, debemos configurar un virtual host en algun servidor web con soporte para PHP5. A continuación se muestra un ejemplo de cómo sería esta configuración para Apache2:

El segundo AliasMatch es opcional, sólo se necesita si PAPOID se va a utilizar con OpenID versión 2 y se va a delegar en el OpenID Provider la elección de la URL identificadora del usuario.

```
<VirtualHost *>
    ServerName me.example.org
    DocumentRoot "/home/spool/openid/"
    AliasMatch ^/[^_] /home/spool/openid/_index.html
    #This AliasMatch is optional
    AliasMatch ^/$ /home/spool/openid/www/_ident_select.html
    AddType application/x-httpd-php .php

    <Location "/">
        Options Indexes FollowSymLinks
        Order Deny,Allow
        Allow from all
    </Location>
</VirtualHost>
```

Figura 6.6: Configuración del virtual host en Apache2

6.5.2. PAPOID

PAPOID se distribuye en un paquete con licencia GPL en el que se incluyen los siguientes archivos:

- papoid.ini: Archivo de configuración de PAPOID.
- _index.html.
- _ident_select.html.
- _NotOpenIDRequest.html
- _sreg_form.txt:.
- OpenIDServer.php:.
- _openid_server.php.
- _process_sreg.php.
- PAPOID_doc.pdf: Documentación en inglés de PAPOID.
- LICENSE: Licencia de PAPOID.

Para instalar PAPOID deben seguirse los siguientes pasos:

1. Copiar el fichero OpenIDServer.php en el include path de PHP.

2. Copiar los archivos _index.html, _ident_select.html, _openid_server.php, _process_sreg.php, _sreg_form.txt y _NotOpenIDRequest.html en el documento raíz del virtual host que acaba de configurar.

3. PHP debe conocer la ruta a las funciones auxiliares que utiliza el OpenID Provider, es decir, a OpenIDServer.php y a la librería JanRain PHP, para ello una directiva similar a la siguiente debe ser incluida en el fichero de configuración de PHP (php.ini).

```
1 include\_path =.:/usr/local/php/lib/php:PAPOID_directory
```

Figura 6.7: Configuración del include_path en php.ini

4. Para que PHP pueda encontrar el archivo papoid.ini un nuevo parámetro llamado PAPOID_ini_file tiene que ser incluido en php.ini con la URL absoluta a papoid.ini.

```
1 PAPOID_ini_file = /usr/local/php/lib/php/papoid.ini
```

Figura 6.8: Configuración de PAPOID_ini_file en php.ini

5. Crear un directorio con permisos de escritura para el usuario que esté ejecutando el servidor web, y que será utilizado por el servidor para almacenar la información que necesite manejar.

6. En _index.html configurar la dirección del servidor de openid:

```
1  <html>
2     <head>
3          <!-- Used in OpenID Authentication version 2 -->
4          <link rel="openid2.provider" href="http://me.example.com/_openid_server.php">
5
6          <!-- Used in OpenID Authentication version 1 -->
7          <link rel="openid.server" href="http://me.example.com/_openid_server.php">
8     </head>
9     <body></body>
10 </html>
```

Figura 6.9: Configuración de _index.html

7. En caso de que vaya a utilizar OpenID Authentication con la opción “http://specs.openid.net/auth/2.0/identifier_select", configurar también la dirección del OpenID Provider en _ident_select.html:

```
<html>
   <head>
        <link rel="openid2.provider" href="http://me.example.com/_openid_server.php">
        <link rel="openid2.local_id" href="http://specs.openid.net/auth/2.0/
             ...identifier_select" />
   </head>
   <body></body>
</html>
```

Figura 6.10: Configuración de _ident_select.html

8. En el archivo phpPoA.ini que contiene la configuración del PoA que protege PAPOID, añadir la siguiente línea a la sección local donde se encuentra definido dicho PoA:

```
Pass_Pattern = check_authentication&associate
```

Figura 6.11: Configuración del Pass_Pattern en phpPoA.ini

6.5.2.1. Configuración

papoid.ini es el archivo de configuración de PAPOID, definido en formato PHP ini y compuesto por varias secciones y variables. A continuación se explica cada una de estas secciones en detalle, junto con las particularidades de cada una de ellas.

Para facilitar la comprensión de cada una de las posibles variables se adjuntan varias listas con el nombre de las mismas, su descripción y un ejemplo de los valores que pueden tomar.

PAPOID_Main

En la sección PAPOID_Main están definidas las variables generales de configuración de este servidor PAPI OpenID.

Hay que hacer especial mención a las siguientes variables:

- **Unwanted**: Lista separada con comas de sitios web a los que el usuario no puede conectarse usando PAPOID. Un sitio web puede formar parte de esta lista si se piensa que puede ser mal intencionado o simplemente no se desea darle soporte OpenID por cualquier otro motivo.

Cuando el OpenID Provider recibe una petición comprueba si alguno de los sitios de la lista coincide con el atributo openid.return_to y si lo hace no se valida la identidad OpenID del usuario.
Por ejemplo, imaginemos que el sitio www.site1.org está intentando atacar el servidor, entonces podríamos añadir su nombre a la lista de la variable Unwanted:

```
1 Unwanted = www.foo.org, www.bar.org, www.site1.com
```

Figura 6.12: Configuración de la variable Unwanted en papoid.ini

Ahora el proceso de autenticación en www.site1.org usando OpenID Authentication con este servidor fallará, pues el usuario no será reconocido.

- **OIDBaseIdentifier, OIDPersonalIdentifier, PAPIAttributes**: Estas variables permiten determinar cómo se construirán las identidades OpenID que este Provider va a reconocer como válidas. Imagínese que se tiene la siguiente configuración en el archivo papoid.ini:

```
1 ...
2 OIDBaseIdentifier = http://me.example.org/
3 OIDPersonalIdentifier = $attr1/$attr2
4 PAPI_Attributes = $attr1:uid, $attr2:uid
5 ...
```

Figura 6.13: Configuración de OIDBaseIdentifier, OIDPersonalIdentifier y PAPIAttributes en papoid.ini

donde se tiene que:

- OIDBaseIdentifier es el nombre del virtual host que se ha configurado en Apache. URI base de PAPOID. Asegurarse de que incluye el símbolo '/' del final.
- OIDPersonalIdentifier es la cadena que habrá que concatenar a OIDBaseIdentifier después de tratarla. Se trata de una expresión utilizada para identificar a los diferentes usuarios de PAPOID. No puede aparecer el carácter '#'. Se recomienda que el nombre de las partes que se van a reemplazar se construyan como '$attr' + i, donde i es un índice numérico no utilizado previamente.

- PAPI_Attributes es una lista de las partes que van a ser reemplazadas en OIDPersonalIdentifier y sus correspondientes atributos de la aserción PAPI. El formato es clave:valor, y la lista de elementos se separa con comas.

Ahora para construir el identificador de OpenID habrá que sustituir '$attr1' y '$attr2' por el valor del atributo 'uid' en '$attr1/$attr2' y concatenarlo a OIDBaseIdentifier.
Supongamos que el 'uid' del usuario que desea autenticarse es 'test.user', entonces su identificador sería 'http://me.example.org/test.user/test.user'.
Si el Relying Party soporta OpenID 2 donde la selección del identificador se lleva a cabo en el OP, el usuario también podría autenticarse con la URL 'http://me.example.org/, aunque al final del proceso el Relying Party identificará al usuario como 'http://me.example.org/test.user/test.user'.

- **PAPOID_Directory**
  Dirección completa del directorio en el que el servidor va a almacenar la información que necesita.
  Ejemplo: */home/spool/openid/OpenID_directory*

- **PoA_Name**
  Nombre de la sección local donde está definido el PoA que protege PAPOID en el archivo de configuración phpPoA.ini.
  Ejemplo: *me.example*

- **OIDRequest_Error_File**
  Página de error mostrada cuando la petición de OpenID recibida no es válida.
  Ejemplo: *_NotOpenIDRequest.html*

- **OIDServer**
  URI de _openid_server.php
  Ejemplo: *http://me.example.org/_openid_server.php*

SREG_Extension

Esta sección es un mapeo entre los atributos de una aserción PAPI y los atributos de la extensión Simple Registration Extension (nickname, fullname, email, dob, gender, poscode, country, language, timezone). Hay dos tipos de variables:

- *sreg_(campo)_name*: Este es el nombre que se le da al campo en el formulario que se le muestra al usuario cuando hay campos requeridos en la petición de OpenID. Estas variables son obligatorias.

- *sreg_(campo)_value*: Indica el atributo de la aserción PAPI que se va a usar como valor predeterminado del campo requerido en la petición de OpenID. Estas variables son opcionales.

A continuación se muestra una tabla resumen con todas las variables que aparecen en esta sección. Todas ellas son obligatorias a menos que se indique lo contrario.

- **sreg_nickname_name**
  Nombre dado al campo 'nickname' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *nick*

- **sreg_email_name**
  Nombre dado al campo 'email' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Correo electrónico*

- **sreg_fullname_name**
  Nombre dado al campo 'nickname' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Nombre completo*

- **sreg_dob_name**
  Nombre dado al campo 'dob' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Fecha de nacimiento*

- **sreg_gender_name**
  Nombre dado al campo 'gender' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Género*

- **sreg_postcode_name**
  Nombre dado al campo 'poscode' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Código postal*

- **sreg_country_name**
  Nombre dado al campo 'country' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *País*

- **sreg_language_name**
  Nombre dado al campo 'language' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Idioma*

- **sreg_timezone_name**
  Nombre dado al campo 'timezone' en el formulario que se le muestra al usuario cuando se solicita en la petición de OpenID.
  Ejemplo: *Zona horaria*

- **sreg_nickname_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'nickname' del usuario.
  Ejemplo: *uid*

- **sreg_email_values**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'email' del usuario.
  Ejemplo: *mail*

- **sreg_fullname_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'fullname' del usuario.
  Ejemplo: *uid*

- **sreg_dob_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'dob' del usuario.

- **sreg_gender_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'gender' del usuario.

- **sreg_postcode_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'poscode' del usuario.

- **sreg_country_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'country' del usuario.

- **sreg_language_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'language' del usuario.

- **sreg_timezone_value**
  Atributo de la aserción PAPI usado como el valor predefinido para el 'timezone' del usuario.

Site Extension

Por otro lado, para cada página que se desee es posible definir una sección en la que se personalicen la propiedades de Simple Registration Extension. En cada sección es posible redefinir las variables explicadas para la sección SREG_Extension.

Además podemos restringir la información que queremos proporcionarles, es decir, imagínese que en la sección SREG_Extension se ha indicado que la variable "sreg_nickname_value" tome el valor del atributo uid, pero no se quiere proporcionar esta información a www.example_site.org, entonces lo que habría que hacer sería añadir la variable nickname, sin importar el valor que se le dé:

```
1 [www.example\_site.org]
2 nickname = whatever
```

Figura 6.14: Restricción de información para un sitio web en papoid.ini

Véase la siguiente lista resumen con todas las variables que aparecen en esta sección. Todas ellas son obligatorias a menos que se indique lo contrario.

- **sreg_*_name**
  Ver SREG_Extension para saber más a cerca de estas variables. La única diferencia es que, en este caso, estas variables son opcionales.

- **sreg_*_value**
  Ver SREG_Extension.

- **nickname**
  No toma el valor del 'nickname' de los atributos de la aserción PAPI. Opcional.

- **email**
  No toma el valor del 'email' de los atributos de la aserción PAPI. Opcional.

- **fullname**
  No toma el valor del 'fullname' de los atributos de la aserción PAPI. Opcional.

- **dob**
  No toma el valor del 'dob' de los atributos de la aserción PAPI. Opcional.
- **gender**
  No toma el valor del 'gender' de los atributos de la aserción PAPI. Opcional.
- **postcode**
  No toma el valor del 'postcode' de los atributos de la aserción PAPI. Opcional.
- **country**
  No toma el valor del 'country' de los atributos de la aserción PAPI. Opcional.
- **language**
  No toma el valor del 'language' de los atributos de la aserción PAPI. Opcional.
- **timezone**
  No toma el valor del 'timezone' de los atributos de la aserción PAPI. Opcional.

Para facilitar la comprensión de este archivo de configuración, véase la figura 6.15 donde se muestra un ejemplo de cómo quedaría un archivo de configuración de PAPOID:

```
[PAPOID_Main]
PAPOID_Directory = /home/spool/openid/OpenID_directory
PAPI_Attribute = uid
Unwanted = www.site1.com, www.site2.com
PoA_Name = me.example
OIDRequest_Error_File = _NotOpenIDRequest.html
OIDServer = me.example.org/_openid_server.php
OIDBaseIdentifier = http://me.example.org/
OIDPersonalIdentifier = $attr1/attr2
PAPI_Attributes = $attr1:uid, $attr2:uid


[SREG_Extension]
sreg_nickname_name = nick
sreg_email_name = correo
sreg_fullname_name = fullname
sreg_dob_name = date of birthday
sreg_gender_name = gender
sreg_postcode_name = postcode
sreg_country_name = country
sreg_language_name = language
sreg_timezone_name = timezone


sreg_nickname_value = uid
sreg_email_value = mail
;; sreg_fullname_value
;; sreg_dob_value
;; sreg_gender_value
;; sreg_postcode_value
;; sreg_country_value
;; sreg_language_value
;; sreg_timezone_value

[www.example_site.com]
sreg_email_name = correo electronico
sreg_email_value = uid
nickname = whatever
```

Figura 6.15: Ejemplo de un archivo de configuración papoid.ini

6.5.2.2. Ejecución

A continuación se incluye una breve descripción del mecanismo usado por PAPOID para autenticar a los usuarios:

1. El Relying Party intenta acceder al servidor de OpenID protegido con PAPI (PAPOID).

2. El servidor de OpenID comprueba si el Relying Party es un sitio web no deseado (“unwanted”).

   a) Si es un sitio web no deseado, responde diciendo que la identidad de OpenID no es válida.

3. El OP comprueba si la identidad de OpenID coincide con la identidad de OpenID que esperaba recibir PAPOID de acuerdo con el archivo de configuración papoid.ini.

   a) Si no coincide, responde que la identidad de OpenID no es válida.

4. El OP comprueba si se está utilizando la extensión Simple Registration Extension y hay datos requeridos en la misma.

   a) Si hay datos del usuario que se requieran, muestra un formulario para que el usuario pueda proporcionarlos.

5. El OP responde al Relying Party.

Éste sería un diagrama de los pasos que tienen lugar durante el procesado de una petición OpenID:

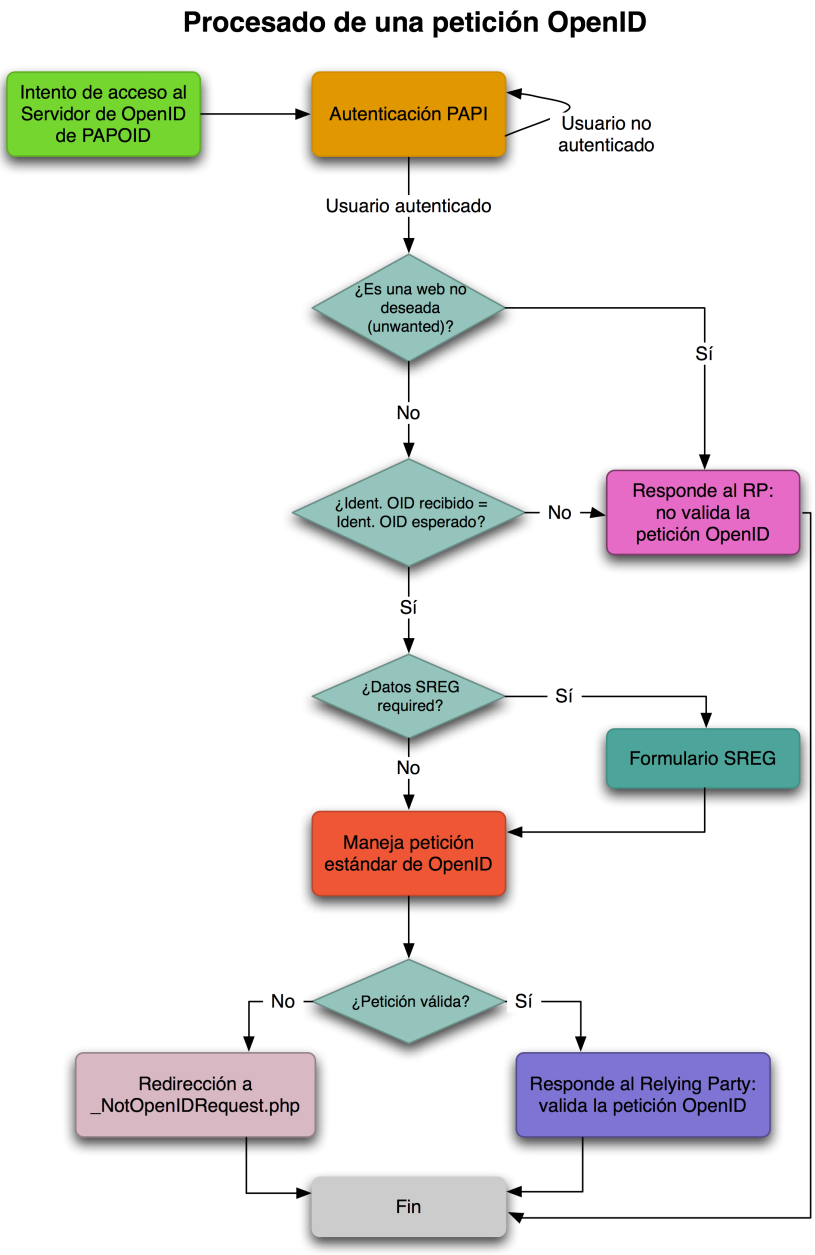

Figura 6.16: Procesado de una petición de OpenID en PAPOID

Una de las partes más interesantes del script utilizado en este proceso de autenticación es la comprobación de que la identidad de OpenID recibida coincide con la esperada, en base a los atributos de la aserción PAPI.

```
1     //function answer
2     $OIDExpected = $this->cfg['OIDPersonalIdentifier'];
3     foreach($this->cfg['PAPI_Attributes'] as $papi_key => $papi_attr){
4         $this->cfg['PAPI_Attributes'][$papi_key] = $userData[$papi_attr];
5     }
6
7     $OIDExpected = preg_replace(array_keys($this->cfg["PAPI_Attributes"]), array_values(
          ...$this->cfg["PAPI_Attributes"]), $OIDExpected);
8
9    /* Check if the openid.identity is valid according to the expected OID user identifier.
10    * Imagine that we have the following configuration in papoid.ini:
11    *
12    *           ...
13    *           OIDBaseIdentifier = http://me.example.org/
14    *           OIDPersonalIdentifier = $attr1/attr2
15    *           PAPI_Attributes = $attr1:uid, $attr2:uid
16    *           ...
17    *
18    * and $userData['uid'] = test.user, then the $OIDIdentity will be 'http://me.example.
          ...org/test.user/test.user',
19    * and it has to match with the received openid.identity.
20    *
21    * If openid.identity es Auth_OpenID_IDENTIFIER_SELECT, the OID user Identifier has to
          ...be built.
22    */
23
24    if($query['openid.identity'] != Auth_OpenID_IDENTIFIER_SELECT){
25        $OIDIdentity = $query['openid.identity'];
26
27        if($valid && in_array($query['openid.mode'], array("checkid_setup", "
              ...checkid_immediate"))){
28            $length = strlen($this->cfg['OIDBaseIdentifier']);
29            $OIDReceived = substr($OIDIdentity, $length);
30            if(!preg_match('_'.$OIDExpected.'_', $OIDReceived) || !preg_match('_'.
                  ...$OIDReceived.'_', $OIDExpected))                {
31                $valid=false;
32            }
33        }
34    }else{
35        $OIDIdentity = $this->cfg['OIDBaseIdentifier'].$OIDExpected;
36    }
```

Figura 6.17: Script de autenticación del usuario en PAPOID

6.6. Otros perfiles a desarrollar

En esta sección se pretenden señalar algunos puntos donde la aplicación puede ofrecer mayor funcionalidad. Esa funcionalidad puede venir desde la propia comunidad del Software Libre y está abierta a cualquiera que quiera participar en este proyecto libre.

Los principales puntos de mejora que pueden considerarse son aquellos referidos a las extensiones que soporta OpenID Authentication 2.0. Actualmente esta versión de PAPOID está preparada para soportar Simple Registration Extension 1.0, pero aún quedan muchas extensiones que podría soportar como:

- *OpenID Attribute Exchange 1.0*[14]: OpenID Attribute Exchange es un servicio de extensión de OpenID para el intercambio de información entre puntos finales. Provee mensajes para la obtención y el almacenamiento de información sobre la identidad.
- *OpenID Assertion Quality Extension 1.0[15]:* Esta extensión al protocolo OpenID Authentication provee los mecanismos necesarios para que un Relying Party pueda pedir información adicional sobre el modo concreto en el que un usuario se autenticó en el proveedor de identidad. Así mismo, permite que el Provider pueda adjuntar dicha información en las aserciones. Esta información puede ser necesaria para casos de usos en los que para que el RP pueda comprobar la fiabilidad del OP la identidad de éste no sea suficiente por si sóla.
- *OpenID DTP Version 1.0 Adjuncts*[16]: OpenID DTP es un protocolo para enviar, recibir y transmitir una cantidad arbitraria de información cifrada y firmada entre dos puntos finales.
- *OpenID Provider Authentication Policy Extension 1.0*[17] : Esta extensión al protocolo OpenID Authentication provee los mecanismos necesarios para que un Relying Party pueda acordar la política de autenticación que va a aplicar el OP antes de que ésta tenga lugar y para que el OP pueda informar al Relying Party sobre las políticas de autenticación que fueron usadas. De esta manera un Reying Party puede solicitar que el usuario se autentique, por ejemplo, utilizando mecanismos que sean resistentes a ataques de phishing. Del mismo modo, el OpenID Provider puede comunicar al Relying Party si el usuario cumple o no con los requisitos de la política, o políticas, solicitada, así como la fuerza general de los credenciales que se estén utilizando.

Además de las posibilidades de ampliación del proyecto que se presentan con las anteriores extensiones, también sería posible una adaptación del mismo para que no sólo soportara identificadores de usuario en formato URI sino también documentos XRDS.

6.7. Herramientas utilizadas

El desarrollo de este proyecto ha sido realizado utilizando diversas herramientas, entre las que pueden destacarse:

- *Plataforma de desarrollo Eclipse*[22]
  El entorno de desarrollo Eclipse se ha convertido desde su nacimiento en una plataforma de desarrollo de código abierto que ha tenido gran acogida dentro de la comunidad de desarrolladores, tanto de Software Libre como no libre. Aunque inicialmente fue concebida para crear aplicaciones Java, su diseño y modularidad permiten que se pueda utilizar para crear una gran variedad de aplicaciones en distintos lenguajes de programación. Esta plataforma introduce una serie de herramientas útiles para el desarrollo de software como integración de proyectos con sistemas de control de versiones o gestores de tareas internas.

- *Apatana*[23]
  Este software es una extensión o módulo de Eclipse para el desarrollo de aplicaciones en Ajax, JavaScript, HTML, DOM, CSS o PHP entre otros.

- *Control de versiones Subversion*[24]
  Se llama control de versiones a la gestión de versiones de todos los elementos que forman la línea base de un producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado junto a las posibles especializaciones realizadas para algún cliente específico. Se ha seleccionado el sistema Subversion por ser el más extendido hasta la fecha y el que mejor se integra con otras herramientas y tecnologías utilizadas en el proyecto.

Capítulo 7

Conclusiones

Este proyecto, cuyo objetivo pretende establecer un punto de conexión entre los servicios de identidad de las instituciones que utilizan PAPI y aquellos sitios web que utilizan el protocolo OpenID Authentication, ha sido decisivo para hacerme ver el esfuerzo y la madurez necesaria para afrontar una tarea de esta envergadura.

Un primer aspecto observado al realizar este proyecto, ha sido la necesidad de documentarse e investigar antes de realizar ninguna implementación. Si bien es importante llevar a la práctica los conceptos asimilados, más importante es realizar un estudio exhaustivo de las tecnologías que se van a utilizar, para así poder tomar las decisiones correctas cuando fuese necesario.

He tenido que poner en práctica muchos de los conocimientos adquiridos en la carrera y aplicar constantemente los aspectos que más me han calado a lo largo de estos años: constancia, intensidad y saber enfrentarme a problemas que a priori parecen superarme.

Quisiera destacar lo fascinante que puede llegar a ser el hecho de trabajar con una tecnología que está en evolución continua, como es el caso del proyecto OpenID, un proyecto vivo. OpenID cuenta con un gran respaldo de la comunidad y el mero hecho de poder formar parte de ella y colaborar me ha transmitido una sensación de satisfacción personal inefable.

No puedo dejar de mencionar los problemas que me encontré al enfrentarme a PAPI por primera vez, pero cuya superación fueron, sin duda alguna, el motor de impulso para darme cuenta de que con trabajo y constancia podemos lograr nuestros objetivos. Además para poder desarrollar PAPOID fue necesario actualizar la versión del phpPoA 1.2 para añadirle algunas funcionalidades nuevas, hecho

me ha permitido integrarme en el equipo de desarrollo de PAPI en RedIRIS y desempeñar una tarea que cada día me parece más satisfactoria, apasionante y constructiva, tanto personal como profesionalmente.

Aún así, lo que quizás vea más importante es la capacidad de abstracción y la seguridad que se adquiere tras enfrentarse a problemas que no tienen una solución aparente y que al final, tras muchos intentos y esfuerzo se es capaz de superar. Sinceramente, pienso que este aspecto ha sido el que más me ha hecho madurar intelectual y personalmente, sobre todo porque ahora sé que puedo ser capaz de enfrentarme a cualquier tecnología desconocida y conseguir entenderla y utilizarla.

Hoy en día este proyecto se sigue desarrollando, intentado mejorarlo y adaptarlo a las nuevas especificaciones y extensiones del protocolo OpenID Authentication que van surgiendo, pues, como se comentó en el apartado de posibles ampliaciones, OpenID es un proyecto que está vivo y algunos de los principales puntos de ampliación que pueden considerarse son aquellos referidos a las extensiones que soporta OpenID Authentication 2.0.

Por último me gustaría recalcar el papel que el Software Libre ha jugado en el desarrollo de este proyecto. Utilizar tecnologías libres ha aportado una base de conocimiento y fuente de formación inagotable, sin la que no podría haberse realizado este proyecto.

Capítulo 8

Anexos

8.1. Ejemplo de uso de PAPOID

El objetivo de esta sección es mostrar un ejemplo de uso del servidor PAPI OpenID desarrollado. En primer lugar se detalla cuál es la configuración del Virtual Host de Apache que se ha usado, seguido del contenido del archivo papoid.ini para este ejemplo concreto.

Una vez quede claro el escenario de uso, se mostrará paso a paso el proceso de autenticación de un usuario en *www.padtube.com* usando este servidor de autenticación. Además se especifica el contenido de los mensajes que se intercambian en cada paso.

8.1.1. Configuración del PAPOID de ejemplo

Esta sería la configuración del Virtual Host en Apache:

```
1  <VirtualHost *>
2      ServerName yo.rediris.es
3      DocumentRoot "/home/spool/teresamc/openid/www"
4      ServerAdmin teresa.matamoros@rediris.es
5      AliasMatch ^/[^_] /home/spool/teresamc/openid/www/_index.html
6      AliasMatch ^/$ /home/spool/teresamc/openid/www/_ident_select.html
7      AddType application/x-httpd-php .php
8      <Location "/">
9          Options Indexes FollowSymLinks
10         Order Deny,Allow
11         Allow from all
12     </Location>
13 </VirtualHost>
```

Figura 8.1: Virtual Host de Apache para el Ejemplo paso a paso

Y el archivo de configuración de PAPOID es el que sigue:

```
[PAPOID_Main]
     PAPOID_Directory = /home/spool/teresamc/openid/OpenID_directory
     Unwanted = www.site1.com, www.site2.com
     PoA_Name = yo.rediris
     OIDRequest_Error_File = _NotOpenIDRequest.html
     OIDServer = http://yo.rediris.es/_openid_server.php
     OIDBaseIdentifier = http://yo.rediris.es/
     OIDPersonalIdentifier = attr1
    PAPI_Attributes = attr1:uid
[SREG_Extension]
     sreg_nickname_name = Nick
     sreg_email_name = Correo electrónico
     sreg_fullname_name = Nombre
     sreg_dob_name = Fecha de nacimiento
     sreg_gender_name = Género
     sreg_postcode_name = Código postal
     sreg_country_name = Pais
     sreg_language_name = Idioma
     sreg_timezone_name = Zona horaria

     //SReg values, taken from the attributes of the PAPI assertion.
     sreg_nickname_value = uid
     sreg_email_value = mail

[www.laudr.com]
     sreg_email_name = Correo
     //Not take the nickname value form the PAPI assertion for this site.
     nickname = asdf
```

Figura 8.2: Papoid.ini para el Ejemplo paso a paso

8.1.2. Ejemplo paso a paso

A continuación se va a mostrar el ejemplo de uso de PAPOID paso a paso. Consideremos la configuración especificada en la sección anterior. En concreto vamos a utilizar este OpenID Provider para autenticarnos en *http://www.padtube.com* con el usuario de OpenID *http://yo.rediris.es/teresa.matamoros*.

Ha de tenerse en cuenta que este Consumer utiliza el modo de funcionamiento *dumb mode*, por lo que no es necesario establecer ni almacenar un secreto compartido entre el Relying Party y el Provider, ahora bien, en contrapartida el sitio web www.padtube.com tendrá que hacer una petición directa a PAPOID con un mensaje de tipo *check_authentication* para comprobar que la respuesta que el Provider le ha dado es correcta.

En primer lugar proporcionamos al cliente web nuestra URL de OpenID:

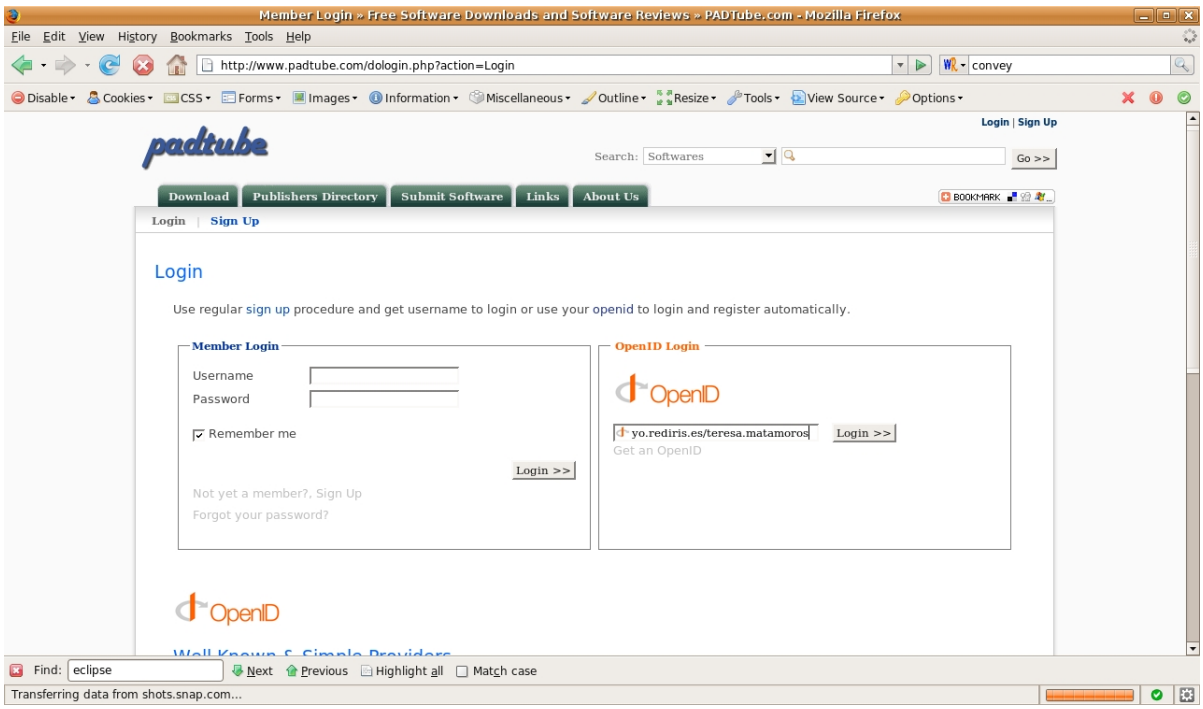

Figura 8.3: Formulario de login en el Ejemplo paso a paso

Ahora lo que hace el sitio web *http://www.padtube.com/dologin.php?action=Register* es aplicar el protocolo Discovery para obtener el documento HTML que está asociado a la URL http://yo.rediris.es/teresa.matamoros. Es decir, accede al contenido de _index.html:

```
1 <html>
2    <head>
3         <link rel="openid2.provider" href="http://yo.rediris.es/_openid_server.php" />
4    </head>
5    <body></body>
6 </html>
```

Figura 8.4: Ejemplo de un documento HTML usado para el OpenID Identifier del ejemplo paso a paso

El Relying Party parsea el documento y obtiene la dirección del servidor de OpenID, *http://yo.rediris.es/_openid_server.php*. El objetivo del Relying Party es pedir al Provider que verifique la identidad del usuario. Para ello, enviará la siguiente petición al servidor:

```
1 http://yo.rediris.es/_openid_server.php?
2       openid.return_to=http://www.padtube.com/dologin.php?action=OpenIdLogin&
3       openid.mode=checkid_setup&
4       openid.identity=http://yo.rediris.es/teresa.matamoros&
5       openid.trust_root=http://www.padtube.com&
6       openid.sreg.required=email,fullname&
```

Figura 8.5: Petición de autenticación en el Ejemplo paso a paso

Al estar protegido por PAPI, si no tenemos permiso de acceso, el phpPoA nos pedirá que nos autentiquemos. Para la autenticación se utiliza un Point of Access PAPI, que a su vez delega el proceso de autenticación en el Authentication Server(AS).

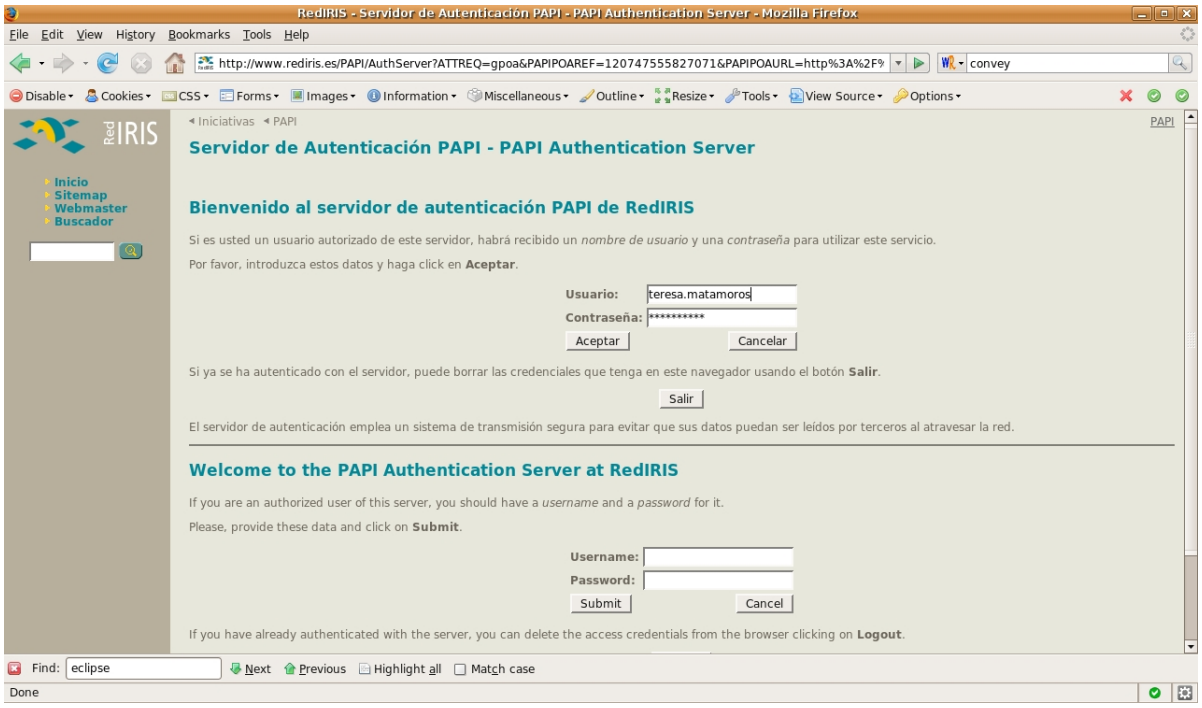

Figura 8.6: Autenticación en un AS PAPI

La petición de autenticación que recibe el AS es:

```
http://www.rediris.es/PAPI/AuthServer?
      ATTREQ=gpoa&
      PAPIPOAREF=12087714291184&
      PAPIPOAURL=http://www.rediris.es/papiGPoA/papiPoA
```

Figura 8.7: Petición de autenticación en un AS PAPI

Una vez que el usuario se autentica en el AS, el navegador recibe una redirección al servidor de OpenID. Ahora, junto a los parámetros de openid que envió inicialmente el Relying Party aparecen los relativos a la autenticación PAPI, que permitirán el acceso al recurso.

```
http://yo.rediris.es/_openid_server.php?
       openid.return_to=http://www.padtube.com/dologin.php?action=OpenIDLogin&
       openid.mode=checkid_setup&
       openid.identity=http%3A%2F%2Fyo.rediris.es%2Fteresa.matamoros&
       openid.trust_root=http%3A%2F%2Fwww.padtube.com&
       openid.sreg.required=email,fullname&&
       ACTION=CHECKED&
       DATA=Ps57jnN7tzkzZPMXnS4rqEOTvKC7HR3KuCNWw7PYg%2BaXgGQPy1r2wHdemkfmEmQhzpH0%2
            ...FVjw5Tl5%0A%2BfjYLICf388h
       65d8lwKMfsgW3nCoMZj1KDBCvuAumiK7YSWoZl8Zs6m9%2F4uC7MwVf%2BmLHx1Yz0ks%0A1OQGO2c%2
            ...BKDwumvgHiF1oqumAdDWo
       OwqRlIs86EMB7JRsgRHH5i7eCOtpnF9tQGpZ6zcXF4shTITJ%0
            ...ANBmWq5noYZohAVgSnOkISH2LR8udXofJ33bVHT7yYZEzKeKJnH6
       C%2FU50ylU%2B5j2o2rUIGlk%2BrVvY%0Aap6UruZdaFP4oBLP57B0RW4Y%2BC9vto6ovzQAaQ%3D%3D%0A
```

Figura 8.8: Petición de autenticación en el Ejemplo paso a paso

En los campos del protocolo OpenID aparece el atributo openid.sreg.required, por ello el OpenID Provider nos pide que rellenemos un formulario con la información que http://www.padtube.com solicita del usuario haciendo uso de la extensión Simple Registration Extension 1.0:

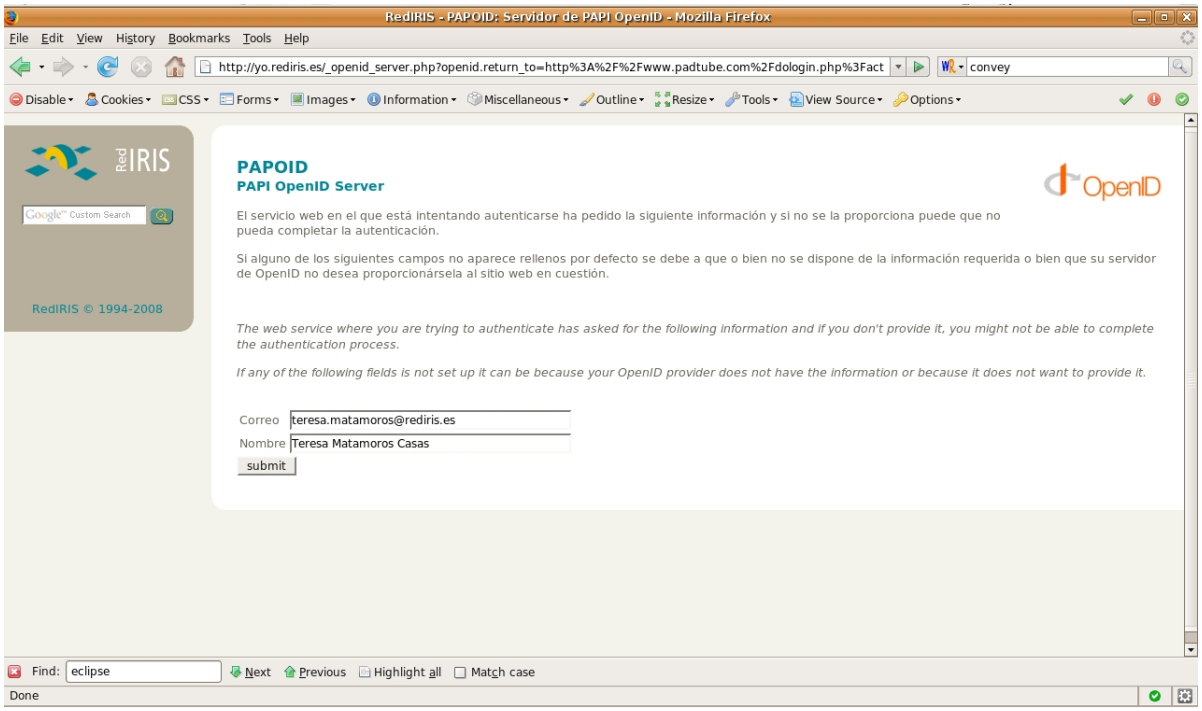

Figura 8.9: Formulario de SREG para el Ejemplo paso a paso

Cuando el usuario pulsa el botón de confirmar la respuesta que se le envía a PadTube es:

```
http://www.padtube.com/dologin.php?action=OpenIdLogin&
       openid.assoc_handle={BHMAC-SHA1}{48073531}{BroLYmA}&
       openid.identity=http://yo.rediris.es/teresa.matamoros&
       openid.mode=id_res&
       openid.op_endpoint=http://yo.rediris.es/_openid_server.php&
       openid.response_nonce=2008-04-17T11/32/01Zk5DyZG&
       openid.return_to=http://www.padtube.com/dologin.php?action=OpenIdLogin&
       openid.sig=IbUxJ7CQ55om03W2hfVRAFbeaeU&
       openid.signed=assoc_handle,identity,mode,op_endpoint,response_nonce,return_to,signed
            ...,sreg.email,sreg.fullname&
       openid.sreg.email=teresa.matamoros@rediris.es&
       openid.sreg.fullname=Teresa Matamoros Casas
```

Figura 8.10: Respuesta a la petición de autenticación en el Ejemplo paso a paso

En el momento en que el Relying Party recibe esta confirmación a su petición de autenticación tiene que contactar directamente con el OpenID Provider para comprobar que la respuesta que ha recibido es correcta:

```
http://www.yo.rediris.es/_openid_server.php?
       openid.assoc_handle={BHMAC-SHA1}{48073531}{BroLYmA}&
       openid.identity=http://yo.rediris.es/teresa.matamoros&
       openid.mode=check_authentication&
       openid.op_endpoint=http://yo.rediris.es/_openid_server.php&
       openid.response_nonce=2008-04-17T11/32/01Zk5DyZG&
       openid.return_to=http://www.padtube.com/dologin.php?action=OpenIdLogin&
       openid.sig=IbUxJ7CQ55om03W2hfVRAFbeaeU&
       openid.signed=assoc_handle,identity,mode,op_endpoint,response_nonce,return_to,signed
            ...,sreg.email,sreg.fullname&
       openid.sreg.email=teresa.matamoros@rediris.es&
       openid.sreg.fullname=Teresa Matamoros Casas
```

Figura 8.11: Petición directa de www.padtube.com al Provider para comprobar la respuesta a la petición de autenticación

Ahora el servidor de OpenID tras comprobar que la petición check_authentication es correcta, envía la siguiente respuesta:

```
<html>
    <head></head>
    <body>
        ns: http://specs.openid.net/auth/2.0
        is_valid: true
    </body>
</html>
```

Figura 8.12: Respuesta del Provider a la petición de de www.padtube.com

Tras seguir con éxito todos los pasos que marca el modo *dumb mode* del protocolo OpenID Authentication el sitio web http://www.padtube.com acepta que el usuario http://yo.rediris.es/teresa.matamoros es válido (nótese que en la esquina superior derecha de la imagen aparece el nombre completo proporcionado por el usuario en el formulario SREG, junto a la opción de hacer logout).

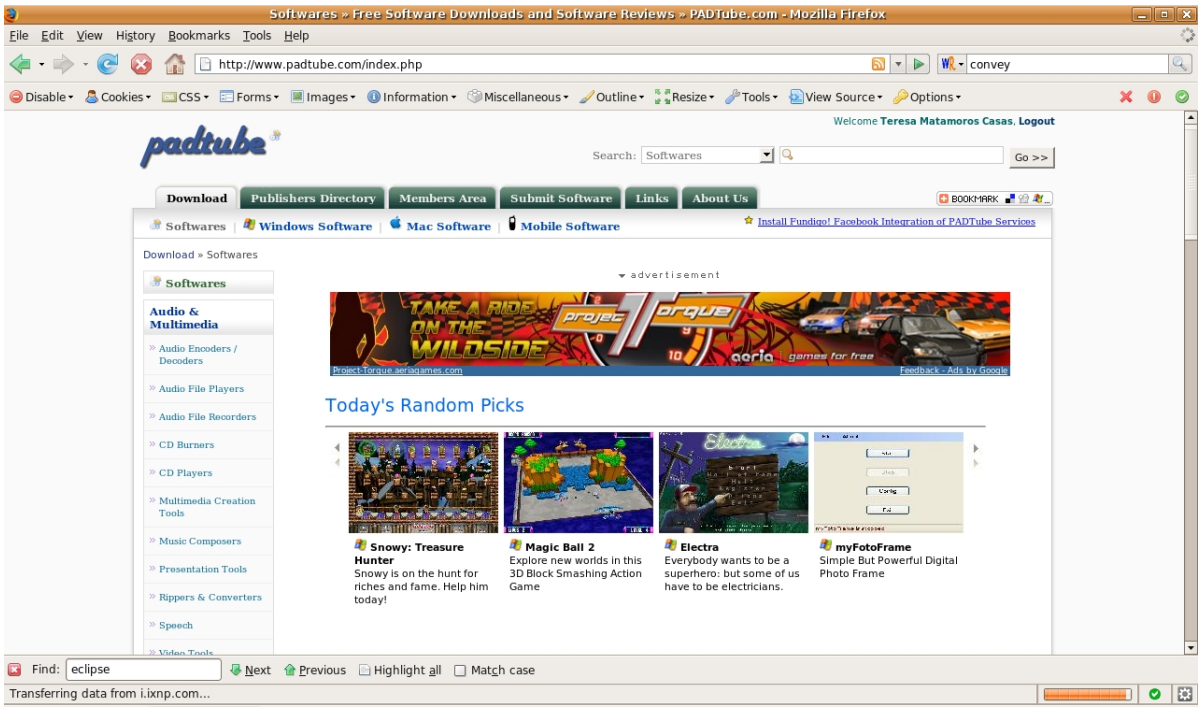

Figura 8.13: Usuario autenticado en Padtube

8.2. Costes temporales

Esta sección incorpora algunas estimaciones temporales del proyecto:

	Días	Horas/Días	Total
Recopilación de información	25	5	125
Diseño	20	5	100
Implementación y pruebas	60	5	300
Generación de la documentación	25	5	125
Total	-	-	650

Bibliografía

[1] M. Erdos, S. Cantor, "Shibboleth architecture draft v05," h*ttp://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-arch- v05.pdf*, (2002).

[2] D. Lopez, R. Castro-Rojo, "Ubiquitous Internet access control: the PAPI system," *Database and Expert Systems Applications, 2002. Proceedings. 13th Interna- tional Workshop on*, (2002): 441-445.

[3] P. Mishra et al., "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)", *OASIS SSTC*, (September 2003).

[4] E. Maler, J. Hughes, "Technical overview of the OASIS Security Assertion Markup Language (SAML)", *OASIS SSTC*, (March 2004).

[5] SURFnet, "The A-select authentication system", *http://www.a-select.surfnet.nl.*

[6] C. Rigney, A. Rubens, W. Simpson, S. Willens, "Remote Authentication Dial In User Service (RADIUS)," RFC2138.

[7] Liberty Alliance, "Liberty Alliace Project" , *http://www.libertyproject.org*.

[8] Sun Microsystems, "Sun Access Manager", *http://www.sun.com/software/products/access mgr/index*.

[9] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management", *Proceedings of the second ACM workshop on Digital identity man- agement*, 11-16, 2006.

[10] M. Atwood et al., "OAuth Core 1.0", *http://oauth.net/core/1.0/*.

[11] D. Recordon, B. Fitzpatrick, "OpenID Authentication 1.1",*http://openid.net/specs/,* (May 2006)

[12] ”OpenID Authentication 2.0 -Final”, *http://openid.net/specs/*, (December 5, 2007)

[13] J Hoyt, J Daugherty, JanRain, D. Recordon, VeriSign, “OpenID Simple Registration Extension 1.0”, *http://openid.net/specs/, (June 30, 2006)*

[14] D. Hardt, J. Bufu, Sxip Identity, J. Hoyt, JanRain, “OpenID Attribute Exchange 1.0 - Final”, http://openid.net/specs/, (December 5, 2007)

[15] D. Recordon, VeriSign, A. Glasser, VxV Solutions, P.Madsen, NTT, “OpenID Assertion Quality Extension 1.0 - Draft 1”, *http://openid.net/specs/*, (November 29, 2006)

[16] G. Monroe, JanRain, “OpenID DTP Version 1.0 Adjuncts - Draft 02”, *http://openid.net/specs/*, (August 9, 2006)

[17] D. Recordon, VeriSign, “OpenID Provider Authenticacion Policy Extension 1.0 - Draft 1”, *http://openid.net/specs/*, (June 22, 2007)

[18] The Apache Software Foundation, *http://www.apache.org*

[19] PHP Hypertext Preprocessor, *http://www.php.net*

[20] PAPI, *http://papi.rediris.es*

[21] Red Española de I+D, RedIRIS, *http://www.rediris.es*

[22] Eclipse, *http://www.eclipse.org*

[23] Aptana, *http://www.aptana.com*

[24] Subclipse, http://subclipse.tigris.org/

[25] Servidor de Identidad de RedIRIS, SIR, *http://rediris.es/sir*

[26] eduGAIN, *http://www.edugain.org/*

[27] Cándido Rodriguez Montes, Trabajo de investigación del programa de doctorado de Informática Industrial, *“Infraestructuras Confederadas de Autenticación y Autorización”*

[28] J. Hodges, Neu Star, “Technical Comparison: OpenID and SAML - Draft 05”, (Diciembre 17, 2007)

[29] H. Granqvist, VeriSign, Inc., “OpenID authentication security profiles“, *http://openid.net/specs/,* (Septiembre, 2006)