Universitat Oberta de Catalunya

uoc.edu

Federación de Identidades: Estado del Arte e implementación de un caso real

Proyecto de fin de máster

Alejandro García de Marina Martín
Máster de Seguridad de las Tecnologías de la Información y las Comunicaciones
Identidad Digital

Jordi Guijarro Olivares

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Federación de Identidades: Estado del Arte y implementación de un caso real*
Nombre del autor:	*Alejandro García de Marina Martín*
Nombre del consultor/a:	Jordi Guijarro Olivares
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Área del Trabajo Final:	*Identidad Digital*
Idioma del trabajo:	*Español*
Palabras clave	*Gestión de Identidades, Esquemas Federados, Estado del arte*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La gestión de identidades (IdM, por sus siglas en ingles Identity Management) es el área que se ocupa de manejar las diversas identidades y su información relacionada, a lo largo de diversos dominios y/o servicios, garantizando así la seguridad y privacidad de los usuarios. Los sistemas de gestión de identidades han ido evolucionando notablemente a lo largo de los años pasando por las listas de control de accesos, sistemas con arquitectura SILO, sistemas con arquitectura centralizada y sistemas con arquitectura federada. Estos últimos (federados) han sido el objeto principal de estudio.

A lo largo del presente trabajo, se ha realizado un análisis en profundidad de las peculiaridades y los principales estándares desarrollados para cada uno de estos sistemas. Así, por ejemplo se puede ver en detalle estándares como LDAP, Kerberos o radius para los sistemas con arquitectura centralizada y estándares como SAML, OAuth, OpenID Connect y Mobile Connect para los sistemas con arquitectura federada.

El presente trabajo por lo tanto se puede definir con un estado del arte de los sistemas de gestión de identidades focalizado mayormente en los sistemas federados. Además, para poder ahondar en estos últimos, se ha desarrollado una implementación de un sistema de gestión de identidades federado para poner en práctica el uso de OpenID Connect en un entorno de laboratorio lo más fidedigno posible con la realidad.

Índice

Índice de ilustraciones

Índice de Tablas

Capítulo 1: Introducción

1.1 Contexto y justificación del Trabajo

La gestión de identidades (IdM, por sus siglas en ingles Identity Management) es el área que se ocupa de manejar las diversas identidades y su información relacionada, a lo largo de diversos dominios y/o servicios, garantizando así la seguridad y privacidad de los usuarios. Gracias al IdM únicamente los usuarios validados puedan interactuar de forma legítima con un sistema específico [36]. Por ejemplo, tomando como sistema un país, la forma de gestionar las identidades de los usuarios (los ciudadanos) y poder identificarlos sería por medio de los documentos oficiales, tales como el DNI o el Pasaporte.

En la actualidad, con el aumento masivo de los sistemas de información y con la llegada de Internet, la gestión de identidades está en constante evolución. En los inicios, los sistemas informáticos estaban pensando para que un único operador tuviese acceso a un equipo y por lo tanto tuviese la posibilidad de autenticarse y autorizarse sobre dicho equipo para poder tener los privilegios necesarios para interactuar con él. Sin embargo, poco a poco se fueron implantando los sistemas multi-usuario, en los que un mismo equipo se encarga de gestionar las identidades de múltiples usuarios. Tomar como ejemplo un ordenador personal de hoy en día, en el que se pueden crear múltiples cuentas de usuario, las cuales tienen distintos privilegios y por lo tanto pueden realizar distintas acciones dependiendo del usuario que este autenticado. Además, estos usuarios no tienen por qué ser conscientes de que otros usuarios están en el sistema, así como de la repartición en tiempo y espacio de los recursos, siendo este un proceso totalmente transparente para ellos. Esta gestión de identidades ha sido llevada al extremo con la entrada de Internet, en los que muchos usuarios pueden acceder simultáneamente al mismo servicio, aplicación o recurso alojado en un servidor externo.

Al mismo tiempo, el concepto de identidad ha ido enriqueciéndose, dejando de ser exclusivo de una persona física (como en el ejemplo del DNI) y pasando a ser también algo digital.

Hoy en día, existen cada vez más formas complejas y diversas de gestionar las identidades digitales de los usuarios, e incluso puedan convivir varios sistemas de gestión de identidades simultáneamente en un mismo sistema. Esto hace que entender los sistemas de gestión de identidades actuales sea una labor cada vez más compleja, a pesar de que todos los usuarios hacen uso en un momento u otro de dichos sistemas.

Por este motivo, en el presente Trabajo de Fin de Máster se aclaran todos los conceptos relacionados en el ámbito de la gestión de identidades con la finalidad de que cualquier usuario pueda entender los sistemas de gestión de identidades actuales

y pueda extraer conclusiones acerca de las ventajas y desventajas poseen cada uno de ellos.

1.2 Objetivos del Trabajo

El presente trabajo de Fin de Máster tiene como objetivo principal desarrollar un estado del arte de la evolución de los sistemas de gestión de identidades, centrándose específicamente en los más actuales y novedosos, es decir, en los sistemas de gestión de identidades federados.

Se profundiza en conceptos como la gestión de identidades, IAAA (Identificar, autenticar, autorizar y auditar) y la identidad digital. Además, se describen especificaciones concretas de soluciones ya propuestas y en uso actualmente para la gestión de identidades federadas como por ejemplo: SAML, OAuth, OpenIdConnect y MobileConnect. El objetivo de esta descripción es poder realizar una comparativa y poder extraer conclusiones acerca del ámbito concreto de aplicación de cada una de ellas y poder analizar sus flujos para extraer similitudes y diferencias y las ventajas e inconvenientes que tienen asociados.

Finalmente, se realiza una implementación de un sistema de gestión de identidades utilizando OpenAM. Esta implementación tiene como finalidad poder comprobar de primera mano los flujos, es decir, los intercambios de información que se realizan entre los diferentes agentes que intervienen en el proceso para poder obtener resultados acerca de su funcionamiento.

Sintetizando lo expuesto anteriormente, los objetivos principales del presente TFM son:

1. Introducir los sistemas de gestión de identidades.
2. Evolución de los sistemas de gestión de identidades
3. Comprender los sistemas de gestión de identidades federadas.
4. Entender las soluciones propuestas actualmente.
5. Elaborar una comparativa de las soluciones actuales.
6. Implementar un caso real y funcional al problema abordado.

1.3 Enfoque y método seguido

La metodología que se ha seguido para desarrollar el trabajo de fin de máster es iterativo e incremental y se basa en la metodología ágil de desarrollo software [29]. Esto significa, que el desarrollo se basa en fases o iteraciones cortas en el tiempo, en las que se repite el proceso de desarrollo. Estas pequeñas fases, logran obtener un producto mínimo viable (MVP) [29]. En el caso concreto del presente trabajo, estos MVP serán versiones desarrolladas del documento escrito (objetivos 1-5) y versiones funcionales de la implementación (objetivo 6)

La elección de esta metodología se debe a las ventajas que facilita tener un MVP al final de cada iteración, ya que permite ir evaluando constantemente el desarrollo del

mismo. Esto se traduce, en una efectiva forma de adaptar el trabajo a nuevos objetivos que vayan surgiendo a lo largo del desarrollo del mismo y a una rápida corrección de errores ante los imprevistos y errores que puedan surgir.

1.4 Planificación del Trabajo

Con la finalidad de cumplir con los objetivos marcados (ver sección 1.2) se han extraído 9 tareas que se han de desarrollar a lo largo de aproximadamente 3 meses. El trabajo realizado a lo largo de cada mes, concluye con una entrega (MVP). El proceso de planificación del trabajo se puede observar en la ilustración 1.

La primera entrega, consta de 4 tareas: búsqueda masiva de información, introducción, explicar la evolución de los sistemas de gestión de identidades (IdM) y descripción de los sistemas de gestión de identidades federados (FIM). Al concluir esta entrega, se espera tener realizado completamente la introducción de la memoria (objetivo 1) y la evolución de los sitemas IdM (objetivo 2). Además, se espera poder empezar con tareas definidas para la segunda entrega.

En la segunda entrega, se desarrollarán las siguientes 3 tareas: descripción de los sistemas FIM (objetivo 3), descripción de las soluciones FIM propuestas actualmente (objetivo 4) y realizar una comparativa de dichas soluciones (objetivo 5). Al concluir esta entrega, se espera tener completamente desarrollado el estado del arte asociado a los sistemas FIM (objetivos 1-5).

La tercera entrega se corresponde con la última entrega del trabajo y consta de tres tareas: preparar el entorno de desarrollo, implementar un sistema de gestión de identidades federado y documentar el proceso de desarrollo. Todas estas tareas se corresponden con el objetivo 6.

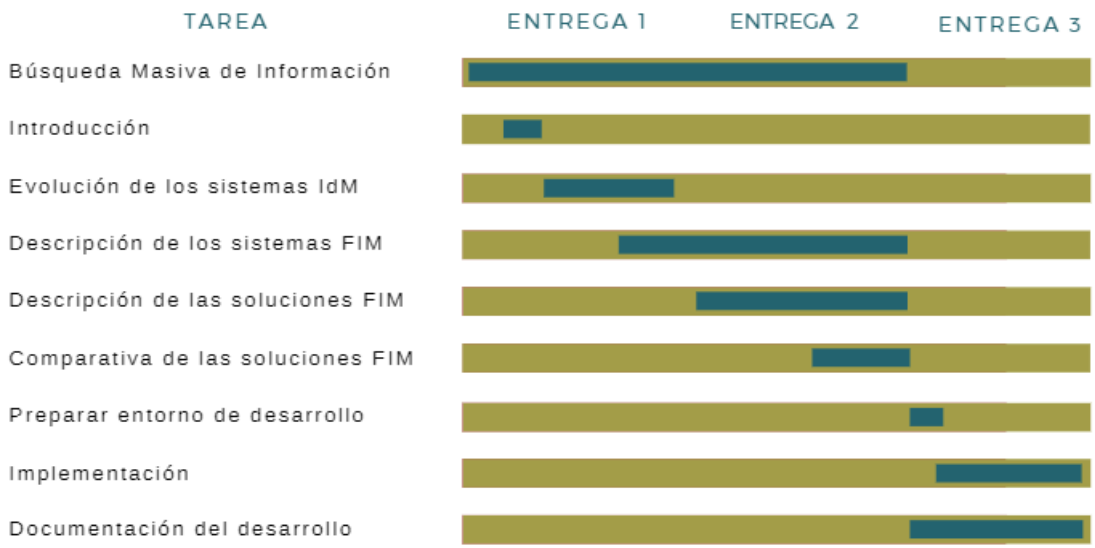

Ilustración 1: Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

El presente trabajo de fin de máster tiene como finalidad obtener 2 productos:

1. Documento del estado del arte entorno a la gestión de identidades federadas.
2. Implementación software de un sistema de gestión de identidades federado.

1.6 Descripción de los otros capítulos de la memoria

El presente documento que sirve de entrega como Trabajo de Fin de Máster se estructura como sigue: En el Capítulo 2 se introducen los sistemas de gestión de identidades; en el Capítulo 3 se profundiza en los sistemas de gestión de identidades federados; en el Capítulo 4 se detalla el proceso de implementación de un caso de uso real de un sistema de gestión de identidades, así como los resultados de dicha implementación; en el Capítulo 5 se exponen las principales conclusiones y las líneas futuras de desarrollo.

Capítulo 2: Sistemas de Gestión de Identidades

En este capítulo se introducen los principales conceptos referidos a los sistemas de gestión de identidades. Así mismo, se recorren los diferentes tipos y modelos de sistemas que se han ido creando en este ámbito a lo largo del tiempo. El objetivo principal es que el lector sea capaz de entender las funcionalidades y objetivos de los sistemas de gestión de identidades para poder entender los capítulos posteriores.

2.1. Preámbulo

En primer lugar es imprescindible definir el concepto de identidad digital. Este se puede definir como el conjunto de características, preferencias y reputación de una entidad frente a un sistema de información digital [6,36]. Note que se habla del concepto entidad, que no solo incluye usuarios sino también empresas, sensores, dispositivos y agentes software (servicios web, clientes web, etc…) entre otros. Normalmente, se suele intentar asociar el concepto de identidad física al concepto de identidad digital con el objetivo de poder identificarlas conjuntamente. Sin embargo, esto no siempre es posible (en el caso de un agente software) o lo deseado (en el caso de un usuario) ya que puede afectar a la privacidad, por lo que hay que hacer especial hincapié en diferenciar ambas identidades e incluso saber que una mismo identidad física puede tener asociadas múltiples identidades digitales [41].

Todos los sistemas de información manejan identidades digitales. Estos sistemas han de poder garantizar los tres principios básicos de la seguridad: la integridad, confidencialidad y disponibilidad [6]. Con el objetivo de asegurar estos principios, dichos sistemas han de establecer los mecanismos de IAAA (identificación, autenticación, autorización y auditoria) pertinentes con el objetivo de restringir el acceso a las identidades digitales no autorizadas. Estas restricciones permiten que una entidad no legítima no pueda modificar la información (integridad), acceder a la información (confidencialidad) o denegar el acceso a una entidad legítima (disponibilidad). Para poder llevar a cabo este proceso se utilizan las credenciales, que no son más que una característica o conjunto de ellas previamente certificadas por una entidad de confianza.

Es interesante detenerse en el concepto de IAAA (ver Ilustración 2). El concepto de IAAA se define como:

- **Identificación**: es el proceso por el cual se puede garantizar la identidad de una entidad de forma exclusiva en un sistema
- **Autenticación**: es el proceso por el cual se valida o demuestra una identidad asegurando así, que una entidad es quien dice ser.
- **Autorización**: es el proceso para proteger los recursos de un sistema, restringiendo su acceso únicamente a las personas autorizadas.

- **Auditoría**: es el proceso para comprobar sucesos pasados con el objetivo de detectar actividades de identificación, autenticación o autorización no legítimas.

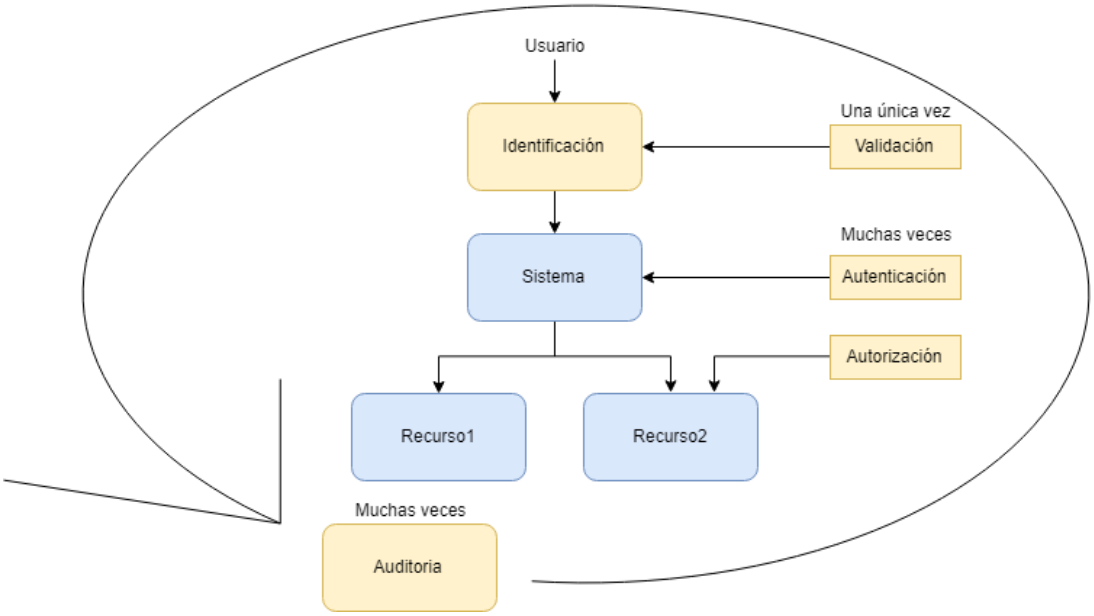

Ilustración 2: Flujo de IAAA

La identificación permite determinar exclusivamente la identidad de una entidad en base a los atributos que la propia entidad ha proporcionado o que el sistema ha recopilado. Es de vital importancia distinguirlo del concepto de autenticación. El proceso de identificación no sirve en ningún momento para validar las credenciales que proporciona una entidad. El concepto de identificación está muy ligado al proceso de validación, el cual se encarga de verificar la identidad previamente identificada, con otra identidad por ejemplo una identidad física. Supóngase un usuario que se da de alta en la aplicación de un servicio de banca, dicho usuario en la fase de registro ha proporcionado la información suficiente como para poder identificar su identidad. Sin embargo, el servicio de banca no posee la certeza de que dicho usuario es quien dice ser, por lo que le solicita que se presente en las oficinas del banco proporcionando su DNI y su nombre de usuario. Este proceso de validación logra verificar la identidad digital de la persona con su identidad física.

Los métodos de autenticación sirven para verificar que una entidad es quien dice ser. Este proceso ocurre muchas veces a lo largo de la vida de un sistema. Se pueden definir de acuerdo a cinco factores específicos [6*]: algo que se sabe, algo que se tiene, algo que eres, algo que haces o donde se está*. Normalmente, los principales sistemas de gestión de autenticación suelen implementar los dos primeros. En el primer caso (*algo que se sabe*), se suele utilizar un código personal (PIN) o contraseña para validar la identidad de una entidad. Este es el método más usado hoy en día con este propósito. En el segundo caso (*algo que se tiene*), se suelen utilizar tarjetas inteligentes o tokens [35]. Hoy en día, técnicas como *User Behavior Analysis (UBA)* [2] están empezando a dar más peso a otros factores como el de *algo que haces*. En diferentes escenarios, es posible que se necesite un nivel extraordinario de seguridad que con un

único factor de autenticación no sea posible. La combinación de múltiples factores de autenticación (MFA de sus siglas en ingles Multi-Factor Authentication) consiste en incorporar 2 o más factores de autenticación. Un claro ejemplo de MFA, es cuando un usuario intenta autenticarse en un sistema de información y este además de validar las credenciales proporcionadas (*algo que se sabe*) envía un PIN al dispositivo móvil del usuario (*algo que se tiene*) para que lo introduzca en su formulario de autenticación.

Hay que hacer especial hincapié en diferenciar el concepto de autenticación frente al de autorización. Mientras que el primero valida la identidad de una entidad, el segundo se encarga de definir y comprobar los privilegios o permisos que tiene dicha entidad sobre el sistema. Supóngase el ejemplo de un ordenador personal. Al iniciar la sesión, el usuario ha de introducir sus credenciales (normalmente la contraseña) para poder autenticarse en el equipo. Una vez autenticado, si el usuario quiere acceder a un archivo específico, el sistema debe garantizar que dicho usuario está autorizado a acceder, y en caso afirmativo, el sistema debe proporcionárselo. El concepto de autenticación y autorización está muy ligado con el concepto de control de accesos que se expondrá más adelante.

Normalmente se suele dejar de lado el proceso de auditoría, sin embargo, este es de vital importancia para el correcto funcionamiento de un sistema de gestión de identidades. Este proceso se encarga de revisar el histórico de interacciones entre las diferentes identidades con un sistema, con el objetivo de detectar anomalías y por consiguiente desplegar las contramedidas necesarias, como por ejemplo levantar una alerta en el sistema. Esto se debe a que dicha anomalía puede haber comprometido el sistema produciendo daños en el mismo.

Los sistemas de gestión de identidades tienen como objetivo fundamental garantizar los flujos de IAAA. Estos sistemas, han ido evolucionando en el tiempo incorporando nuevas funcionalidades y nuevos modelos y/o arquitecturas para lograr de forma efectiva estos procesos. Además, estos sistemas han de garantizar en todo momento la privacidad y la seguridad de las entidades que manejan [41]. Los requisitos fundamentales de un sistema de gestión de identidades son [12]:

- **Confidencialidad:** La información sobre cada identidad debe ser únicamente accedida por las entidades autorizadas a ello.
- **Revocación:** Un usuario ha de poder solicitar que el sistema deje de almacenar la información asociada a su identidad digital.
- **Integridad:** La información almacenada en un sistema no debe ser alterada por alguien no autorizado.
- **Anonimato:** La información sobre una identidad digital debe ser privada y no se ha de poder asociar con otra identidad digital con el objetivo de obtener información extra.
- **Libre elección:** El usuario debe poder elegir entre múltiples proveedores de identidades.
- **Verificación:** El usuario debe poder verificar que la información almacenada por el proveedor de identidades es correcta.

- **No repudio:** Un usuario no puede negar su participación en una operación, pues el sistema tiene almacenado evidencias de las operaciones en las que ha participado las diversas identidades.
- **Anti-fraude:** Un agente externo no ha de poder realizar operaciones suplantando una identidad o robando las credenciales. Para ello se han de establecer los mecanismos de seguridad para evitar estas situaciones.
- **Mínima información:** El sistema no ha de almacenar más información de la necesaria sobre una identidad.

Con el objetivo de poder comprender las secciones posteriores es conveniente definir los agentes principales que intervienen en los sistemas de gestión de identidades:

- **Entidad:** puede ser un usuario, dispositivo, organización, sensor o agente software que inicia cualquier flujo con los sistemas de gestión de identidades.
- **Identidad**: Es el conjunto de características que poseen las entidades y permiten diferenciarlas entre ellas.
- **Service provider (SP):** comúnmente también conocido como Relaying Party (RP), es el agente que aloja un recurso, servicio o aplicación la cual solicita que se comprueben los procesos IAAA sobre una entidad.
- **Proveedor de Identidades (IdP):** es el encargado de comprobar los procesos IAAA sobre una entidad. Almacenan toda la información y características relacionadas con las identidades.

Los sistemas de IdM se pueden categorizar en 3 niveles de acuerdo con su arquitectura: modelo Silo, modelos centralizados y modelos federados. Para poder comprender estos sistemas, es también de interés exponer previamente el concepto de control de accesos.

2.1.1 Control de accesos

En los inicios, los sistemas informáticos estaban pensando para que un único operador tuviese acceso a dicho sistema. Este operador era la única persona capaz de poder interactuar con el equipo, pues era el único usuario disponible. Sin embargo, este hecho no exentaba a que el usuario se tuviese que autenticar y autorizar sobre el sistema. Poco a poco empezaron a proliferar los sistemas multiusuario, en el que varios usuarios interactuaban simultáneamente con el mismo sistema informático creyendo disponer de la máquina en su totalidad, sin embargo, estaban compitiendo por el reparto en tiempo y espacio de los recursos disponibles. Es en este momento cuando los desarrolladores de los sistemas informáticos empezaron a percatarse de la importancia de desarrollar sistemas de identificación de usuarios sólidos y consistentes para garantizar la seguridad de dichos sistemas, dejando acceder únicamente a los recursos a los usuarios legítimos y denegando el acceso a los usuarios no legítimos.

El control de acceso es el proceso que se encarga de permitir o denegar el acceso a un objeto (recurso, servicio o aplicación) en base a unas políticas de acceso. Las políticas de acceso no son más que las reglas que determinan los privilegios de una entidad

sobre un objeto. Este proceso se integra con los mecanismos de IAAA. Todo sistema de control de accesos debe garantizar los siguientes principios [42]:

- **Integridad**: El sistema no puede ser alterado o manipulado por ningún tercero. En caso de ser manipulado, debería ser posible detectarlo en los procesos de auditoria.
- **No eludible**: El sistema debe garantizar que todas las peticiones de acceso sean tratadas y por lo tanto nadie pueda saltarse las políticas de acceso.
- **Seguridad central**: La seguridad debe concentrarse en un núcleo y no estar dispersa por el sistema informático.
- **Tamaño pequeño**: El sistema debe ser lo suficientemente pequeño como para ser usable, testeado y manejable.

Al comienzo de los sistemas de control de acceso surgieron dos tipos de soluciones: el control de acceso obligatorio (MAC, de sus siglas en ingles Mandatory Access Control), el control de acceso discrecional (DAC, de sus siglas en ingles Discretionary Access Control).

Los sistemas de control de acceso mandatorio se basan en políticas [42]. El sistema se encarga de evaluar los atributos tanto del objeto como del usuario para determinar si la petición de acceso es legítima. Normalmente se suelen usar niveles de seguridad, es decir, se categorizan los usuarios y objetos por niveles y se define que categorías de usuarios pueden realizar qué acciones sobre que objetos. Un ejemplo típico es el sistema de clasificación de archivos de la NSA, en el que un archivo puede ser categorizado en niveles, como por ejemplo archivo confidencial, secreto o archivo de máximo secreto. El sistema únicamente permite el acceso a los archivos confidenciales a los usuarios con un rango alto, denegándoselo a los usuarios corrientes, a los usuarios de un rango mayor les permite acceder a archivos confidenciales y secretos y para los de mayor rango les permite acceder a todos los tipos de archivo. Note que, las políticas de acceso están centralizadas, es decir, el sistema almacena el conjunto de reglas y es un administrador el encargado de actualizarlas para lograr los requisitos deseados.

Los sistemas de control discrecionales [13] fueron definidos por *Trusted Computer System Evaluation Criteria*. Este tipo de sistemas se basan en que son los propios usuarios los que definen el conjunto de reglas (permisos) sobre los objetos que son propietarios, es decir, un usuario propietario transfiere los permisos necesarios a otros usuarios para cumplir los requisitos deseados. Note que este tipo de sistemas son descentralizados, pues son los propios usuarios los encargados de generar los permisos. Este tipo de sistemas son los típicos utilizados por GNU/Linux.

Existen dos formas de almacenar el conjunto de reglas de acceso. En primer lugar por medio del modelo HRU (Harrison, Ruzzo y Ullman) [39] o en segundo lugar, por medio de las listas de control de accesos [40].

El modelo HRU se basa en implementar una matriz en la que cada fila viene definida por cada usuario del sistema. En cada columna de la matriz se representa todos los

usuarios del sistema y todos los objetos del mismo (recursos). El valor para cada campo de la matriz viene dado por la acción que el usuario tiene permitido realizar sobre el recurso o usuario, el cual puede tomar los valores “Lectura”, “Escritura”, “Ejecución” y “Propiedad”. La ilustración 3 muestra un ejemplo del modelo HRU.

	Archivo 1	Archivo 2	Ejecutable 1	….	Objeto M
Usuario 1	Acción1…AcciónK	Acción1…AcciónK	Acción1…AcciónK		
Usuario 2	Acción1…AcciónK	Acción1…AcciónK	Acción1…AcciónK		
….					
Usuario N					

Ilustración 3: Modelo HRU

Las listas de control de acceso (ACLs, de sus siglas en ingles Access Control Lists) son la evolución de la representación del modelo HRU. En esta ocasión, cada usuario y objeto (recurso) del sistema vienen definidos por un elemento de la lista. Una conexión entre dos elementos significa que el primer elemento puede realizar las acciones indicadas sobre el segundo elemento (ver ilustración 4).

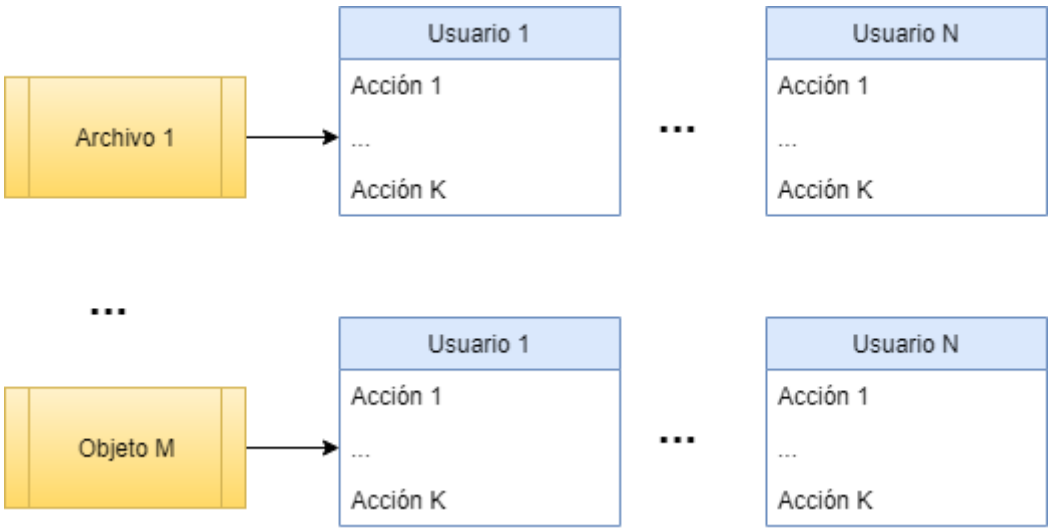

Ilustración 4: Listas de control de acceso

A raíz de los sistemas MAC y DAC empezaron a surgir múltiples sistemas que implementaban las funcionalidades de uno o ambos sistemas de una forma u otra y añadiendo nuevos factores para conseguir solventar el control de accesos [30]. A continuación, se describen los más importantes.

El control de accesos basado en roles (RBAC, de sus siglas en inglés Role-Based Access Control) se basa en asignar conjuntos de roles a los que se les permite realizar ciertas operaciones. A los diferentes usuarios del sistema se les añade uno o más roles que definen las operaciones que pueden realizar sobre los objetos del sistema. Un ejemplo de rol podría ser el rol jefe, el cual puede realizar todo tipo de operaciones sobre todos los objetos, mientras que el rol usuario solo puede realizar las operaciones básicas sobre ciertos objetos no relevantes. Gracias a esto, el manejo de las identidades se convierte en una tarea sencilla en la que únicamente se han de ir definiendo los roles

necesarios y añadiéndoselos a los usuarios correspondientes. Note que el concepto de grupo de usuarios y el de rol difieren, pues el primero es una colección de usuarios mientras que el segundo es una colección de permisos. Este tipo de sistemas de control de accesos puede implementar tanto MAC como DAC.

El control de accesos basado en atributos (ABAC, de sus siglas en inglés Attribute-Based Access Control) se basa en definir tres tipos de atributos: atributos de usuario, atributos del objeto y atributos del entorno. Una vez definidos estos atributos, se elabora una política de accesos en base a estos atributos. Por ejemplo, supóngase un usuario que tiene como atributo jefe. Este usuario intenta acceder a un objeto que tiene como atributo confidencial. Puesto que en las políticas de acceso se ha definido que los usuarios con atributo jefe pueden acceder a los archivos confidenciales se le debería dar acceso, sin embargo, el atributo de entorno indica que dicho usuario está intentando acceder fuera del horario permitido. En este caso, el acceso es revocado. Note que este tipo de sistemas es capaz de implementar y mejorar los sistemas MAC, DAC y RBAC.

En [30] describen muchos más sistemas de control de accesos. Además, siempre hay que tener en cuenta que estos sistemas no son excluyentes entre sí, por lo que siempre se pueden implementar combinaciones de todos ellos con el objetivo de lograr los requisitos de seguridad del sistema. Note que, en muchas ocasiones tener un sistema de control de accesos complejo puede perjudicar la usabilidad y rendimiento del sistema obteniendo políticas de seguridad muy elaboradas y difíciles de mantener. Es por esto, que hay que intentar buscar un equilibrio entre todos estos factores logrando siempre un sistema seguro que sea lo más sencillo posible.

2.2. Modelo Silo

El modelo Silo fue la primera arquitectura que se creó para manejar las identidades digitales, es decir, fue la primera arquitectura para solventar el IdM [31]. En esta arquitectura el sistema de gestión de identidades posee dos funcionalidades principales. En primer lugar, actúa de RP proveyendo de un servicio a los usuarios que quieran acceder a él. En segundo lugar, tiene la funcionalidad de hacer de IdP, es decir, tiene la responsabilidad de almacenar todas las identidades digitales del sistema, junto a sus credenciales, y es el encargado de validar dichas credenciales para poder realizar los flujos de IAAA. Note que para este modelo, la figura del SP y el IdP se solapan convirtiéndose únicamente en un agente, es decir, el proveedor del servicio y el sistema de gestión de identidades son lo mismo. Este sistema permite realizar cualquier flujo de IAAA entre un servicio y una entidad. En la ilustración 5 se puede observar el funcionamiento de este modelo.

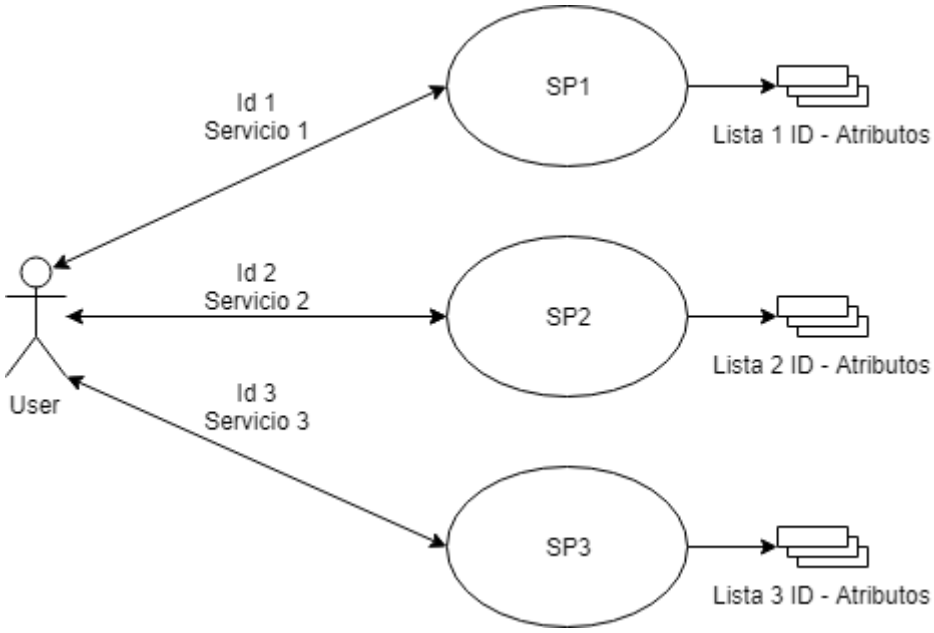

Ilustración 5: Modelo Silo

El principal inconveniente de este modelo es que puesto que los SP son independientes entre sí, un mismo usuario ha de gestionar múltiples identidades a lo largo de todos los servicios para poder realizar los flujos de IAAA con su respectivo IdP. Tomar como ejemplo un usuario que se quiere dar de alta en dos servicios web, en ambos servicios ha de introducir un nombre de usuario y unas credenciales, que aunque puedan coincidir no existe la condición de que sean idénticas. Esto se traduce en que dicho usuario posee dos identidades digitales distintas y cada una de ellas le sirve para poder realizar los flujos de IAAA sobre servicios distintos.

Esta arquitectura es la más antigua que se implementó para este propósito ya que, entonces existía un número razonable de servicios y por lo tanto un mismo usuario podía encargarse de gestionar múltiples identidades a lo largo de todos los servicios. Sin embargo, a medida que el número de servicios ofertados ha ido aumentando drásticamente, estos sistemas han ido poniendo más impedimentos en su usabilidad, dando paso a nuevas arquitecturas. Cabe destacar, que hoy en día estos sistemas siguen siendo muy utilizados.

2.3. Sistemas de Identidades Centralizados

Los sistemas de gestión de identidades centralizados surgieron para solventar los problemas del modelo Silo, unificando en un solo punto la gestión de identidades para múltiples servicios [31]. A diferencia del modelo Silo, en el sistema de gestión de identidades centralizado el IdP se desvincula de la RP convirtiéndose en un agente independiente. El IdP se encarga de almacenar toda la información acerca de las identidades digitales que interactúan con múltiples RP. La RP delega todo el proceso de IAAA al IdP, es decir, cuando un usuario inicia un flujo de IAAA sobre un SP, este no lo inicia con la propia SP sino que se comunica directamente con el IdP. Esto permite que un mismo usuario pueda realizar los flujos de IAAA sobre múltiples servicios utilizando una identidad única a lo largo de todos los servicios. Sin embargo, esto

también puede convertirse en una debilidad, pues en caso de sufrir un robo de credenciales, el atacante tendrá acceso total a todos los servicios a los que responde dicha identidad.

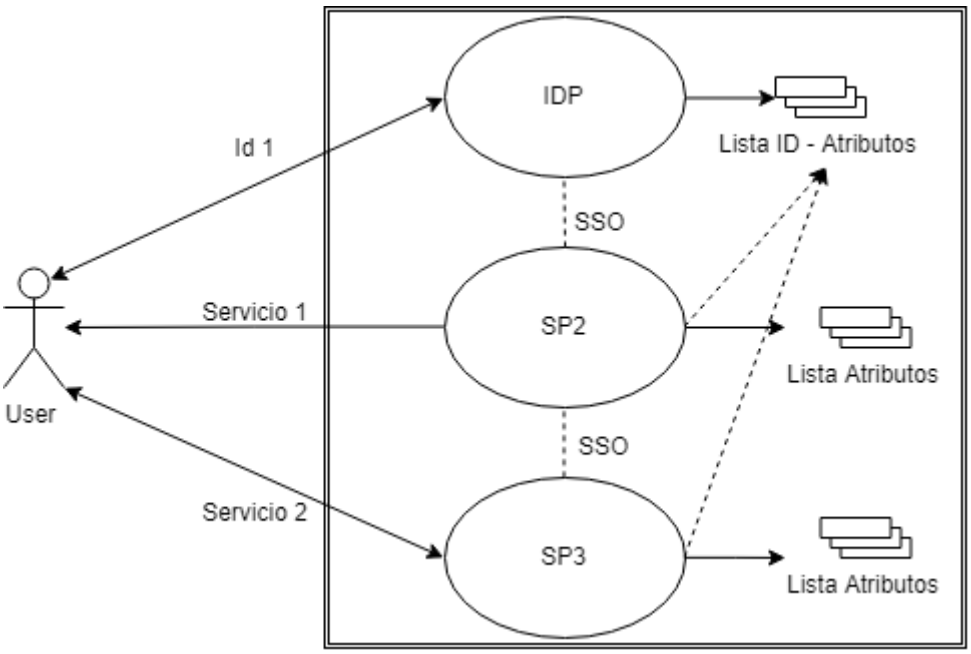

Ilustración 6: Modelo Centralizado

El funcionamiento de este tipo de sistemas se puede observar en la ilustración 6. Cuando un usuario inicia un flujo de IAAA sobre un SP específico, este es redirigido al IdP, el cual se encargara de verificar las credenciales proporcionadas. Una vez este proceso ha sido verificado, el usuario tendrá acceso al servicio del SP, así como al resto de servicios bajo los que opera dicho IdP.

La manera más simple de implementar un sistema de gestión de identidades centralizado es utilizando Single Sign On (SSO). Un sistema SSO sirve para poder realizar los flujos de IAAA de un grupo de usuarios en diferentes servicios a través de un único punto de acceso común a todos ellos. Estos sistemas permiten que un usuario introduzca sus credenciales una única vez y con ello obtenga acceso a múltiples recursos o servicios alojados en diferentes RP. La forma más intuitiva de implementar un sistema SSO es por medio de una base de datos centralizada que contenga las identidades de todos los servicios, así como sus credenciales.

Existen múltiples estándares y protocolos de sistemas de gestiones de identidades centralizados. En el presente documento se presentan los más destacados: LDAP, Kerberos y Radius.

2.3.1. LDAP

La primera aproximación de los sistemas de gestión de identidades centralizados surgió en el año 1888 cuando la ITU [24] desarrollo el estándar X.500 [3]. Este estándar cubría los servicios de directorios, los cuales son bases de datos de direcciones electrónicas en forma de aplicación que almacenan y organizan la información acerca de los usuarios de una red y permite administrar y gestionar el acceso a los recursos

alojados en el servidor Se basan en organizar los directorios de manera jerárquica. X.500 define el protocolo DAP (Directory Access Protocol) para que el cliente y el servidor de autenticación puedan comunicarse. Este protocolo se define a nivel de aplicación por lo que el cliente y el servidor han de implementar correctamente la torre de protocolos OSI [15]. Además, DAP es bastante pesado al tener el requisito de operar a nivel de aplicación, por lo que surgió su variante LDAP.

LDAP (Lightweight Directory Access Protocol) [7] es un protocolo de aplicación sobre TCP/IP que permite el acceso a un servicio de directorios. LDAP no utiliza todos los protocolos del modelo OSI, por lo que es mucho más eficiente. Además LDAP está basado en la arquitectura Cliente-Servidor [33] haciendo uso de APIs. Esto se debe a que si una aplicación desea acceder a la base de datos donde se almacena la información de roles y privilegios, esto no es posible y tendrá que utilizar las funciones de la API habilitadas para ello, aumentando así la seguridad.

En LDAP se distinguen 3 tipos de elementos:

1. **Entradas**: Son las unidades básicas de LDAP. Cada una de ellas define un concepto u objeto del mundo real, es decir, usuarios, organizaciones o hosts. Puesto que LDAP está jerarquizado, cada entrada está definida de forma relativa a un nodo padre. La representación de todas las entradas de forma organizada se denomina Directory Information Tree (DIT).
2. **Atributos:** Cada entrada posee atributos, que no son más que pares de nombre del atributo y su valor. Cabe destacar, que cada atributo puede tener múltiples valores asignados.
3. **Clases:** Definen los atributos que ha de poseer o deben poseer, una entrada determinada. Toda entrada debe tener asignada obligatoriamente, al menos una clase.

El funcionamiento de LDAP se puede observar en la ilustración 7:

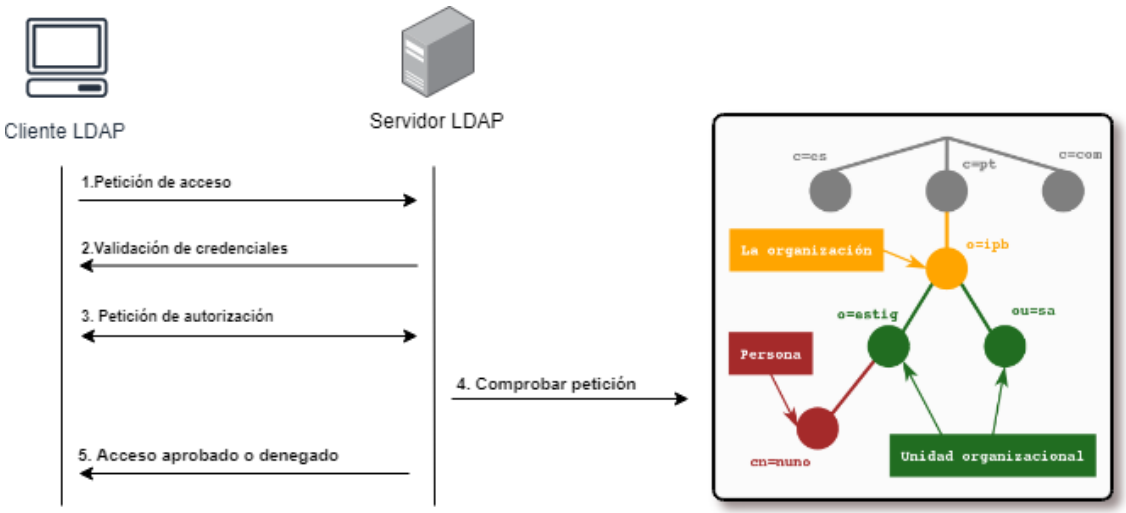

Ilustración 7: LDAP

2.3.2. Kerberos

Kerberos [34] es un protocolo de autenticación de red creado por el MIT. Actualmente su versión actual es la krb5-1.16 y trabaja sobre el puerto UDP 88. Está basado en una arquitectura cliente-servidor y permite la autenticación de ambos agentes.

Kerberos está construido basado en criptografía simétrica. El servidor de Kerberos construye una base de datos que contiene las claves secretas de los agentes y validadas por un tercero de confianza. Para poder generar esta base de datos se requiere que los servicios de red que necesiten su uso se registren en kerberos, al igual que los clientes que desean utilizar estos servicios. La autenticación y/o autorización es posible gracias al uso de tickets. Los clientes presentan dichos tickets generados por el servidor de kerberos para validar su identidad.

Una vez construida la base de datos, kerberos proporciona tres niveles de protección. En primer lugar el programador de la aplicación determina el nivel de seguridad que se desea de acorde a los requisitos de la aplicación. En segundo lugar, proporciona un mecanismo para transmitir mensajes seguros, garantizando la autenticidad del mismo pero sin incluir el contenido del mensaje. Por último, proporciona mecanismos para garantizar los mensajes privados, donde cada mensaje no solo se autentica sino que también se cifra para que pueda viajar por red de forma más segura.

El funcionamiento de kerberos se puede observar en la ilustración 8:

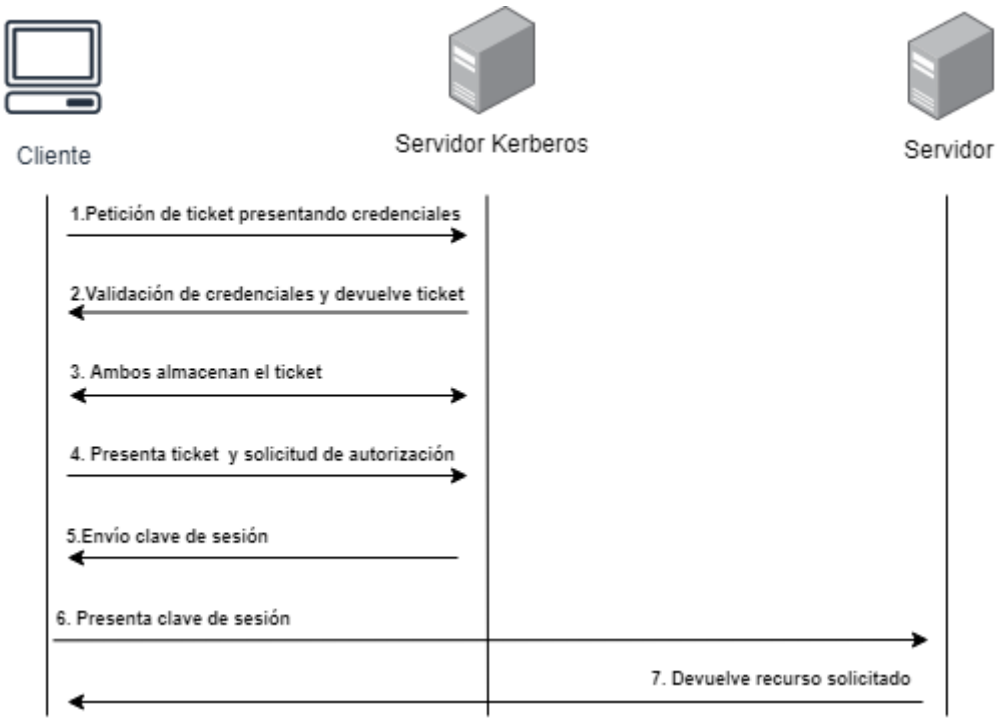

Ilustración 8: Kerberos

2.3.3. Radius

Radius (Remote Authentication Dial In User Server) [38] es un protocolo desarrollado por Livingston Enterprises, Inc. para gestionar la autenticación, autorización y registro de usuarios remotos sobre un determinado recurso. Note que autenticación, autorización y registro se corresponden con las tres "A" de las siglas IAAA. Radius surgió para administrar el control de acceso de usuarios en dispositivos tales como módems, routers y switches. Radius es la opción recomendada para solventar servicios de autenticación en redes inalámbricas, según la especificación IEEE 802.11.

Radius está basado en una arquitectura cliente-servidor, donde el servidor es el servidor Radius que contiene toda la información acerca de los usuarios y los atributos almacenados en el sistema para garantizar los procesos de autenticación autorización y registro, y un NAS (Networks Acesss Server) actuara como cliente que mediara entre el usuario final y el servidor NAS haciendo de punto intermedio en la comunicación entre ambos. Esta comunicación es segura gracias al uso de un secreto compartido y que nunca ha viajado por la red por lo que se puede garantizar la confidencialidad de la comunicación.

El funcionamiento de Radius se puede observar en la ilustración 9:

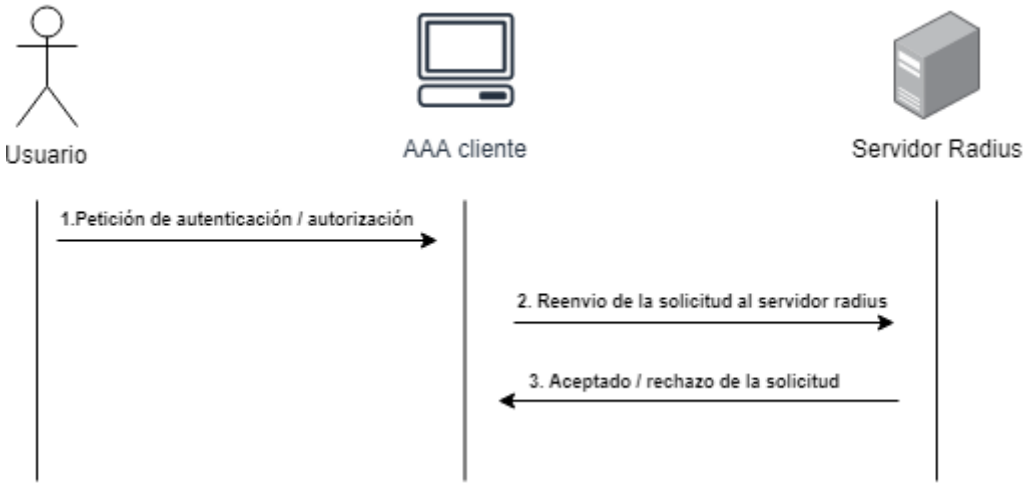

Ilustración 9: Radius

Capítulo 3: Sistemas de Gestión de Identidades Federados

En este capítulo se introducen los sistemas de gestión de identidades federados. El objetivo principal es comprender el motivo por el cual surgen este tipo de sistemas, discutir las principales ventajas e inconvenientes que presentan este tipo de sistemas frente a los expuestos anteriormente y finalmente exponer los principales estándares desarrollados en la actualidad.

3.1. Federación de Identidades

La federación de identidades es el conjunto de tecnologías y estándares que permiten distribuir de forma dinámica las identidades y su información a lo largo de múltiples dominios seguros [32]. Estos modelos de gestión de identidades permiten que un usuario pueda realizar SSO a lo largo de múltiples dominios, es decir, permiten que un usuario pueda autenticarse una única vez en un dominio e inmediatamente tenga acceso a recursos protegidos de otros SP aunque estos estén en diferente dominio. Esto es posibles gracias que se crean vínculos de confianza entre los SP y los IdP para poder delegar el proceso de autenticación a un agente externo. Estas relaciones de confianza entre se denominan círculos de confianza CoT (Circle of Trust). El IdP en este caso es capaz de crear seudónimos entre las identidades que maneja una entidad a lo largo de múltiples SP. En la ilustración 10 se puede observar el funcionamiento de estos modelos.

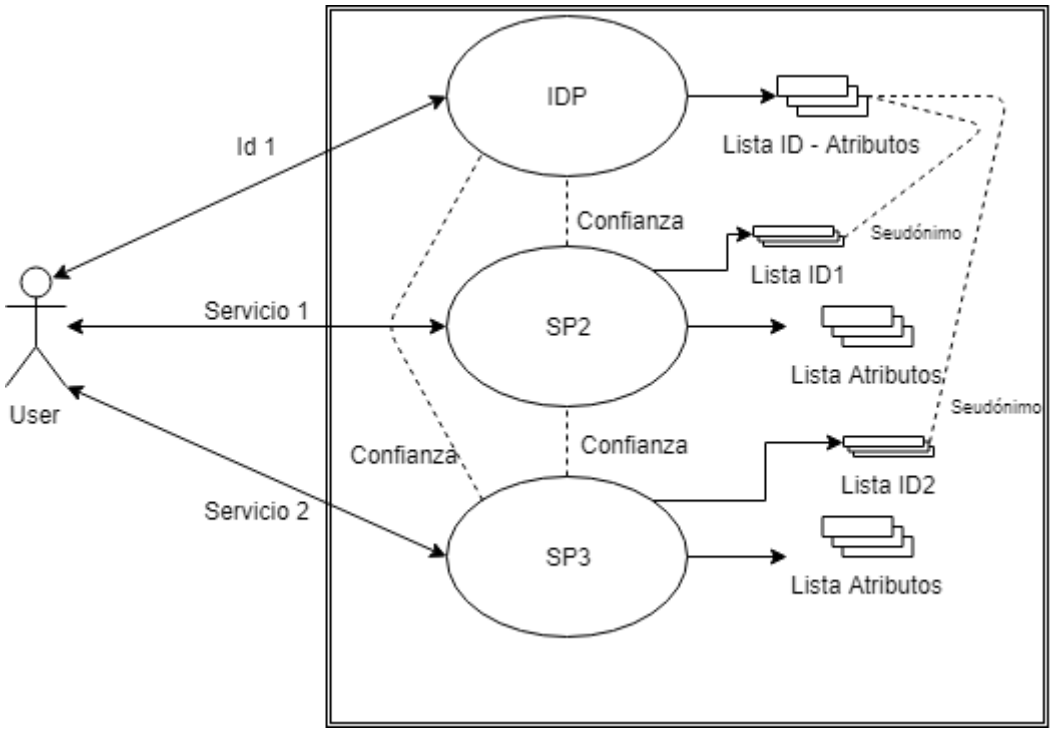

Ilustración 10: Federación de identidades

Este tipo de modelos son bastante recientes ya que el primer estándar que se ha extendido surgió en 2003. En el presente documento se exponen los principales estándares y protocoles que han surgido para resolver la gestión de identidades federadas: SAML, OAuth, OpenIdConnect y MobileConnect.

3.2. SAML

SAML (Security Assertion Markup Language) [8] se creó en el año 2003 y es un producto del comité OASIS (Organization for the Advancement of Structured Information Standards). Actualmente se encuentra en la versión 2.0 que fue aprobada en 2005.

SAML es una arquitectura que define un estándar a la hora de intercambiar información de autenticación y autorización entre dos entidades. Más concretamente define como se ha de codificar, transportar e interpretar la información referente a una identidad, entre un IdP y un SP, para lograr completar un proceso de autenticación o autorización. Este esquema está basado en el lenguaje de marcado XML (Extensible Markup Language) [28] para crear la estructura del mensaje (codificar) y HTTP (Hypertext Transfer Protocol) [4] o SOAP (Simple Object Access Protocol) [1] para transportarlo entre un emisor y un receptor.

3.2.1. Conceptos fundamentales

El estándar de SAML distingue entre cuatro tipos de elementos (ver ilustración 11):

- **Asertos (Assertions)**: contienen información referente a la propia identidad, autenticación y autorización de un determinado sujeto. Están codificados en XML y contienen toda la información necesaria para poder autenticar o autorizar a un usuario.
- **Protocolos (Protocols)**: definen el proceso de solicitud de un aserto por parte de un SP y el proceso de envió de un aserto por parte del Idp.
- **Regulaciónes (Bindings)**: define como transportar un aserto utilizando los protocolos HTTP o SOAP.
- **Perfiles (Profiles)**: definen el conjunto de asertos, protocolos y regulaciones que se pueden combinar para poder satisfacer las casuísticas que surgen al realizar los flujos de SAML.

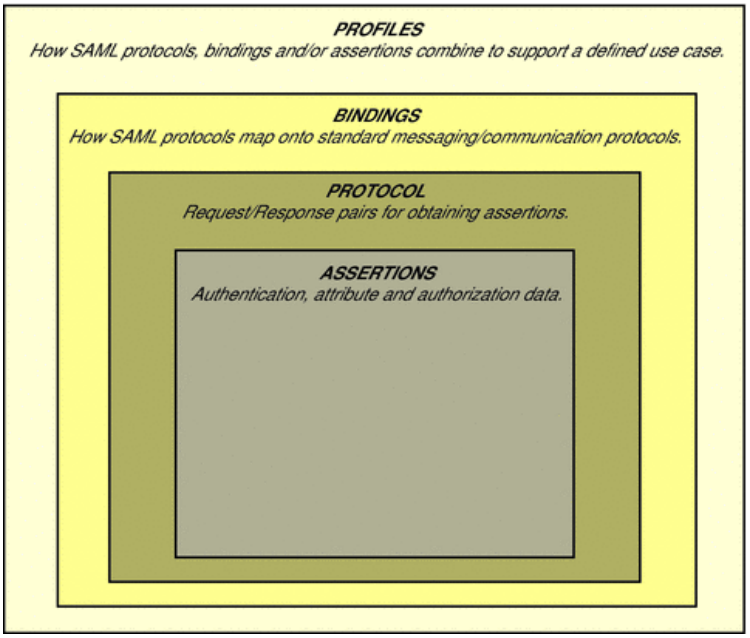

Ilustración 11: Elementos SAML

Cada uno de los elementos expuestos anteriormente tiene a su vez varios subtipos de acuerdo con su naturaleza.

En primer lugar, SAML distingue entre tres tipos de asertos:

- **Autenticación**: Son expedidas por el IdP y contienen información acerca del sujeto autenticado, el periodo de validez, la versión de SAML utilizada y el instante en el que se ha validado entre otros.
- **Atributos**: Contienen información relevante sobre un sujeto determinado que quiere autenticarse o autorizarse.
- **Autorización**: Contienen información acerca de los privilegios y roles que posee un sujeto, es decir, contienen información acerca de las acciones y accesos que tiene un usuario sobre un recurso o servicio.

En segundo lugar, SAML distingue entre 6 tipos de protocolos:

- **Authentication Request Protocol:** Este protocolo define como han de intercambiar información el SP y el IdP para poder autenticar a un usuario. En primer lugar, el SP debe solicitar al IdP que autentique a un usuario mediante una *AuthnRequest* en la que se incluye los requisitos para poder autenticar a dicho usuario. Posteriormente, el IdP solicita las credenciales al usuario de acuerdo con los requisitos expuestos en la *AuthnRequest* y debe contestar al SP proporcionándole si el proceso de autenticación ha sido completado con éxito. Este último aserto del tipo autenticación se denomina *Response* y no tiene por qué ser un único mensaje.
- **Assertion Query and Request Protocol:** Este protocolo sirve para recuperar un aserto ya en uso. Lo utiliza el SP enviando el identificador del aserto mediante un *AssertionIdRequest* al IdP, el cual almacena el histórico de asertos en uso. En el *AssertionIdRequest* se pueden incluir consultas específicas para recuperar información acerca del sujeto (*SubjectQuery)*, los asertos de autenticación disponibles para un sujeto (*AuthnQuery)*, los atributos que contienen los

asertos sobre un sujeto (*AttributeQuery*) y los permisos que tiene autorizados realizar un sujeto en base a los asertos disponibles (*AuthzDecisionQuery*).

- **Artifact Resolution Protocol:** Este protocolo define un mecanismo para poder transportar los mensajes de SAML por referencia en vez de por valor. Este protocolo se implementa de tal forma que el emisor del mensaje envía únicamente un pequeño trozo del mensaje original denominado *artifact* utilizando un elemento de regulación especial. De esta forma, el receptor ha de ser capaz de determinar quién ha sido el agente emisor de dicho *artifact*. En caso de que el mensaje sea de interés para el receptor procede a solicitar el mensaje completo al emisor utilizando las regulaciones habituales. Este protocolo se suele utilizar cuando un mensaje de SAML no se puede transportar de la forma habitual, en tal caso se envía un *artifact* y se fijan canales de comunicación alternativos para poder enviar correctamente el mensaje.
- **Single Logout Protocol:** Este protocolo provee los mecanismos necesarios para poder eliminar todos los asertos de autenticación o autorización sobre un sujeto concreto, es decir, permite el cierre de sesión único para un sujeto.
- **Name Identifier Mapping Protocol:** Este protocolo define como asociar los diferentes espacios de nombres de una entidad para poder identificarlas.
- **Name Identifier Management Protocol:** Este protocolo define el proceso para que un IdP o SP pueda notificar al contrario, el cambio del identificador de un recurso o servicio.

En tercer lugar, SAML distingue entre 5 tipos de regulaciones:

- **SAML SOAP Binding**: Define como se han de transportar los mensajes bajo el protocolo SOAP 1.1 haciendo uso de HTTP.
- **REVERSE SOAP Binding**: Define como cumplir con la especificación PAOS [37] para poder utilizar "Reverse HTTP" en el intercambio de mensajes de SAML.
- **HTTP Redirect Binding:** Define como usar mensajes de redirección HTTP [4] para transportar mensajes SAML.
- **HTTP POST Binding:** Especifica cómo utilizar los formularios HTML en base64 para transportar mensajes SAML.
- **HTTP Artifact Binding:** Especifica cómo utilizar formularios HTML o una URI para transportar un Artifact.
- **SAML URI Binding:** Especifica cómo utilizar las URI para identificar un aserto**.**

Por último, SAML define 8 tipos de perfiles:

- **Web Browser SSO Profile:** Este perfil proporciona un mecanismo SSO mediante un navegador web utilizando un aserto de autenticación en combinación con las regulaciones de HTTP Redirect, HTTP Post y HTTP Artifact.
- **Assertion Query/Request Profile:** Este perfil define el uso de la regulación con mismo nombre haciendo uso de SOAP.
- **Artifact Resolution Profile:** Este perfil define el uso de la regulación con mismo nombre haciendo uso de SOAP.
- **Enhanced Client and Proxy (ECP) Profile:** Este perfil esta basado en SAML Authentication Request Protocol junto a la especificación PAOS y sirve para poder averiguar el IdP apropiado.

- **Identity Provider Discovery Profile:** Este perfil define los medios para que un SP descubra los IdP en los que un usuario esta dado de alta y poder solicitárselos en el momento de iniciar el flujo de autenticación o autorización.
- **Name Identifier Mapping Profile:** Este perfil define el uso de la regulación con mismo nombre haciendo uso de SOAP, HTTP Redirect, HTTP POST o Artifact.

- **Name Identifier Management Profile:** Este perfil define el uso de la regulación con mismo nombre haciendo uso de SOAP, HTTP Redirect, HTTP POST o Artifact.

3.2.2. Flujo de información

SAML se puede utilizar para implementar simples sistemas SSO, ECP y sistemas federados. En el presente documento se va a exponer como es el flujo de datos, es decir, como se produce el intercambio de mensajes e información entre el IdP, SP y el usuario en un sistema de gestión de identidades federado utilizando SAML (ver ilustración 12).

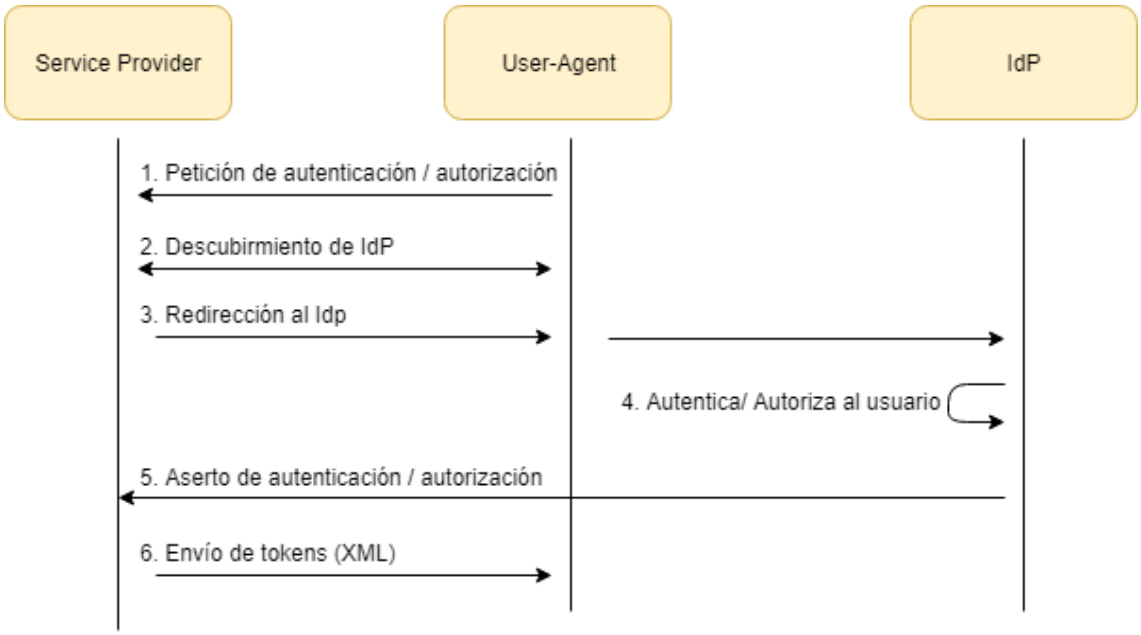

Ilustración 12: Flujo SAML

En primer lugar (1), el usuario que está navegando por un servicio web trata de autenticarse o autorizarse sobre el SP. En este instante (2), el SP hace uso del Identity Provider Discovery Profile para determinar cuál son los IdP en los que está dado de alta dicho usuario. Posteriormente (3), el SP redirige al usuario al IdP escogido para que pueda introducir sus credenciales y le envía una *AuthnRequest* por HTTP POST a dicho IdP con las condiciones para que se complete correctamente el proceso. En el siguiente paso (4), el IdP determina que las credenciales introducidas son correctas y procede a enviar un aserto de autenticación al SP (5). Finalmente, el SP posee verificación de que el usuario se ha autenticado correctamente, por lo que procede a enviarle los tokens correspondientes para que el usuario pueda acceder a los recursos

que solicite y esté autorizado (6). Note que el intercambio de información y mensajes se ha de producir de acuerdo con los asertos, protocolos y regulaciones definidas en el apartado anterior y que pueden ser diferentes a los expuestos en este ejemplo siempre y cuando se cumpla esta condición.

3.3. OAuth

OAuth es un framework open-source de sistemas de gestión de identidades federados iniciado en el año 2006. No fue hasta el año 2010 cuando surgió su primera versión estable OAuth 1.0 publicado como RFC 5849 [20]. Actualmente se encuentra en su versión OAuth 2.0 [21]. Este framework provee los mecanismos necesarios para poder solventar únicamente la autorización en aplicaciones web, aplicaciones de escritorio y dispositivos inteligentes.

El proceso de autorización se realiza de forma federada, pues el SP delega todo el proceso de verificación de roles y privilegios sobre el IdP. Cuando un usuario, desde un cliente (e.g, navegador web) realiza una solicitud de acceso a un recurso protegido alojado en el SP, este le redirige al IdP el cual decide si dicha petición es legítima o no y en caso afirmativo le garantiza acceso al usuario. Note que este proceso no incluye en ningún momento un proceso de autenticación, por lo que OAuth asume que este proceso se realiza por otra vía como por ejemplo de forma local con un modelo Silo o con un modelo centralizado o de forma federada con SAML, OpenID, OpenIdConnect o MobileConnect,

Puesto que la versión OAuth 1.0 se encuentra obsoleta, en el presente documento se va a exponer el funcionamiento de OAuth 2.0

3.3.1. Conceptos fundamentales

OAuth 2.0 define 7 conceptos fundamentales:

- **Resource Owner**: Es la entidad propietaria del recurso protegido y la encargada de garantizar el acceso al recurso.
- **Resource server:** Es el servidor que aloja el recurso protegido. Tiene que ser capaz de aceptar y responder correctamente las peticiones de acceso a los recursos protegidos haciendo uso de los access tokens.
- **Client:** Es una aplicación que solicita acceso a los recursos protegidos. Por ejemplo, puede ser el navegador web que utiliza el usuario para solicitar acceder a un recurso protegido de una aplicación web (Resource server). Pueden ser confidenciales (e.g., una cliente protegido que aloja las credenciales de forma segura) o públicos (e.g., una aplicación que utiliza el propio dispositivo del usuario para almacenar las credenciales) dependiendo de su capacidad para mantener la confidencialidad de las credenciales del usuario.
- **Authorization Server:** Es el servidor de verificar los privilegios y roles de un cliente para acceder a un recurso protegido. Además, genera los Access tokens.
- **Access Token**: Es el valor que utiliza el cliente para poder acceder a los recursos protegidos alojados en el Resource Server, es decir, es el valor que sustituye a

las credenciales de usuario y que garantiza el flujo de autorización. Es generado por el Authorization Server.

- **Refresh Token:** Es el valor que utiliza el cliente para solicitar un nuevo acces stoken al Authorization Server cuando este último queda invalidado.

OAuth 2.0 está pensado para trabajar con el protocolo HTTP. Esto quiere decir, que el intercambio de información entre los agentes ha de ocurrir por medio de las cabeceras de HTTP. Además, se hace uso de las redirecciones HTTP para que el cliente o el Authorization server puedan redirigir correctamente las peticiones entrantes al agente deseado. Por otro lado, OAuth está pensado para utilizar TLS (Transport Layer Security) protegiendo así las comunicaciones entre los distintos agentes que intervienen en el flujo.

3.3.2. Flujo de Información

En la ilustración 13 se presenta el flujo de información básico de un proceso de autorización de OAuth 2.0:

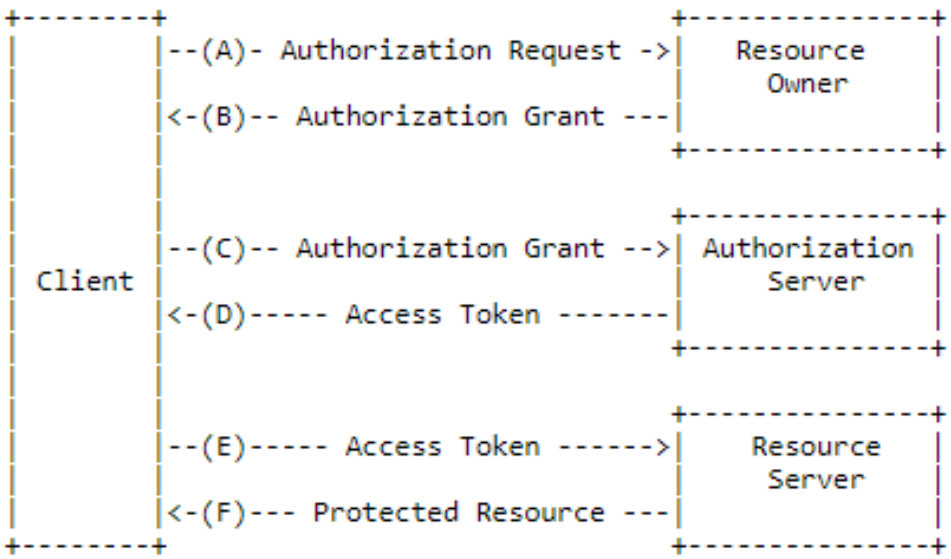

Ilustración 13: Flujo básico OAuth 2.0
(Fuente: https://tools.ietf.org/html/rfc6749)

(A) En primer lugar, el cliente solicita autorización al propietario del recurso protegido.
(B) El cliente recibe las credenciales (Authorization Grant) que le otorgan la autorización por parte del propietario para poder acceder al recurso protegido.
(C) El cliente utiliza las credenciales obtenidas en el paso B para solicitar al Authorization server el Access token.
(D) El Authorization Server comprueba que las credenciales proporcionadas por el cliente son válidas y en caso afirmativo le responde un el access token.
(E) El cliente utiliza el acces token recibido para solicitar al Resource server acceder al recurso protegido.

(F) El Resource Server comprueba que el acces token es válido y en caso afirmativo le devuelve el recurso protegido solicitado al cliente.

Este es el flujo de información más básico que propone OAuth 2.0 para lograr la autorización sobre un recurso protegido. Sin embargo, existen múltiples casuísticas que pueden variar ligeramente este flujo y que son interesantes de analizar.

En múltiples ocasiones es interesante considerar que los access tokens tienen un periodo de validez durante el cual un cliente puede solicitar acceso a los recursos protegidos pero que pasado dicho periodo ese access token queda invalido. Esto se debe a que si un atacante logra robar un access token podría tener acceso permanente a todos los recursos protegidos alojados en el resource Server. En el caso de que los periodos de validez sean demasiado cortos en el tiempo con el objetivo de aumentar la seguridad, el proceso de autorización se tendría que repetir constantemente en el tiempo haciendo que el usuario tenga que introducir sus credenciales múltiples veces y por lo tanto haciendo del sistema poco usable. Es por esto, que OAuth propone el uso de los refresh tokens, los cuales permiten solicitar de nuevo un access token al cliente al authorization server sin necesidad de volver a solicitar el authorization grant al resource owner. En la ilustración 14 se observa este proceso:

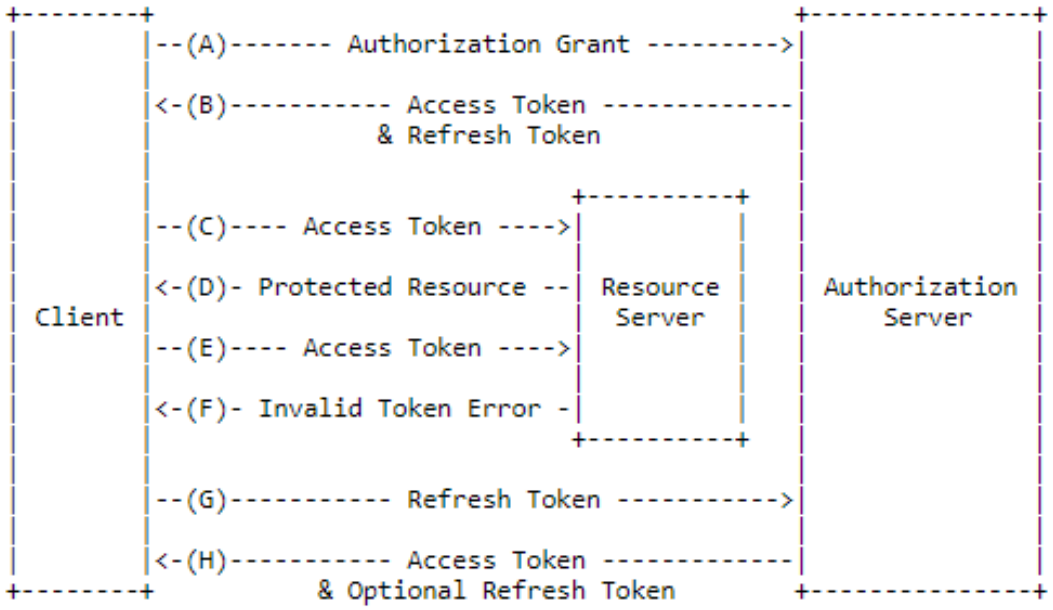

Ilustración 14: Flujo refresh token OAuth 2.0
(Fuente: https://tools.ietf.org/html/rfc6749)

Observe que en el paso B, el authorization server proporciona al cliente 2 tipos de tokens, el access token y el refresh token. Note que en el paso F, el resource server devuelve que el access token proporcionado por el cliente es inválido. En esta casuística, el cliente tendría que volver a iniciar el flujo mostrado en la ilustración 13, sin embargo, este proceso es innecesario puesto que en esta ocasión el cliente posee un refresh token que le permite volver a obtener un acces token válido directamente del authorization server (pasos G y H).

Es interesante detenerse en los dos flujos mediante los cuales un cliente puede obtener el access token y el refresh token por parte del authorization server (Pasos A y B de los flujos anteriores): authorization code grant y implicit grant.

En el caso del Authorization Code Grant esta optimizado para el uso con clientes confidenciales (ver ilustración 15):

(A) El cliente inicia el flujo redirigiendo al user-agent del resource owner al authorization server. El cliente incluye su identificador, el entorno de la petición, su estado y la URI de redirección a la cual se redirige al user-agent cuando el authorization server ha validado la petición.

(B) El authorization server autentica al Resource owner por medio del user-agent permitiendo así establecer cuando una petición de un cliente es válida o no.

(C) Si el resource owner valida una petición realizada por un cliente, el authorization server redirige al user-agent al cliente usando la URI proporcionada en el paso A y dotándole de un authorization grant.

(D) El cliente ya en posesión del authorization grant solicita al authorization server el access token y el refresh token.

(E) El authorization server autentica al cliente y valida el authorization code y la URI de redirección proporcionada. En caso de ser una petición válida, devuelve al cliente el access token y el refresh token.

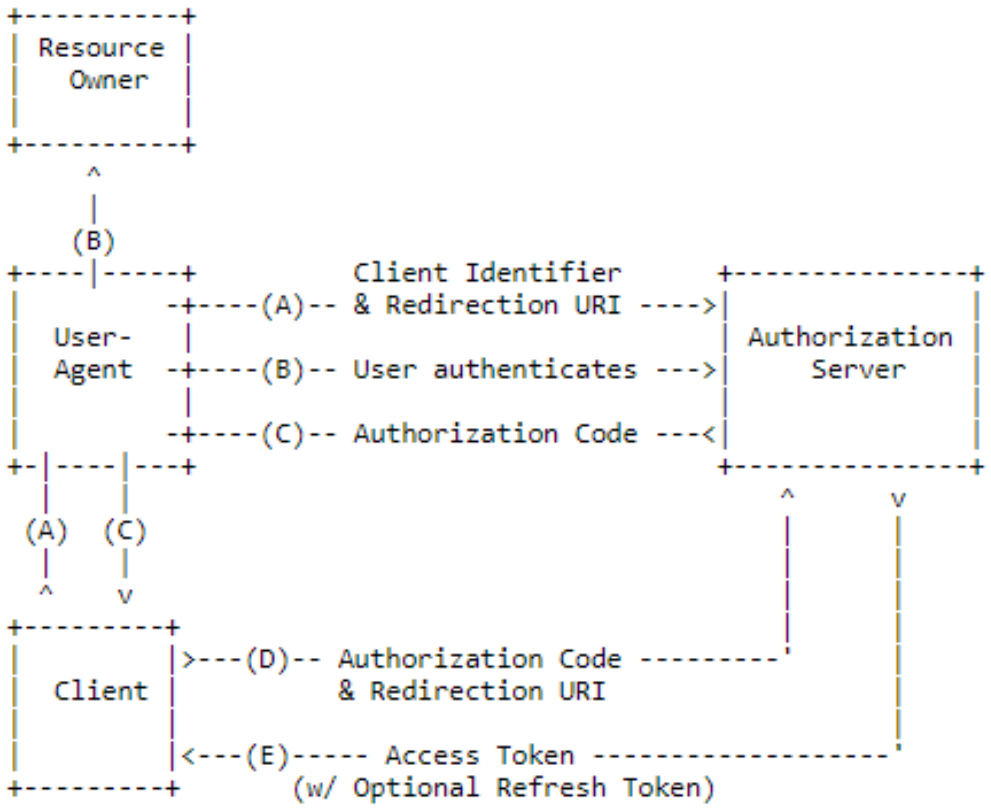

Ilustración 15: Flujo Authorization Code OAuth 2.0
(Fuente: https://tools.ietf.org/html/rfc6749)

En el caso del Implicit Flow esta optimizado para el uso con clientes públicos (ver ilustración 16):

(A) El cliente inicia el flujo redirigiendo al user-agent del resource owner al authorization server. El cliente incluye su identificador, el entorno de la petición, su estado y la URI de redirección a la cual se redirige al user-agent cuando el authorization server ha validado la petición.
(B) El authorization server autentica al Resource owner por medio del user-agent permitiendo así establecer cuando una petición de un cliente es válida o no.
(C) Si el resource owner valida una petición realizada por un cliente, el authorization server redirige al user-agent al cliente usando la URI proporcionada en el paso A y dotándole de un authorization grant
(D) El user agent sigue las instrucciones proporcionadas y envía una petición al web-hosted client.
(E) El web hosted client devuelve una web, normalmente un HTML con el uso de algún script (e.g., javascript) para poder acceder a la URI y extraer el access token entre otros.
(F) El user agent ejecuta el script alojado en el web hosted client y extrae el Access token.
(G) El user agent entonces en posesión del access token se lo proporciona al cliente.

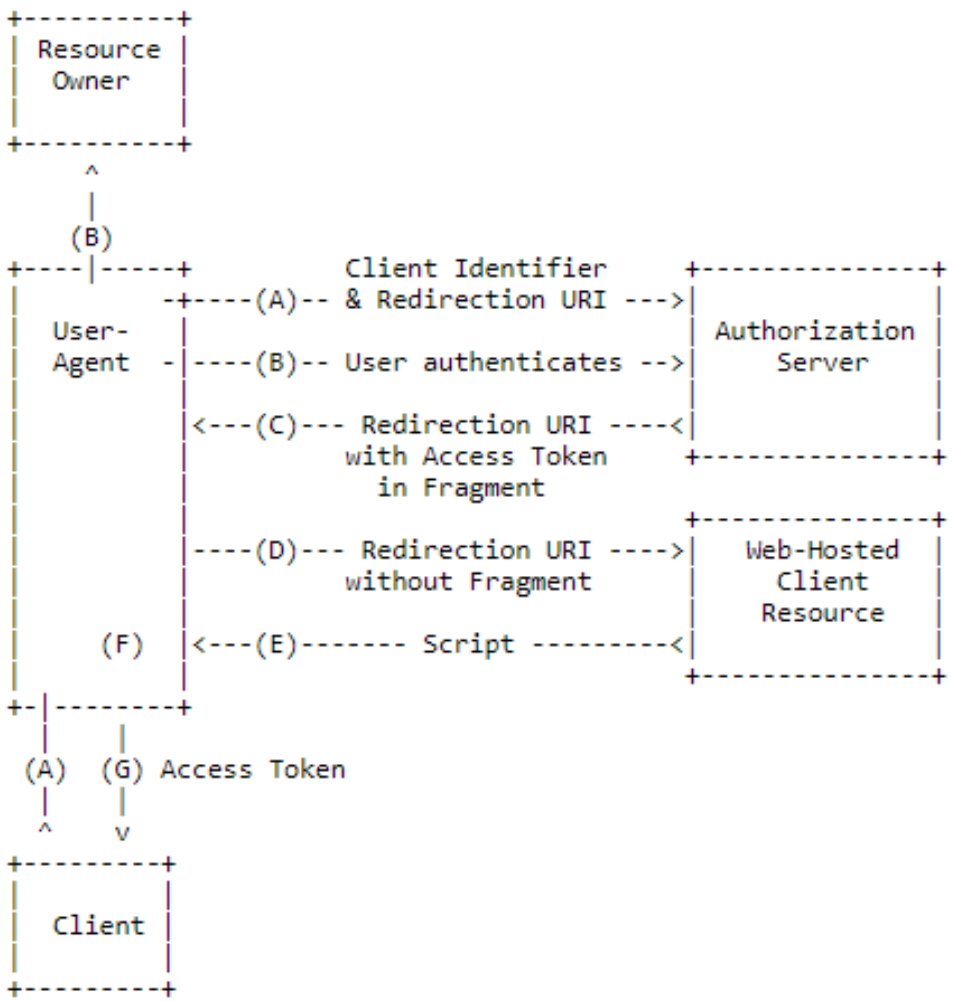

Ilustración 16: Flujo Implicit OAuth 2.0
(Fuente: https://tools.ietf.org/html/rfc6749)

3.4. OpenIdConnect

OpenId [18] se creó en el año 2005 por una comunidad open source. Este modelo se define como un framework abierto, descentralizado y gratis para la gestión de identidades. Hace uso de los protocolos de Internet típicos como HTML y SSL. Está orientado a solventar problemas de autenticación en escenarios web. Actualmente, su versión más reciente, OpenId 2.0 se encuentra obsoleta ya que el protocolo ha evolucionado hacia OpenId Connect.

OpenId Connect [17] es un framework que permite tanto procesos de autenticación como autorización de forma federada. Es una capa por encima de OAuth 2.0 (autorización y auditoria) que integra conceptos de OpenID para dotarle de procesos de autenticación e identificación no soportados anteriormente como se ha podido ver en la sección 3.3. Se creó en el año 2014 por la OpenId Foundation.

Es un protocolo ligero y emergente que permite que todo tipo de clientes como Javascript, aplicaciones web y aplicaciones móviles puedan tanto verificar las identidades de los usuarios, sí como recuperar información de los mismos. El intercambio de mensajes se realiza gracias al uso de JSON. Estas cualidades, hacen que hoy en día sea la solución más amigable para desarrolladores, permitiendo además ofrecer mecanismos de firma y encriptación superiores a los de otras soluciones.

3.4.1. Conceptos fundamentales

OpenId Connect define 3 conceptos fundamentales, además de los vistos en el protocolo de OAuth 2.0:

- **Relaying party (RP):** Es el proveedor de servicios. En el protocolo de OAuth 2.0 también se refieren a este concepto como cliente.
- **OpenID Provider (OP):** Es el IdP. En el protocolo de OAuth 2.0 también se refieren a él como Autorization server.
- **End User (EU):** Es el usuario final, es decir, la persona que desea acceder a los recursos protegidos
- **ID Token:** Es el token que contiene las declaraciones sobre la autenticación del EU. Esta codificado en JWT (JSON Web Token) [16]
- **Claims:** son los parámetros del ID Token que sirven para incluir la información necesaria en cuanto a la identidad del EU. Entre ellos cabe destacar el *iss* que es el identificador de la entidad emisora del token, *auth_time* que es el momento en el que el EU se autentico y el *iat* que es el momento en el que se emitió el token.

OpenId Connect define 3 tipos de arquitecturas:

- **Core:** Define la funcionalidad básica para resolver los problemas de IAAA. Hace uso de 3 Endpoints para que la RP y el OP se puedan comunicar. El Authorization server EndPoint, el Token EndPoint y el UserInfo Endpoint.

- **Dynamic:** Se añade al Core para incluir los servicios de Discovery Dynimaic Client Registration. Esto es posible gracias al uso de dos nuevos Endpoints: Discovery Endpoint y Client Registration Endpoint.
- **Completa:** Suma los servicios de Session Management para monitorizar el estados de inicio de sesión del EU y poder revocar los tokens en caso de que sea necesario y el Form Post Response Mode para codificar los parámetros de respuesta de autorización como valores de formularios HTML. Esto es posible gracias al uso de los Endpoints específicos del inicio y cierre de sesión.

OpenID Connect define 6 tipos de solicitudes o respuestas que se realizan entre la RP y el OP:

- **Solicitud de autorización:** Es realizada por la RP al Authorization Server Endpoint del OP para solicitar el Authorization Code, Access Token y/o el ID Token dependiendo del flujo de información utilizado. Se realiza mediante HTTP GET/POST.
- **Respuesta de autorización:** Se envía desde el Authorization Server Endpoint del OP a la URI de redirección indicada por la RP en la solicitud de autorización. Incluye el authorization code y los tokens solicitados, así como el estado y opcionalmente la caducidad de los tokens y el tipo de token.
- **Solicitud de Tokens:** Es realizada por la RP al Token Endpoint del OP para solicitar los tokens en caso de que se esté realizando un flujo de Authorization Code o hibrido. Se realiza por medio de HTPP POST.
- **Respuesta de Tokens:** El Token Endpoint realiza esta respuesta a la RP devolviendo los Tokens solicitados en la solicitud de tokens, es decir, el Access token y/o el ID Token.
- **Solicitud de información del EU:** La RP realiza esta solicitud al UserInfo EndPint del OP para pedir información relativa al usuario autenticado. Se codifica mediante una solicitud HTTP GET/POST incluyendo el Acess Token referente al EU sobre el que se quiere recoger información
- **Respuesta de información del usuario:** El UserInfo EndPoint devuelve en formato JSON la información pedida en la solicitud de información del EU a la RP.

3.4.2. Flujo de Información

El flujo de información básico de OpenId Connect es muy similar al proporcionado por OAuth 2.0. OpenID Connect contempla tres casuísticas a la hora de realizar los procesos de autorización o autenticación: OpenId Connect Authorization Code flow, OpenId Connect Implicit flow y el hybrid flow.

En primer lugar el OpenId Connect Authorization Code flow es el análogo al Authorization Code flow de OAuth 2.0. Esto se puede observar en la ilustración 17:

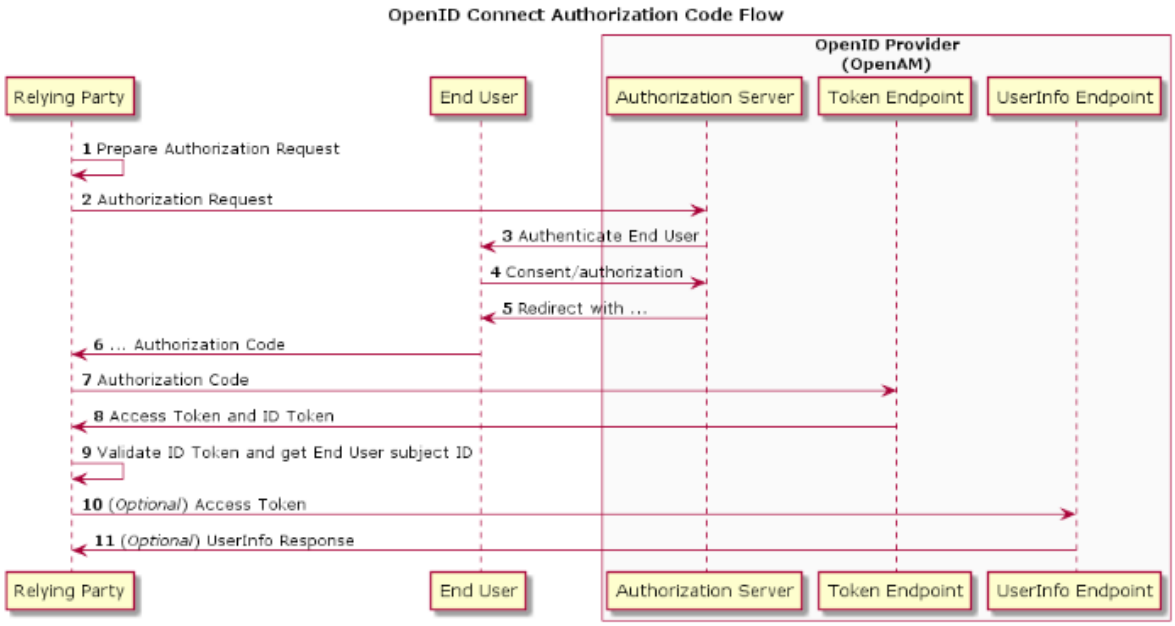

Ilustración 17: Flujo Authorization code OpenID Connect
(Fuente: https://backstage.forgerock.com/docs/am/5/oidc1-guide/)

1. La RP prepara una petición de autorización
2. La RP envía la petición construida en el paso 1 al authorization server
3. El authorization server se comunica con el usuario final para poder autenticarle
4. El usuario proporciona las credenciales al authorization server
5. El Authorization server valida dichas credenciales y en caso satisfactorio redirige al usuario final a la RP proporcionando el authorization code.
6. La RP recibe el Authorization code
7. La RP se comunica con el Token Endpoint para solicitar el acces token y el ID token.
8. El Token EndPoint valida el authorization code y en caso satisfactorio proporciona a la RP el Access token y el ID token.
9. La RP valida los tokens recibidos garantizando que el usuario ya está autorizado.
10. En caso necesario, la RP puede solicitar información adicional sobre el usuario proporcionando su Access token.
11. En caso necesario, el UserInfo Endpoint devuelve la información solicitada en 10.

El OpenId Connect Implicit flow se utiliza para simplificar el proceso de autorización o autenticación cuando el usuario ya ha realizado dicho flujo en alguna ocasión cercana en el tiempo. En la ilustración 18 se puede observar dicho flujo. Note que la única diferencia viene en que la RP en el paso 6 recibe directamente el ID token y el Access Token, pues el authorization server puede determinar por medio de algún lenguaje de scripting (e.g, javascript) que dicho usuario ya estaba autenticado o autorizado y por consiguiente puede recuperar esa información (normalmente de una cookie) y generar un nuevo access token o ID token directamente. Esto es posible ya que el authorization server puede tener acceso directamente al user-agent del usuario. Esto hecho puede

suponer una brecha de seguridad, pues un atacante puede obtener el ID token y el access Token secuestrando una sesión activa del usuario con el authorization Server.

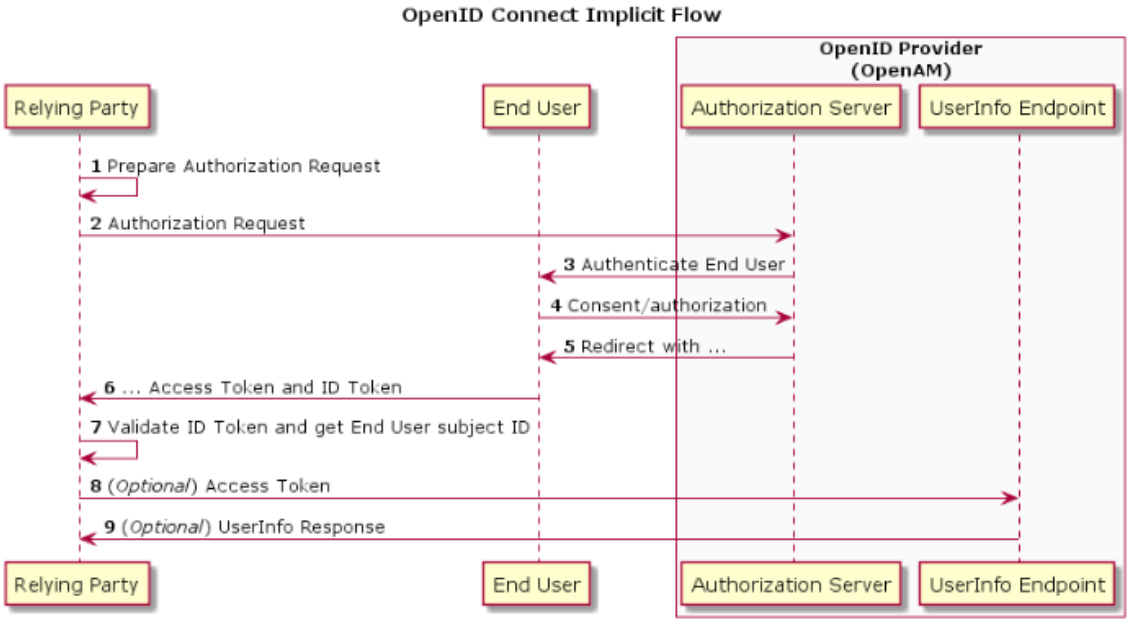

Ilustración 18: Flujo implicit OpenID Connect
(Fuente: https://backstage.forgerock.com/docs/am/5/oidc1-guide/)

El OpenId Connect hybrid flow es una combinación de los dos flujos anteriores. Dependiendo del tipo de conexión que se quiera utilizar, el ID Token y el Access token es proporcionado por el authorization server o por el token endpoint.

En la tabla 1 se muestran las peculiaridades de cada uno de estos flujos:

	Authorization Code	**Implicit**	**Hybrid**
Authorization Endpoint devuelve tokens	No	Si	No
Token EndPoint devuelve tokens	Si	No	No
Los tokens pasan por el user-agent	No	Si	Si
El cliente se puede autenticar	Si	No	Si
Puede usar refresh tokens	Si	No	Si
La comunicación se realiza en una sola interacción	No	Si	No

Tabla 1: Comparativa flujos OpenID Connect

3.5. Mobile Connect

Mobile connect [23] está construido sobre OpenID Connect con la particularidad de solventar problemas de autenticación y autorización haciendo uso del dispositivo móvil, más concretamente de la SIM de dichos dispositivos. Este estándar ha sido impulsado por la GSMA (Global System for Mobile Communications Association).

La idea detrás de Mobile connect es que un usuario pueda utilizar el número de teléfono móvil para poder identificar, validar y autenticar a los usuarios sin que estos

tenga que utilizar más credenciales. Para que un usuario pueda llegar a utilizarlo, en primer lugar ha de solicitar dicho servicio a su operadora telefónica, la cual le proporciona una SIM adaptada para poder soportar este estándar. Posteriormente, los proveedores de servicios pueden ofrecer a los usuarios dados de alta, este mecanismo para identificarlos en su servicio y que puedan así interactuar satisfactoriamente con él.

3.5.1. Conceptos Fundamentales

Mobile Connect define los siguientes conceptos fundamentales además de los expuestos en OpenId Connect y OAuth 2.0:

- **Level of Assurance (LoA):** Grado de confianza que se tiene durante un proceso de autenticación. Es definido por el SP y se incluye en la solicitud de acceso para que el IdP pueda seleccionar correctamente el autenticador a utilizar. Un LoA bajo indica muy poco riesgo en aceptar la solicitud, mientras que un LoA alto indica que se corre mucho riesgo en aceptar la solicutd y por tanto se tendrá que aumentar la seguridad en el proceso incluyendo autenticadores más sofisticados y no solo solicitando al usuario su contraseña

3.5.2. Flujo de Información

En la ilustración 19 se muestra el flujo de información que se produce en Mobile Connect:

1. El usuario por medio de un dispositivo envía una petición de acceso al service provider
2. El service provider descubre el operador que utiliza dicho usuario
3. El service provider crea una solicitud de autenticación para enviársela al operador móvil.
4. El operador móvil valida dicha petición y procede a autenticar a dicho usuario por medio de su dispositivo móvil.

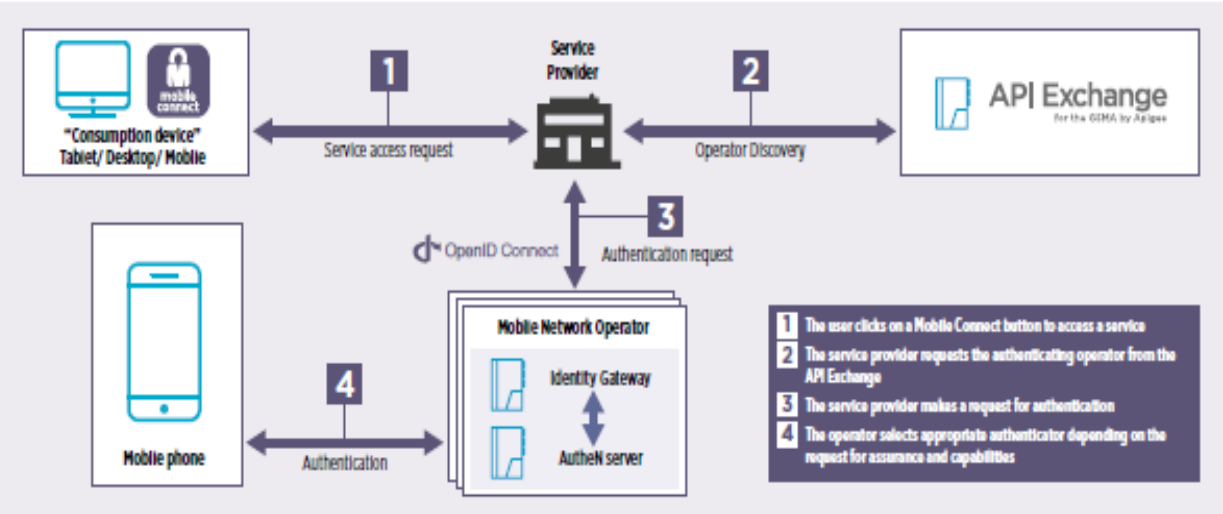

Ilustración 19: Flujo Mobile Connect
(Fuente: https://developer.mobileconnect.io/)

3.6. Conclusiones

Todos los estándares federados presentados están en uso en la actualidad y sirven para solventar el problema de la gestión de identidades de forma correcta. Sin embargo, cada uno de ellos tiene sus peculiaridades. En la tabla 2 se muestra una comparativa entre todos los estándares visto a lo largo de este capítulo.

	SAML	**OAuth2.0**	**OpenID Connect**	**Mobile Connect**
Fecha de Creación	2003	2006	2014	2015
Autenticación	Si	No	Si	SI
Autorización	Si	SI	SI	Si
Protocolos	SAM, XML, HTTP, SOAP	HTTP	HTTP, TLS	HTTP
Formato token	XML	JWT	JWT	JWT
Responsable del consentimiento de usuario	No	Si	Si	Si

Tabla 2: Comparación estándares de sistemas de gestión de identidades federados

Capítulo 4: Implementación de un sistema de gestión de identidades federado

En este capítulo se desarrolla el proceso que se ha llevado a cabo para completar la implementación de un sistema de gestión de identidades federado. A lo largo del mismo, se justifica la elección de las tecnologías y frameworks utilizados para el correcto funcionamiento del sistema. Se entra en detalle en el proceso para levantar el entorno de laboratorio necesario para implementar el sistema, así como en el proceso de crear el IdP y la RP.

4.1. Entorno del Laboratorio

El entorno del laboratorio consta principalmente de 3 componentes físicos. En primer lugar, un primer ordenador que hace la función de IdP. En segundo lugar, un segundo ordenador que hace la función de RP. Ambos ordenadores poseen el sistema operativo Ubuntu en su versión 16.04. Finalmente, un switch para conectar ambos ordenadores en una red local para poder realizar las comunicaciones de forma correcta y emulando así el comportamiento de este tipo de sistemas de la forma más fidedigna posible.

Hay que hacer especial inciso, en el proceso de creación de la red local. Para lograr este objetivo se ha de conectar ambos ordenadores mediante cables Ethernet al switch. Una vez realizado este proceso, se procede a configurar la interfaz de red con el objetivo de establecer las IPs de cada ordenador, máscara de red, puerta de enlace y DNS. Además, se va ha configurado el nombre de cada ordenador con el objetivo de poder identificarlos sin necesidad de recordar su IP. En la ilustración 20 se puede observar la arquitectura del laboratorio, así como las tecnologías necesarias para el correcto funcionamiento en cada uno de los agentes. Note que el ordenador que hace la función de IdP (192.168.1.200) se le ha asignado el nombre de iam.pruebas.com y será referenciado de aquí en adelante como tal, mientras que al segundo ordenador que hace la función de RP (192.168.1.220) se le ha asignado el nombre de server.pruebas.com y será referenciado de aquí en adelante como tal.

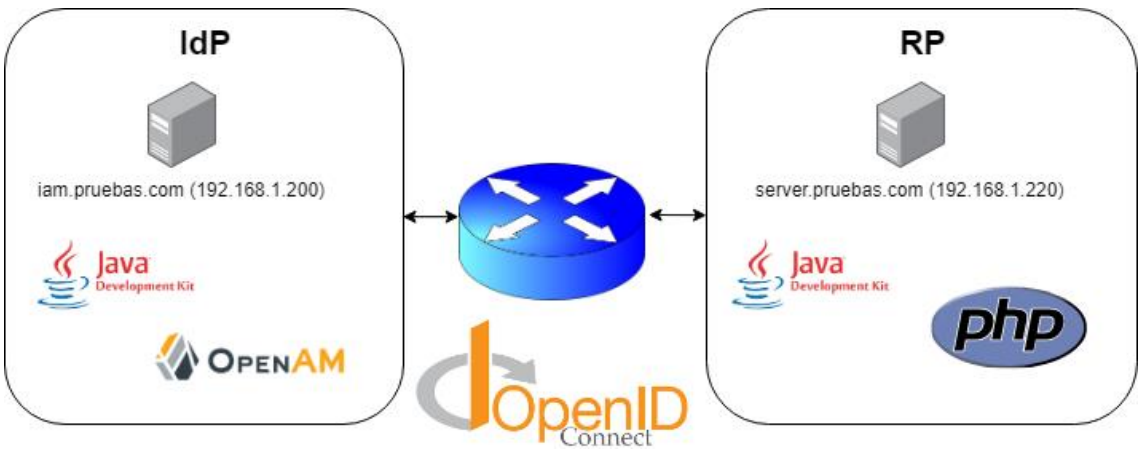

Ilustración 20: Arquitectura del laboratorio

En las tablas 3 y 4 se exponen los pasos llevados a cabo en cada ordenador para montar la red local.

Descripción	Comando ejecutado
Abrir fichero de interfaces de red	sudo nano /etc/network/interfaces
Añadir la configuración de red	auto etch0 iface eth0 inet static address 192.168.1.200 netmask 255.255.255.0 broadcast 192.168.1.255 gatewat 192.168.1.255 dns-nameservers 8.8.8.8 8.8.4.4
Abrir fichero de IP-Hosts	sudo nano /etc/hosts
Añadir la configuración de nombres	127.0.0.1 iam.pruebas.com 192.168.1.220 server.pruebas.com
Reiniciar el ordenador	sudo reboot

Tabla 3: iam.pruebas.com comandos de configuración para red local

Descripción	Comando ejecutado
Abrir fichero de interfaces de red	sudo nano /etc/network/interfaces
Añadir la configuración de red	auto etch0 iface eth0 inet static address 192.168.1.220 netmask 255.255.255.0 broadcast 192.168.1.255 gatewat 192.168.1.255 dns-nameservers 8.8.8.8 8.8.4.4
Abrir fichero de IP-Hosts	sudo nano /etc/hosts
Añadir la configuración de nombres	127.0.0.1 server.pruebas.com 192.168.1.200 iam.pruebas.com
Reiniciar el ordenador	sudo reboot

Tabla 4: server.pruebas.com comandos de configuración para red local

4.2. Implementación del IdP

La implementación del IdP se ha llevado a cabo en la máquina cuyo nombre es iam.pruebas.com. Para poder realizar esta implementación en primer lugar, se ha tenido que realizar un análisis en profundidad de las soluciones propuestas que permitan la implementación de un esquema de gestión de identidades federado. Más concretamente, se ha optado por realizar una implementación del protocolo OpenId Connnect puesto que es una de las soluciones con más repercusión en la actualidad. Una vez escogida la solución, se ha de definir como se ha llevado a cabo la instalación de dicha solución sobre iam.pruebas.com. Finalmente, se ha de configurar dicha implementación para completar un correcto funcionamiento del sistema general de

gestión de identidades, así como garantizar las comunicaciones con la RP implementada en el capítulo 4.3.

4.2.1. Análisis de las soluciones propuestas para esquemas de gestión de identidades federados

Hoy en día existen múltiples implementaciones certificadas para implementar un sistema de gestión de identidades federado y que permitan la implementación del protocolo OpenId Connect. Todas estas soluciones tienen sus particularidades en cuanto al tipo de licencia, lenguaje de programación y protocolos soportados. A continuación se va a realizar un pequeño análisis de muchas de ellas con el objetivo de justificar la elección de una de ellas para la implementación del presente TFM.

4.2.1.1. OpenAm

La primera solución a tener en cuenta es OpenAm [10]. Esta solución nace de OpenSSO [11] que fue desarrollado por Sun Mycrosystems y posteriormente adquirido por Oracle en el año 2009 [14]. Posteriormente, Oracle decidió no continuar con el desarrollo de OpenSSO. Es entonces cuando Forgerock decidió continuar con el proyecto Open Source renombrándolo a OpenAM, consiguiendo unificar todos los componentes de OpenSSO para autenticación, autorización y auditoria en una única solución final. Esta solución esta implementada en los lenguajes de programación Java y C.

OpenAM posee actualmente dos versiones. Una versión gratuita y Open Source y una versión de pago que añade soporte extra para poder pasar a producción las implementaciones desarrolladas. Ambas versiones proporcionan una solución viable para realizar servicios de gestión de identidades, así como lograr el SSO en todo tipo de aplicaciones y servicios web. Cabe destacar, que OpenAM soporta la gran mayoría de protocolos para lograr la gestión de identidades, entre los que hay que hacer especial mención: SAML 1.1, SAML 2.0, OAuth2, OpenID Connect y Kerberos.

Para el correcto funcionamiento de OpenAm se necesita que el ordenador posea un sistema operativo como Red Hat, CentOS, SuSE, Ubuntu, Solaris o Windows Server. Además, se debe tener instalado el JDK (Java Development Kit) [26] y un servidor de aplicaciones web.

La instalación es muy sencilla puesto que se parte de una única aplicación en formato WAR (Web application Resource). Una vez instalado, la solución es escalable, flexible y de fácil personalización puesto que consta de múltiples módulos que se pueden ir añadiendo o quitando al gusto del usuario. Además, cuenta con una comunidad de desarrolladores bastante amplia lo que permite encontrar mucha documentación y soluciones a los problemas de forma sencilla y rápida.

4.2.1.2. Keycloack

Keycloack [25] es un producto software Open Source desarrollado por JBoss que es una subdivisión de RedHat en el año 2014. Su última versión data de abril de 2019 y está implementada en el lenguaje de programación Java. Keycloack se define a sí mismo como una solución sencilla para securizar aplicaciones y servicios sin apenas líneas de código proporcionando una solución a la gestión de identidades. Además, proponen una solución SSO. Este producto soporta la mayoría de protocolos para la gestión de identidades, entre los que cabe destacar kerberos, LDAP, OpenID Connect y SAML.

Por otro lado, los requisitos para instalar keycloack son una base de datos como Postgres, MySQL u Oracle, el JDK y puede ser instalado en cualquier sistema operativo que pueda ejecutar Java. Además, se necesita por lo menos 512 MB de memoria RAM y un 1GB de memoria de almacenamiento.

La instalación es sencilla puesto que los binarios para implementar el servidor donde se instala keycloack vienen incluidos en la propia distribución. Cabe destacar que esta solución es escalable, en cuanto a que permite implementar una arquitectura en cluster de forma sencilla. La comunidad de desarrolladores utiliza Jira para compartir las soluciones y problemas que se van encontrando, lo cual hace muy sencillo obtener retroalimentación del resto de la comunidad.

4.2.1.3. Gluu server

Gluu server [22] es un conjunto de productos desarrollados por la empresa Gluu bajo el mando de Mike Schwartz que comenzó en el año 2009. Estos productos son: Shiboleth, Asimba SAML Proxy, oxAuth, Gluu OpenDJ LDAP y oxTrust. Al igual que openAM, su primera versión está basada en OpenSSO, sin embargo en la actualidad este producto se ha ido reinventando hasta no poseer ninguno de los elementos de OpenSSO.

Las soluciones de Gluu server son totalmente gratuitas, sin embargo, ofrecen distintas opciones de pago para garantizar que sus usuarios obtienen un soporte y mantenimiento correcto. Además, ofrecen al igual que OpenAM y Keycloack soporte para la mayoría de protocolos de gestión de identidades entre los que cabe destacar SAML, OAuth2 y OpenId connect.

Los requisitos de Gluu server son un sistema operativo Ubuntu, CentOs o Rhel. Además, se necesita especialmente una gran capacidad de cómputo (2 CPUs) y gran memoria RAM (4 GB).

La instalación de Gluu server es bastante sencilla, puesto que al tratarse de un servicio se puede instalar haciendo uso de los gestores de paquetes nativos de los sistemas operativos comentados anteriormente, como por ejemplo apt-get en el caso de Linux. En cuanto a la comunidad de desarrolladores de Gluu server, hay que destacar que es

bastante amplia y posee un portal propio para que los desarrolladores puedan comunicarse entre sí.

4.2.1.4. Conclusiones

Tras analizar en las secciones anteriores las principales soluciones para solventar la gestión de identidades, en el presente TFM se ha optado por utilizar la solución de OpenAM. En la tabla 5 se puede observar una comparativa de las 3 soluciones aquí expuestas.

Esta decisión se fundamenta en que OpenAm cumple con los requisitos del presente trabajo, ya que soporta el protocolo OpenID Connect. Además, su instalación es relativamente sencilla, requiere poca capacidad de cómputo y es muy flexible y personalizable. Finalmente, también se ha tenido en cuenta la gran calidad de su documentación y la gran comunidad de desarrolladores detrás de dicho producto.

	OpenAM	**Keycloack**	**Gluu Server**
Empresa	ForgeRock	Red Hat	Gluu
Lenguaje de desarrollo	Java y C	Java	Python y Java
Primera versión	2010	2014	2010
Última versión	Abr 2019	Abr 2019	Abr 2019
Documentación y Comunidad	Muy Buena	Muy Buena	Buena
Protocolos	SAML, WS-Federation, WS-Trust, OAtuh2, OpenID Connect, XACML, Kerberos	kerberos, LDAP, OAuth2 OpenID Connect, SAML	SAML, OpenID, OpenID Connect, UMA, Radius, LDAP

Tabla 5: Comparación de las soluciones propuestas para esquemas de gestión de identidades federados

4.2.2. Instalación y Configuración de OpenAM

En esta sección se explica cual han sido los pasos llevados a cabo para poder instalar OpenAM sobre la máquina cuyo nombre es iam.pruebas.com. Forgerock dispone actualmente de dos versiones de OpenAm, la versión 13 y la versión 13.5. Puesto que la versión 13 sigue teniendo soporte por parte de ForgeRock y la comunidad de desarrolladores es más amplia se puede considerar que es la versión que más se adecua a los requisitos del presente trabajo y por lo tanto la elegida para instalar.

En primer lugar, Forgerock solicita un registro en su página web para poder permitir la descarga de OpenAM. Una vez completado el registro, se procede a descargar la versión WAR de OpenAM. Esta descarga se realiza por medio de un navegador web en un ordenador externo y posteriormente es traspasada por medio de un dispositivo pendrive a la máquina iam.pruebas.com, puesto que esta máquina no tiene salida a internet.

Una vez se dispone de la aplicación de OpenAM en la máquina, para su correcto funcionamiento es necesario instalar un servidor web y el JDK (ver capítulo 4.2.1.1). La elección del servidor web ha sido Tomcat 7 debido a su facilidad de instalación y facilidad de uso, siendo este uno de los servidores web más utilizados actualmente. El proceso de instalación se puede ver resumido en la tabla 6.

Descripción	Comando ejecutado
Instalar Tomcat 7	sudo apt-get install tomcat7
Instalar JDK	sudo apt-get install default.jdk
Cambiar permisos	sudo chown –R tomcat7 /usr/share/tomcat7 sudo chmod –R 775 /var/log/tomcat7
Mover OpenAM a Tomcat	mv /home/openam/openam.war /var/lib/tomcat7/webapps/

Tabla 6: iam.pruebas.com comandos de instalación servidor web

Finalizada la instalación de OpenAM se procede a levantar el servidor web de tomcat. En este instante, gracias al entorno de laboratorio montado, es posible acceder desde la máquina server.pruebas.com a la dirección http://iam.pruebas.com:8080/openam y proceder a la configuración de OpenAM.

En este instante OpenAm, solicita si se desea realizar una configuración estándar o una configuración personalizada. Para adecuar OpenAM a los requisitos específicos se selecciona esta segunda opción y se procede a introducir los valores mostrados en la tabla 7.

Parámetro	Valor
Contraseña usuario amAdmin	*****
URL del servidor	Iam.pruebas.com:8080/openam
Dominio de Cookies	Iam.pruebas.com
Idioma	Español
Directorio de Configuración	/usr/share/tomcat7/openam
Almacén de datos de configuración	OpenAM
Puerto	50389
Puerto de administración	4444
Puerto de JMX	1689
Clave de cifrado	Por defecto
Almacén de usuarios	Almacén de OpenAM
Contraseña para el usuario UrlAccessAgent	*****

Tabla 7: Parámetros de Configuración OpenAM

Una vez finalizada la configuración principal de OpenAM, se redirige a la pantalla de inicio de OpenAM en la cual se ha de introducir las credenciales del usuario de

administración amAdmin. El primer paso una vez se ha obtenido acceso, es crear un nuevo dominio de cookie para la máquina server.pruebas.com desde el panel configuración/system/platform.

En la pantalla de inicio se puede observar como OpenAM ha generado automáticamente un elemento del tipo Realm denominado "Top Level Realm" (ver ilustración 21). Este elemento contiene toda la configuración sobre las políticas de seguridad de autenticación y autorización, módulos, almacén de datos a utilizar, agentes y sujetos creados en el sistema. Además, es posible crear subdivisiones de este elemento para considerar diferentes configuraciones dependiendo de la aplicación sobre la que se quiera realizar un proceso de securización.

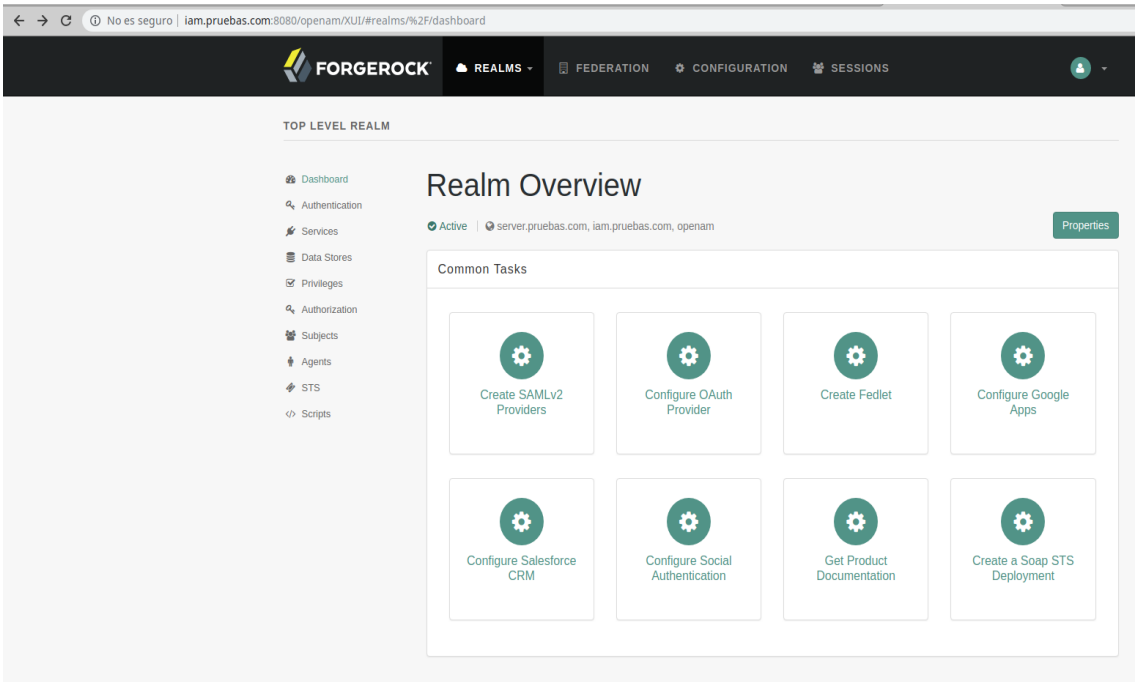

Ilustración 21: OpenAm realm pantalla de configuración

OpenAM dispone de múltiples agentes, que no son más que librerías o componentes que se pueden instalar en los contenedores web para protegerlos. Hacen de controladores de acceso, permitiendo así realizar los procesos de IAAA. Entre los agentes cabe destacar:

- **Agentes web:** El objetivo es proteger sitios web alojados en servidores Apache o Microsoft IIS
- **Agentes J2EE:** El objetivo es proteger aplicaciones web escritas en Java y alojadas en servidores Jboss, Jetty, WebLogic, GlassFish, IBM y Apache Tomcat.
- **Agentes 2.2:** Son un hibrido entre los agentes web y los agentes J2EE
- **Clientes OAuth 2.0/OpenID Connect**: El objetivo es que OpenAM pueda ser utilizado como proveedor de estos estándares haciendo uso de una API Rest para lograr la comunicación con la RP.
- **Agentes de autenticación:** Permite que se pueda leer perfiles de usuario mediante OpenAM haciendo uso de pares usuario-contraseña
- **Agentes Security Token Service de SOAP**: Su objetivo es securizar las solicitudes del servicio de token de seguridad.

Puesto que el requisito fundamental del presente trabajo es realizar una implementación de un esquema de gestión de identidades federado, es necesario únicamente centrarse en el cliente OAuth 2.0/OpenID Connect. En primer lugar, es necesario habilitar OpenAM como proveedor OAuth 2.0 y OpenID connect. Para ello, desde el panel de administración es necesario acceder a la ruta "/Top Level Realm/Dashboard/Configure OAuth Provider/Configure OpenID Connect" e introducir los parámetros requeridos entre los que cabe destacar la elección del tiempo de vida los token y códigos (ver ilustración 22). Una vez realizado este proceso, en el panel de administración se navega a "Top Level Realm/Agents/OAuth2.0/OpenID Connect/OIDC Agent" y se procede a crear un cliente OAuth 2.0/ OpenID connect introduciendo los parámetros necesarios para que haya concordancia con la RP desarrollada. Los parámetros introducidos se pueden observar en la tabla 8, entre los que cabe destacar las URIs de redirección, pues sin una correcta configuración de dichas URIs, OpenAM no podrá validar la página a la que redirigirá a los usuarios tras realizar un proceso de IAAA, imposibilitando así el correcto funcionamiento de la aplicación.

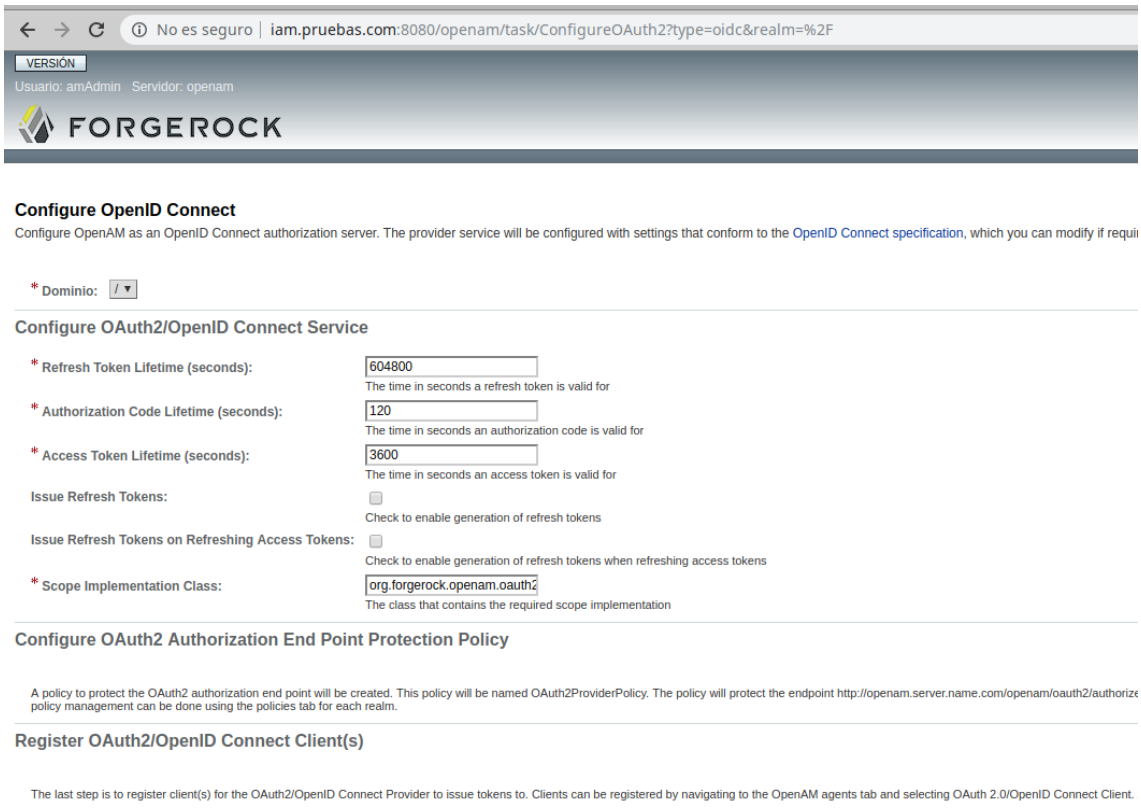

Ilustración 22: OpenAM configuración de OpenID Connect

Parámetro	Valor
Nombre del agente	myRP
Contraseña del agente	*****
Tipo de cliente	Confidencial
Uris de redirección	http://server.pruebas.com:80/rp/
Scope solicitado	Openid, profile, email
Scope por defecto	Openid, profile, email
Tipo de respuesta aceptados	Code, tokem, id_token, token id_token, code id_token, code token id_token

Tabla 8: Parámetros de configuración cliente OAuth 2.0/ OpenID connect

4.3. Implementación de la RP

La implementación de la RP ha sido desarrollada en el lenguaje de programación PHP haciendo uso de la guía expuesta en la documentación oficial de OpenAM [9] así como de los ejemplos expuestos en la misma guía escritos en javascript [19].

El objetivo de esta RP es utilizar la API Rest que proporciona OpenAM para poder comunicarse con los EndPoints correspondientes de OpenID Connect, consiguiendo así completar satisfactoriamente los flujos de IAAA.

La aplicación desarrollada (adjunta en la entrega), se encarga de realizar las solicitudes correspondientes, por medio de la API Rest para poder recibir los tokens correspondientes y validarlos. Esta implementación, permite completar ambos flujos de OpenID Connect, es decir, el Authorization Code y el implicit. Para su elaboración, dentro de la carpeta /rp/flow/ se encuentran las clases desarrolladas para lograr cada uno de estos flujos. Además, en el directorio /rp/token se encuentran las clases desarrolladas para poder almacenar cada uno de los tokens recibidos así como la publickey. Por último, en el directorio /rp/utils se encuentra el archivo de configuración desde el cual hay que configurar las rutas de los Endpoints, así como el cliente y su contraseña para el correcto funcionamiento de la aplicación.

El desarrollo final contiene tres pantallas principales. En la primera pantalla (index.php), se solita al usuario que pulse sobre el flujo que quiere utilizar para completar el proceso de IAAA. Esta pantalla, redirige al usuario a la máquina iam.pruebas.com, donde deberá introducir sus credenciales. Note que, para lograr autenticarse correctamente, el usuario ha de ser almacenado previamente en el almacén de usuarios de OpenAM. Tras comprobar las credenciales, la máquina iam.pruebas.com devuelve los tokens solicitados y redirige al usuario a la segunda pantalla (admin.php) donde se muestra información referente a los tokens devueltos, así como su validación. La tercera pantalla (forbidden.html) únicamente muestra al usuario un acceso indebido, es decir, en caso de que un usuario que no posea tokens e intente acceder a la segunda pantalla (admin.php), este será redirigido a esta pantalla indicándole su falta de permisos y un botón para volver a la primera pantalla (index.php).

Cabe destacar, que un usuario que posea tokens incorrectos o caducados, podrá acceder a la segunda pantalla (admin.php), siendo informado por pantalla de cual son los parámetros que no han podido ser validados. Este no es el funcionamiento normal de una web en producción, sin embargo, para el presente trabajo se ha optado por permitir dicho acceso ya que el propósito es poder entender y comprobar los flujos de información y los mensajes transmitidos entre el IdP y la RP. Además, adaptar la web para no permitir el acceso es una tarea trivial, puesto que la validación de la información si se está realizando y mostrándosela al usuario por pantalla.

Finalmente, en las ilustraciones 23, 24, 25, 26 y 27 se puede observar cada una de las pantallas de la RP desarrollada para poder entender el funcionamiento de la misma. Además, en el video de la defensa del presente trabajo se realiza una demostración en directo del funcionamiento de la misma.

Ilustración 23: Pantalla principal index.php

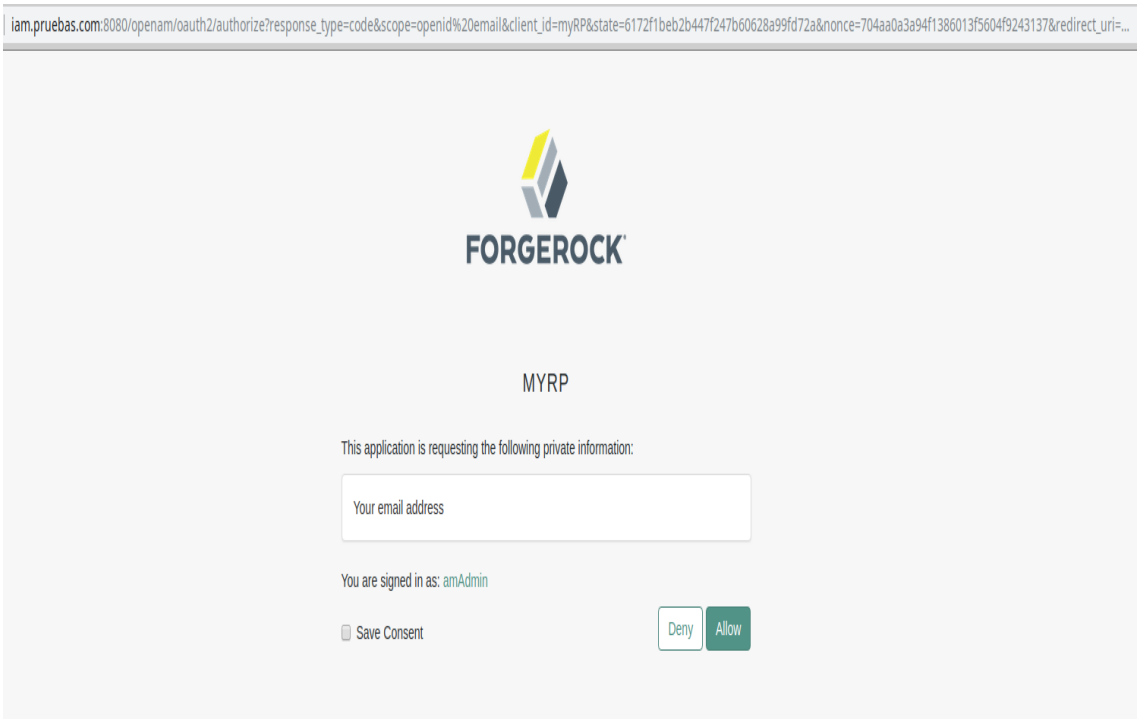

Ilustración 24: Pantalla de login – Redirección de RP a IdP

TFM - UOC - OpenID Connect POC

LogOut

Validación de Tokens

Validación del Access Token

State Code: Correcto

Hash Access Token: Correcto

Validación del ID Token

ISS ID Token: Correcto

AUD ID Token: Correcto

AZP ID Token: Correcto

EXP ID Token: Correcto

IAT ID Token: Correcto

NONCE ID Token: Correcto

Acepted Time ID Token: Correcto

Signature ID Token: Correcto

Datos

Datos del Access Token


```
{
   "info": "Code Flow"
   "access_token": "ee818d89-16db-4469-a126-6aff10cce3c0"
   "state": "6172f1beb2b447f247b60628a99fd72a"
}
```


Ilustración 25: Pantalla de validación admin.php 1/2

Signature ID Token: Correcto

Datos

Datos del Access Token


```
{
   "info": "Code Flow"
   "access_token": "ee818d89-16db-4469-a126-6aff10cce3c0"
   "state": "6172f1beb2b447f247b60628a99fd72a"
}
```


Datos del ID Token


```
{
   "id_token":
"eyAidHlwIjogIkpXVCIsICJhbGciOiAiUlMyNTYiLCAia2lkIjogIlN5bExDNk5qdDFLR1FrdEQ5TXQrMHpjZVFTVT0iIH0.eyAidG9rZW5OY
W1lIjogImlkX3Rva2VuIiwgImF6cCI6ICJteVJQIiwgInN1YiI6ICJhbWFkbWluIiwgImF0X2hhc2giOiAial9Gei1Pbm1DZHFGYWRPWkphSjdr
USIsICJpc3MiOiAiaHR0cDovL2lhbS5wcnVlYmFzLmNvbTo4MDgwL29wZW5hbS9vYXV0aDIiLCAib3JnLmZvcmdlcm9jay5vcGVuaWRjb2
5uZWN0Lm9wcyI6ICI5OTMyZWU0OS00ZDQ2LTRlNzMtOWQ2MC04NDdmMTIyMzA5NjgiLCAiaWF0IjogMTU1OTU3MTcyNiwgImF1
dGhfdGltZSI6IDE1NTk1NzExNzUsICJleHAiOiAxNTU5NTc1MzI2LCAidG9rZW5UeXBlIjogIkpXVFRva2VuIiwgIm5vbmNlIjogIjcwNGFhM
GEzYTk0ZjEzODYwMTNmNTYwNGY5MjQzMTM3IiwgInJlYWxtIjogIi8iLCAiYXVkIjogWyAibXlSUCIgXSwgImNfaGFzaCI6ICItTGh3d
mFGMnZGZ243OVREcVRSaVRnIiB9.WyE4rp8gaMENYvN8d7egzPeT8cC9nLP1dzGdAUQgV9gOkqo-
JemiJaxsWu0L9LaYNty5181XTAgO6yagvo4qT4A7ijEVLpCmT0_eqPIUbilxdu9jQDwjzNu72pBKvR_CkNtr-Q1QZLFZqoCzTTd-9qWjKM5-
Wso2eiqJ7DS3CGU",
   "header": {
      "typ": "JWT",
      "alg": "RS256",
      "kid": "SylLC6Njt1KGQktD9Mt+0zceQSU="
   },
   "payload": {
      "tokenName": "id_token",
      "azp": "myRP",
      "sub": "amadmin",
      "at_hash": "j_Fz-OnmCdqFadOZJaJ7kQ",
      "iss": "http://iam.pruebas.com:8080/openam/oauth2",
      "iat": "1559571726",
      "auth_time": "1559571175",
      "exp": "1559575326",
      "tokenType": "JWTToken",
      "nonce": "704aa0a3a94f1386013f5604f9243137",
      "realm": "/",
      "aud": ["myRP"],
   },
   "sign": "WyE4rp8gaMENYvN8d7egzPeT8cC9nLP1dzGdAUQgV9gOkqo-
JemiJaxsWu0L9LaYNty5181XTAgO6yagvo4qT4A7ijEVLpCmT0_eqPIUbilxdu9jQDwjzNu72pBKvR_CkNtr-Q1QZLFZqoCzTTd-9qWjKM5-
Wso2eiqJ7DS3CGU"
}
```


Ilustración 26: Pantalla de validación admin.php 2/2

Ilustración 27: Pantalla de falta de permisos – forbidden.html

Capítulo 5: Conclusiones

En el presente trabajo se ha expuesto la evolución que han seguido los esquemas de gestión de identidades desde el control de accesos, pasando por los modelos SILO, los modelos centralizados hasta llegar a los esquemas de gestión de identidades federados. Estos últimos, han sido analizados en profundidad explicando los principales agentes que intervienen en cada uno de ellos, su funcionamiento (flujos) y las peculiaridades de cada uno de ellos.

También, se ha podido comprobar de primera mano el funcionamiento de un esquema de gestión de identidades federado gracias a la implementación de un caso práctico. Este caso práctico ha permitido poder entender como se ha de securizar una aplicación web (RP) para que disponga en los flujos de IAAA de un sistema de gestión de identidades federado, así como poder entender como se ha de gestionar las identidades de los usuarios por parte del IdP.

Tras finalizar el estudio, se ha podido comprobar que los sistemas de gestión de identidades federados están en auge y posiblemente la gran mayoría de propuestas software en las que interviene la gestión de identidades acaben optando por incluirlas. Sin embargo, estos sistemas son muy recientes y por consiguiente los análisis de riesgos realizados hasta el momento determinan que todavía existen muchas debilidades y puntos vulnerables que han de ir solucionado en sus nuevas versiones [5,32].

Durante el desarrollo del presente trabajo se ha podido poner en práctica los conocimientos adquiridos durante el máster universitario de seguridad en las TIC (Tecnologías de la Información y Comunicación). Más concretamente, de las asignaturas “Identidad Digital”, “Seguridad en sistemas operativos”, “Seguridad en redes” y “Vulnerabilidades de seguridad”. Además, se ha podido profundizar en todas ellas, llevando a cabo labores de investigación puesto que el ámbito de desarrollo del presente trabajo es muy nuevo y actualmente siguen surgiendo nuevas versiones de las tecnologías aquí expuestas.

Bibliografía

[1] Box, Don, et al. "Simple object access protocol (SOAP) 1.1." (2000).

[2] Brosso, I., La Neve, A., Bressan, G., & Ruggiero, W. V. (2010, February). A continuous authentication system based on user behavior analysis. In 2010 International Conference on Availability, Reliability and Security (pp. 380-385). IEEE.

[3] Chadwick, D. (1994). Understanding X. 500: the directory. Chapman & Hall, Ltd..

[4] Fielding, Roy, et al. Hypertext transfer protocol--HTTP/1.1. No. RFC 2616. 1999.

[5] Ghazizadeh, E., Zamani, M., & Pashang, A. (2012, December). A survey on security issues of federated identity in the cloud computing. In 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings(pp. 532-565). IEEE.

[6] Gollmann, D. (2010). Computer security. Wiley Interdisciplinary Reviews: Computational Statistics, 2(5), 544-554.

[7] Howes, T. A., Smith, M. C., & Good, G. S. (2003). Understanding and deploying LDAP directory services. Addison-Wesley Longman Publishing Co., Inc..

[8] http://saml.xml.org/saml-specifications. Acceso: 03/06/2019

[9] https://backstage.forgerock.com/docs/am/6/oidc1-guide/ . Acceso: 03/06/2019

[10] https://backstage.forgerock.com/docs/openam/13.5/getting-started/. Acceso: 03/06/2019

[11] https://docs.oracle.com/cd/E19575-01/820-3740/adrab/index.html. Acceso: 03/06/2019

[12] https://ebrary.net/24577/computer_science/identity_management_system_requirements. Acceso: 03/06/2019

[13] https://en.wikipedia.org/wiki/Discretionary_access_control . Acceso: 03/06/2019

[14] https://en.wikipedia.org/wiki/OpenAM. Acceso: 03/06/2019

[15] https://en.wikipedia.org/wiki/OSI_model. Acceso: 03/06/2019

[16] https://jwt.io/. Acceso: 03/06/2019

[17] https://openid.net/specs/openid-connect-core-1_0.html. Acceso: 03/06/2019

[18] https://openid.net/what-is-openid/. Acceso: 03/06/2019

[19] https://stash.forgerock.org/projects/COM/repos/openid/browse. Acceso: 03/06/2019

[20] https://tools.ietf.org/html/rfc5849. Acceso: 03/06/2019

[21] https://tools.ietf.org/html/rfc6749. Acceso: 03/06/2019

[22] https://www.gluu.org/. Acceso: 03/06/2019

[23] https://www.gsma.com/latinamerica/wp-content/uploads/2016/06/techdoc-MC-MNO_Implementation_Requirements-1.pdf. Acceso: 03/06/2019

[24] https://www.itu.int/en/about/Pages/default.aspx. Acceso: 03/06/2019

[25] https://www.keycloak.org/. Acceso: 03/06/2019

[26] https://www.oracle.com/technetwork/java/javase/documentation/index.html. Acceso: 03/06/2019

[27] https://www.untruth.org/~josh/security/radius/radius-auth.html. Acceso: 03/06/2019

[28] https://www.w3.org/TR/xml/. Acceso: 03/06/2019

[29] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn,W.Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries,et al., “Manifesto for agile software development,” 2001.

[30] Karp, Alan H., Harry Haury, and Michael H. Davis. "From ABAC to ZBAC: the evolution of access control models." Journal of Information Warfare 9.2 (2010): 38-46.

[31] Laurent, M., & Bouzefrane, S. (2015). Digital identity management. Elsevier.

[32] Maler, Eve, and Drummond Reed. "The venn of identity: Options and issues in federated identity management." IEEE Security & Privacy 6.2 (2008): 16-23.

[33] Marini, E. (2012). El Modelo Cliente/Servidor. Recuperado el, 5.

[34] Miller, S. P., Neuman, B. C., Schiller, J. I., & Saltzer, J. H. (1988). Kerberos authentication and authorization system. In In Project Athena Technical Plan.

[35] O'Gorman, L. (2003). Comparing passwords, tokens, and biometrics for user authentication. Proceedings of the IEEE, 91(12), 2021-2040.

[36] Pato, J., & Center, O. C. (2003). Identity management: Setting context. Hewlett-Packard, Cambridge, MA.

[37] R. Aarts. Liberty Reverse HTTP Binding for SOAP Specification Version 1.0.

[38] Rigney, C., Rubens, A., Simpson, W., & Willens, S. (1996). Remote authentication dial in user service (RADIUS) (No. RFC 2058)

[39] Sandhu, Ravi S. "The typed access matrix model." Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy. IEEE, 1992.

[40] Sandhu, Ravi S., and Pierangela Samarati. "Access control: principle and practice." IEEE communications magazine 32.9 (1994): 40-48.

[41] Vacca, J. R. (2012). Computer and information security handbook. Newnes.

[42] Williamson, G., Yip, D., Sharoni, I., & Spaulding, K. (2009). Identity management: A primer. MC Press, LLC.