Trabajo de grado

Single Sign On

Facultad de Informática UNLP

Alumno: A.C. Luciano Iglesias
Director: Lic. Javier Díaz
Codirectora: C.C. Lía Molinari

Índice

Capítulo 1

Introducción

1.1 Presentación del problema

En la actualidad muchas organizaciones padecen las consecuencias de un legado tecnológico de años de evolución y acumulación de plataformas, redes, y sistemas diversos. Esta heterogeneidad en la infraestructura provoca vulnerabilidades no contempladas en el diseño de los esquemas originales, incrementando proporcionalmente el riesgo y los esfuerzos de administración asociados. La falta de una visión global e integradora de todos los mecanismos de seguridad obliga a las organizaciones a resolver los problemas mediante soluciones puntuales lo cual no aporta en un fortalecimiento de la seguridad de la infraestructura tecnológica en su totalidad. En este escenario operativo, los usuarios deben trabajar con múltiples esquemas de acreditación, complejos, y poco eficientes. La información de los usuarios está dispersa en las distintas plataformas provocando inconsistencia de datos y, además, la necesidad de administrar información operativa en diversos puntos. Las distintas plataformas tienen, en muchos casos, su propio modelo de autenticación.

Consideremos una compañía que tiene varios departamentos. Durante años de operación la compañía ha comprado varios sistemas de computación, algunos de los cuales ni siquiera están preparados para trabajar en red. Cada uno de esos recursos necesita estar disponible para los empleados, de forma segura. Lo que necesitan es un sistema que provea autorización y autenticación. Típicamente los usuarios son ingresados en una base de datos por cada sistema y se les entrega un nombre de usuario y una contraseña para acceder. El resultado final, en estos sistemas heterogéneos, resulta ser que cada usuario necesita tener una contraseña para cada sistema, ingresando a los mismos en forma separada. Esta situación está lejos de ser la ideal. Los usuarios encuentran esto molesto para trabajar. Cada vez que empiezan a trabajar deben ingresar varias contraseñas para entrar a todos los sistemas que necesitan. Todas las contraseñas deben ser recordadas, aún cuando estas son cambiadas periódicamente. Cuando un nuevo usuario ingresa a la compañía, requiere de un largo rato antes de que sean configuradas todas las cuentas que él necesita para acceder a los recursos.

Desde el punto de vista del administrador los sistemas de seguridad heterogéneos son una verdadera complicación. Hay varias bases de datos para manejar. Configurar una cuenta es una tarea engorrosa, pero esto no es nada comparado con la situación de que alguien abandone la organización. Nadie sabe a ciencia cierta a cuantos lugares tiene el individuo acceso. Debe entonces el administrador ir sistema por sistema borrando las cuentas de usuario, teniendo mucho cuidado de no olvidar ninguna pues esto iría en detrimento de la seguridad de la organización. Los usuarios que no recuerden sus contraseñas recargan al administrador con la tarea de cambiarlas. Además, los integrantes de la organización bajan su productividad haciendo tareas como éstas, o aún peor, como no pueden recordar las contraseñas las escriben debajo del escritorio, el teclado o detrás de la computadora. Como el proceso de iniciar las sesiones es una tarea engorrosa, los usuarios evitan tener que hacerlo dejándose a ellos mismos con la sesión iniciada, permitiendo ataques internos.

Por todo esto es de particular interés investigar aspectos relacionados con los sistemas de seguridad actuales, en especial analizar las posibilidades que ofrecen respecto a diferentes aspectos tales como autenticación, autorización, políticas de acreditación, administración de los datos de los usuarios, etc. También, es de interés estudiar sus puntos débiles y fuertes, analizando la solución que ofrecen a algunos de los puntos anteriormente mencionados como característicos de los sistemas tradicionales. A continuación se presenta la motivación de este trabajo así como también los objetivos propuestos para el mismo.

1.2 Motivación y Objetivos

El concepto de Single Sign On (*SSO*) [Open Group 2002] permite complementar los sistemas de seguridad actuales de las organizaciones que desean robustecerlo, incorporando un único punto de acceso a las diversas aplicaciones y/o recursos existentes. El principal objetivo de implementar un esquema de SSO dentro de una organización, es proveer al usuario final con la habilidad de contar con un único punto de acceso a las aplicaciones y/o recursos que necesita. En un esquema SSO, la identidad de los usuarios es verificada sólo una vez y su identidad electrónica es transferida a los diferentes recursos de computación de forma segura y automática. A través de este esquema de acceso único, los componentes tecnológicos con los cuales se implementa, se encargan de autenticar y administrar las entradas y salidas a los elementos de red o aplicaciones involucradas de manera transparente para el usuario. De acuerdo a la forma en que la identidad y las credenciales de usuario son distribuidas y manejadas, las soluciones de SSO pueden distinguirse entre sistemas centralizados y sistemas federados [Beatty 2002].

Los principales objetivos de este trabajo son:

- Estudiar y analizar los distintos modelos de SSO. Investigar algunas de las soluciones ofrecidas actualmente en el mercado.
- Plantear un caso de estudio sobre aplicaciones reales y llevar a cabo su implementación.
- Comparar las diferentes soluciones a partir de los modelos analizados y del caso de estudio planteado.

1.3 Plan de Trabajo

Este trabajo se divide en 6 capítulos donde se irán presentando de manera gradual los distintos temas para poder abordar los objetivos.

El **Capítulo 1** presenta la problemática del tema planteado en este trabajo de grado, su motivación y los objetivos.

El **Capítulo 2** tiene como objetivo definir los conceptos relacionados así como también terminología que se utilizará a lo largo de este trabajo, de manera tal de contextualizar al lector. También se presentan las diferentes soluciones que se estudiarán y que permiten conocer el estado del arte de la temática tratada.

El **Capítulo 3** presenta el concepto de Servicio de Directorio, las ventajas de utilizar un Servicio de Directorio único para varias aplicaciones, cómo se conforma un directorio, y por último, se estudia el protocolo LDAP.

El **Capítulo 4** detalla la propuesta de SSO, estudiando su propósito. Además aquí se presenta el caso de estudio planteado, y las implementaciones realizadas. En particular los casos de estudio muestran la integración del modelo SSO con algunas aplicaciones web que actualmente no utilizan este modelo.

En el **Capítulo 5** se realiza una comparación de las soluciones analizadas, se evalúa como es la política de seguridad de cada una, tomando aspectos puntuales como base de la comparación. Se realiza un cuadro comparativo basado en los estudios previos y en el caso de estudio.

Finalmente, el **Capítulo 6** presenta un resumen de los resultados obtenidos, donde se detallan algunas conclusiones y posibles líneas de investigación futuras.

Al final de la tesis se anexa un glosario de términos técnicos, que aparecen en este texto, y un conjunto de referencias de consulta con las que se desarrolló este trabajo.

Capítulo 2

Análisis del Estado del Arte

2.1 Conceptos básicos

En este capítulo se definen algunos conceptos necesarios para la comprensión del texto. En general, estos conceptos están relacionados con las necesidades de seguridad que existen en las redes de hoy en día, y especialmente de Internet. Algunos de los conceptos básicos relacionados con el traslado de información son:

Privacidad: asegura que los datos no son leídos durante su tránsito.
Autenticación: garantiza que una persona o entidad es quien dice ser.
Integridad: asegura que los datos no son modificados durante su tránsito.
Autorización: determina si una persona o entidad tiene el derecho o no a acceder a un determinado recurso. Requiere de la autenticación previa.

Un objetivo central de la seguridad en redes es garantizar la privacidad de la información y la continuidad de los servicios prestados. En muchos casos se trata de proteger las redes privadas y sus recursos, mientras que se mantienen los beneficios de la conexión a una red pública. Dado que los servicios en Internet no fueron originalmente diseñados para contemplar estos aspectos de seguridad, se han ido incorporando mecanismos para garantizarlos. Las técnicas de encriptación (simétrica o asimétrica) constituyen piezas fundamentales para la implementación de estos mecanismos [Stallings 2002].

En la **encriptación de clave compartida o simétrica**, la misma clave (una cadena cualquiera de bytes) es utilizada para encriptar y desencriptar los datos en ambos extremos de la comunicación. Tanto el emisor como el receptor comparten la misma clave. La encriptación simétrica tiene el beneficio de realizar un rápido procesamiento de los datos, sin embargo posee importantes desventajas, dado que la clave debe ser compartida por ambas partes involucradas en la comunicación, por lo cual es necesario contar con un mecanismo seguro de distribución de claves. Por otro lado, si alguien desea comunicarse con varias personas de manera segura, debe contar con una clave por cada comunicación que desea establecer, lo que resulta poco práctico. Algunos ejemplos de algoritmos de clave simétrica son: DES (*Data Encryption Standard*), IDEA (*International Data Encryption Algorithm*) y 3DES [Stallings 2002].

En la encriptación de clave compartida la seguridad de la clave depende de las dos partes involucradas, es decir que tanto el receptor como el emisor pueden comprometer la clave. Dichas desventajas son resueltas a partir del uso de la **encriptación asimétrica o de clave pública**. Este mecanismo utiliza dos claves diferentes pero matemáticamente relacionadas. Estas claves complementarias se denominan "clave privada" y "clave pública" respectivamente. La finalidad de este sistema es proveer a cada usuario de un único par de claves (una pública y una privada) independientemente del número de usuarios con los que desee comunicarse. Estas claves tienen la propiedad que cada una de ellas invierte la acción de la otra pero, y aquí está el punto más relevante, a partir de una no se puede deducir la otra. Un usuario da a conocer su clave pública a otros usuarios y guarda en secreto su clave privada. Ambas claves pueden ser usadas para cifrar y descifrar datos.

2.2 SSO Centralizados

En la aproximación centralizada, la información de autenticación y de perfiles es gestionada en forma centralizada, por ejemplo, por un portal de Internet. Una base de datos central contiene la información de autenticación de los clientes junto con la información de perfil. En otras palabras, toda la información de su identidad. En cuanto a los aspectos de seguridad y mantenimiento, resulta fácil gestionar una base de datos centralizada que almacena los datos de los usuarios. Además, el control de acceso a estos datos resulta transparente. Dicho control puede realizarse a través de un portal donde cada usuario tiene que ser autenticado y autorizado con ayuda de los datos almacenados en la base de datos. Por otra parte, el establecimiento de un sistema centralizado requiere que todo participante, que delega la autenticación a este sistema tenga que cooperar y confiar en él. Como ejemplo, de sistemas de SSO centralizados, veremos a continuación Kerberos y Passport.

2.2.1 El modelo de Kerberos

Introducción

Kerberos [Tung 1996] fue creado en el Instituto de Tecnología de Massachusetts (MIT) a comienzos de la década de los ochenta, con el objetivo de permitir a los usuarios y servicios autenticarse mutuamente. El nombre Kerberos surge de la Mitología Griega de un perro de tres cabezas que custodiaba las puertas del Hades.

Kerberos proporciona tres servicios de seguridad que son autenticación, integridad y privacidad. La idea detrás de Kerberos es lograr que el usuario, inicie la sesión al principio en un programa cliente y de ahí en más, sea ese programa cliente el encargado de asegurar a los servicios que ese usuario está autenticado.

Los usuarios no requieren inventar claves de encriptación sino que utilizan simplemente contraseñas. Pero esas contraseñas no son utilizadas para encriptar los datos enviados entre clientes y servidores. En lugar de eso, las contraseñas son utilizadas sólo durante el inicio de sesión (*login*) y otras claves creadas dinámicamente son utilizadas para encriptar y desencriptar los datos enviados a través de la red.

En realidad, la clave que un usuario utiliza para iniciar la sesión es un derivado (*hash*) de la contraseña de ese usuario. Un algoritmo de derivación (*hashing*) produce una cadena de bits que es función de la información suministrada como entrada, que no puede ser utilizada para recuperar el valor de entrada original. En otras palabras, el derivado funciona en un sólo sentido. Por lo tanto, dado un valor derivado de una contraseña, es imposible recuperar la contraseña original.

Arquitectura y funcionalidad

Se conoce como "principal" a un usuario o servicio que puede ser autentificado mediante el uso de Kerberos. Todos los principales tienen su propia clave, que se deriva de su contraseña para el caso de los usuarios o se genera aleatoriamente para el caso de los servicios. El servidor de Kerberos en sí mismo, se conoce como el Centro de Distribución de Claves (*Key Distribution Center, KDC*) y tiene acceso a las claves de cada principal en su dominio. Generalmente el KDC no almacena las contraseñas de los usuarios en texto plano sino que almacena directamente el derivado de estas.

El KDC está separado en dos servicios bien distinguidos. Uno el "servidor de autenticación" que es el encargado de otorgar los TGTs y el otro el "servidor de concesión de boletos" que es el encargado de otorgar los boletos para los servidores objetivo (conocido como TGS). Por simplicidad, nos referiremos de ahora en más a los dos servicios como KDC.

El protocolo de Kerberos permite la negociación de los algoritmos de encriptación. Algunas de las implementaciones de Kerberos utilizan, predeterminadamente, DES o RC4.

El hecho de que se utilice la encriptación con claves para enviar datos privadamente es muy obvio, pero es menos obvio cómo se puede utilizar para autentificar, lo cual es la función más importante de Kerberos. Si se envía un mensaje encriptado entre dos entidades, y sólo esas dos entidades conocen la clave de encriptación, entonces si la entidad receptora del mensaje puede desencriptarlo correctamente implica que el mensaje debió haber sido encriptado utilizando la misma clave. Como solamente las dos entidades conocen la clave, implica que quien mandó el mensaje sólo puede ser la otra entidad.

Este sencillo método no es muy práctico a la hora de conectar un cliente con varios servidores. Cada par cliente/servidor tendría que compartir una clave secreta, lo que significa que se necesitaría una contraseña separada para cada servicio que se deseara acceder de manera segura y esto es justamente lo que queremos evitar. Kerberos utiliza un sistema de boletos para resolver este problema, como veremos a continuación.

Boletos

Cuando un usuario desea probar su identidad a una aplicación de servidor en algún otro sistema, ese usuario debe de alguna manera proporcionar al servidor un boleto (*ticket*) apropiado. Cada boleto permite que un usuario específico pruebe su identidad a una aplicación de servidor específica. Un boleto de Kerberos contiene tanto información encriptada como no encriptada.

La parte no encriptada del boleto contiene dos piezas principales de información:
- El nombre del dominio en el cual el boleto fue expedido.
- El nombre del principal al que el boleto identifica.

La parte encriptada del boleto contiene un poco más de información. Aquí están algunos de los campos:
- Varias banderas (*flags*).
- Una clave de encriptación, a la que comúnmente se le refiere como clave de sesión, que puede ser utilizada para encriptar la información que se intercambia entre el usuario de este boleto y el servidor objetivo (*target*) de este boleto.
- Copias del nombre del principal y del dominio.
- Los tiempos de inicio y fin de la validez del boleto.
- Una o más direcciones IP que identifican al sistema del principal.

Existen además otros campos que no se muestran aquí. Todos estos campos están encriptados utilizando la clave de la aplicación de servidor a la que apunta este boleto. Ninguno, ya sea el usuario o cualquier atacante observando el tráfico en la red, puede leer o modificar los campos encriptados de un boleto, dado que no conocen la contraseña del servidor utilizada para la encriptación. Cada boleto tiene, además, un tiempo de vida finito lo que significa que un atacante tiene sólo un período limitado de tiempo para desencriptar el boleto, haciendo un ataque exitoso más difícil debido a que un boleto robado puede ser utilizado solamente hasta que expira.

Notación

La siguiente se ha convertido en la notación estándar para Kerberos:
- K_X: La clave secreta de X (esto es la clave derivada de la contraseña, en el caso del usuario), donde X es un usuario cliente ($_C$), una aplicación de servidor ($_S$) o el KDC ($_K$).
- {Cualquier cosa}K_X: Cualquier cosa encriptada con la clave secreta de X.
- {T}K_S: Un boleto encriptado con la clave secreta del servidor S. En otras palabras, este es un boleto para el servidor S.
- $K_{X,Y}$: Una clave de sesión utilizada entre X e Y (es la clave compartida que usan para mandarse mensajes).
- {Cualquier cosa}$K_{X,Y}$: Cualquier cosa encriptada con la clave de sesión utilizada entre X e Y.

Inicio de sesión

Cuando un usuario quiere probar su identidad a un determinado servidor, debe adquirir un boleto para ese servidor. La primera vez que un usuario pide un boleto de Kerberos es cuando se inicia la sesión con alguna cuenta en un dado dominio. Desde el punto de vista del usuario el proceso es sencillamente ingresar un nombre de usuario, un nombre de dominio y una contraseña en alguna máquina cliente y luego esperar a que el inicio de sesión sea exitoso o falle. A partir de esto se disparan una serie de acciones.

La petición de inicio de sesión del usuario causa que el sistema del cliente envíe un mensaje al KDC (servidor de autenticación, como se puede ver en (1) de la Figura 2.1) del dominio correspondiente. El mensaje contiene varias cosas, entre ellas, el nombre de usuario, información de preautenticación, que consiste de una marca de tiempo (*time-stamp*) encriptado utilizando K_C, la clave derivada de la contraseña del usuario, y una petición de un boleto de concesión de boletos (*ticket-granting ticket, TGT*). Un TGT es sólo un boleto ordinario y como con todos los boletos se utiliza para probar la identidad de un usuario. Pero el TGT se utiliza de manera ligeramente distinta al resto de los boletos, el cliente lo presenta al KDC cuando hace peticiones de boletos para las aplicaciones de servidor específicas. Cuando esta petición llega al KDC, este busca el registro asociado con el nombre de usuario en la base de datos. A partir de este registro, el KDC extrae la clave (derivada de la contraseña del usuario), y luego la utiliza para desencriptar los datos de preautenticación. Si funciona la desencriptación y tiene una marca de tiempo reciente, el KDC puede estar seguro que este usuario es quien dice ser, dado que el usuario ha demostrado conocer el código de acceso correcto. Además, ésto ha sido hecho sin tener que enviar el código de acceso a través de la red. En cambio, si la desencriptación falla, el usuario debe de haber dado un código de acceso erróneo y el inicio de sesión fallará.

Si los datos de preautenticación son validados correctamente, el KDC construye y envía a la máquina cliente un TGT (ver (2) de la Figura 2.1). Como todos los boletos, este contiene el nombre del usuario y el nombre del dominio en el cual fue generado, junto con una clave de sesión ($K_{C,K}$ generada aleatoriamente por el KDC), los tiempo de inicio y fin de validez para este boleto y varias banderas. Como siempre, parte del boleto está encriptado utilizando la clave del servidor al cual se enviará este boleto. Dado que el TGT sólo se utiliza para pedir otros boletos, y dado que sólo el KDC puede dar boletos, la parte encriptada del TGT lo está utilizando K_K, la clave del KDC misma. Junto con el TGT, el KDC también envía a la máquina cliente la clave de sesión $K_{C,K}$. Esta clave de sesión es enviada encriptada, utilizando K_C. Cuando el sistema cliente obtiene este mensaje utiliza K_C para desencriptar la clave de sesión recibida. Esta desencriptación siempre funcionará, dado que sólo a

los usuarios que demostraron conocimiento de la contraseña correcta con la información de preautenticación, se les enviará este mensaje.

Autenticación ante un servicio remoto

Una vez que el usuario ha iniciado la sesión exitosamente, dicho usuario puede comenzar a acceder servicios que se ejecutan en computadoras de la red. Para hacer esto de manera segura, el usuario debe al menos contar con una manera de probar su identidad a estos servicios. El cliente puede presentar un TGT al KDC, pidiéndole un boleto para un servicio específico. Para distinguirlos de los TGTs, estos boletos son a veces llamados boletos de servicio, pero el formato es el mismo en ambos tipos. Ese boleto es luego enviado al servicio objetivo, el cual puede utilizarlo para determinar quien es el usuario. De hecho, después de adquirir el boleto TGT, cada cliente completa el proceso de inicio de sesión al pedir un boleto de servicio para su propia computadora, permitiéndole probar su identidad a los servicios locales.

Si una aplicación cliente requiere acceder a un servicio remoto entonces necesitará adquirir un boleto antes de poder acceder al servio deseado. Ese boleto sirve únicamente para autenticar a ese usuario frente a ese servio particular. Cuando la aplicación cliente hace su primera petición remota al servidor, se hace una petición de boleto automáticamente al KDC (servidor de concesión de boletos, como se puede ver en (3) de la Figura 2.1). Esta petición contiene varias cosas, incluyendo el TGT del usuario, el nombre de la aplicación de servidor para la cual el boleto de servicio se solicita y un autenticador que prueba que este TGT pertenece a este usuario. El autenticador contiene la fecha y hora actuales y el nombre de usuario, y esta encriptado utilizando la clave de sesión $K_{C,K}$ que fue recibida al momento de iniciar la sesión. Cuando el KDC recibe esta petición, desencripta el TGT con la clave K_K, que fue utilizada para encriptar este boleto y que sólo el KDC conoce y luego extrae la clave de sesión $K_{C,K}$ del boleto. Entonces utiliza esta clave de sesión para desencriptar el autenticador. El autenticador sirve para dos propósitos. Primero, dado que está encriptado utilizando la clave de sesión $K_{C,K}$, prueba que el usuario es quien dice ser, porque, como se describió antes, la única manera de tener esta clave de sesión es conociendo la contraseña correcta al momento de iniciar la sesión. Si el intento del KDC de desencriptar el autenticador es exitoso, el sistema cliente debe de estar en posesión de la clave de sesión. Segundo, dado que el autenticador contiene una marca de tiempo, evita que un atacante tome de la red el TGT del usuario y luego lo presente como propio. Un nuevo autenticador es creado cada vez que un boleto es utilizado y dado que el sello de tiempo está encriptado con la clave de sesión, que sólo es conocida por el sistema cliente y por el KDC, un autenticador válido no puede ser creado por nadie más. Para evitar que se reenvíen autenticadores, el KDC rechazará cualquier autenticador cuya marca de tiempo sea demasiado vieja. Las marcas de tiempo de un autenticador pueden diferir en no más de 5 minutos del servidor que lo recibe. Esto implica que los relojes de las máquinas que utilizan Kerberos deben de estar sincronizadas al menos laxamente. Para corroborar que el usuario no está presentando un boleto robado, el KDC también verifica que la dirección IP en el TGT concuerde con la dirección de IP del sistema que envía la petición.

Si las verificaciones son realizadas con éxito, el KDC envía de vuelta el boleto de servicio requerido (ver (4) de la Figura 2.1). El KDC copia algunos de los campos del TGT en el nuevo boleto de servicio, incluyendo el nombre de usuario, dominio e información de autorización. Ajusta las banderas del boleto de servicio y los tiempos de inicio y final apropiadamente, y genera una nueva clave de sesión aleatoria, $K_{C,S}$, la cual coloca en el boleto. Esta nueva clave puede ser utilizada para encriptar la información que es enviada entre el cliente y el servidor. El KDC luego encripta este boleto utilizando la clave secreta del servidor K_S y la envía de vuelta

al cliente, junto con la nueva clave de sesión $K_{C,S}$. Para evitar que los atacantes aprendan esta nueva clave, se envía encriptada utilizando la clave de sesión compartida por el KDC y el cliente, $K_{C,K}$. Finalmente la meta de todo este trabajo puede ser alcanzada permitiendo que el usuario pueda probar su identidad al servidor. Ahora, en su primer petición hacia el servidor, el cliente presenta el boleto de servicio que acaba de recibir junto con un autenticador encriptado utilizando $K_{C,S}$ (ver (5) de la Figura 2.1). El sistema receptor desencripta el boleto con su clave secreta, extrae la clave de sesión $K_{C,S}$ del boleto, luego desencripta y valida el autenticador. Si todo se verifica correctamente, la identidad del usuario es extraída del boleto. Dado que el boleto de servicio que el usuario presentó fue encriptado utilizando la clave secreta del servidor y dado que sólo el KDC y el servidor objetivo conocen esta clave, el boleto, debió de haber sido creado por el KDC. Y dado que el KDC sólo dará boletos de servicio a quienes puedan probar que conocen la contraseña correcta al encriptar la información de preautenticación correctamente, este usuario debe ser quien dice ser. Cuando se presenten con autenticadores válidos, los boletos que son utilizados por Kerberos proporcionan autenticación de cliente confiable.

Aún y cuando Kerberos por sí mismo, no trata con el problema, la información acerca del usuario que es extraída del boleto recibido puede eventualmente ser utilizada para tomar una decisión de autorización. Esto puede ser realizado, dependiendo de la implementación, en el KDC (mediante una extensión del estándar) que indica que usuarios tienen acceso a los distintos servicios. La otra alternativa es que lo delegue al servidor objetivo en cuyo caso, será la aplicación de servidor la encargada de realizarlo. Puede buscar el nombre del usuario en una lista de usuarios autorizados a desempeñar determinadas funciones. Ver Figura 2.1, tabla de relación usuarios y servicios, en contraste con la tabla de control de acceso.

Con la estrategia usada hasta ahora es posible que un cliente demuestre a un servidor que es quien dice ser. Pero, el problema inverso no está resuelto. Es necesario además que el servidor se autentique frente al cliente, para evitar que un posible atacante pueda hacerse pasar por el servidor requerido por el cliente. El estándar Kerberos define una opción para autenticación mutua. No sólo el cliente prueba su identidad al servidor, sino que el servidor debe también probar su identidad al cliente. Para hacer esto, el servidor crea un mensaje conteniendo la marca de tiempo del autenticador del cliente encriptado utilizando la clave de sesión $K_{C,S}$. Cuando este mensaje es recibido por el cliente, puede utilizar su copia de la clave de sesión para desencriptar dicho mensaje. Si el cliente encuentra que el sello de tiempo es el que acaba de enviar, puede estar seguro que el servidor también conoce la clave de sesión. Y dado que el saber la clave de sesión requiere desencriptar el boleto de servicio, lo que requiere conocer la clave del servidor K_S, entonces este servidor tiene que ser el auténtico.

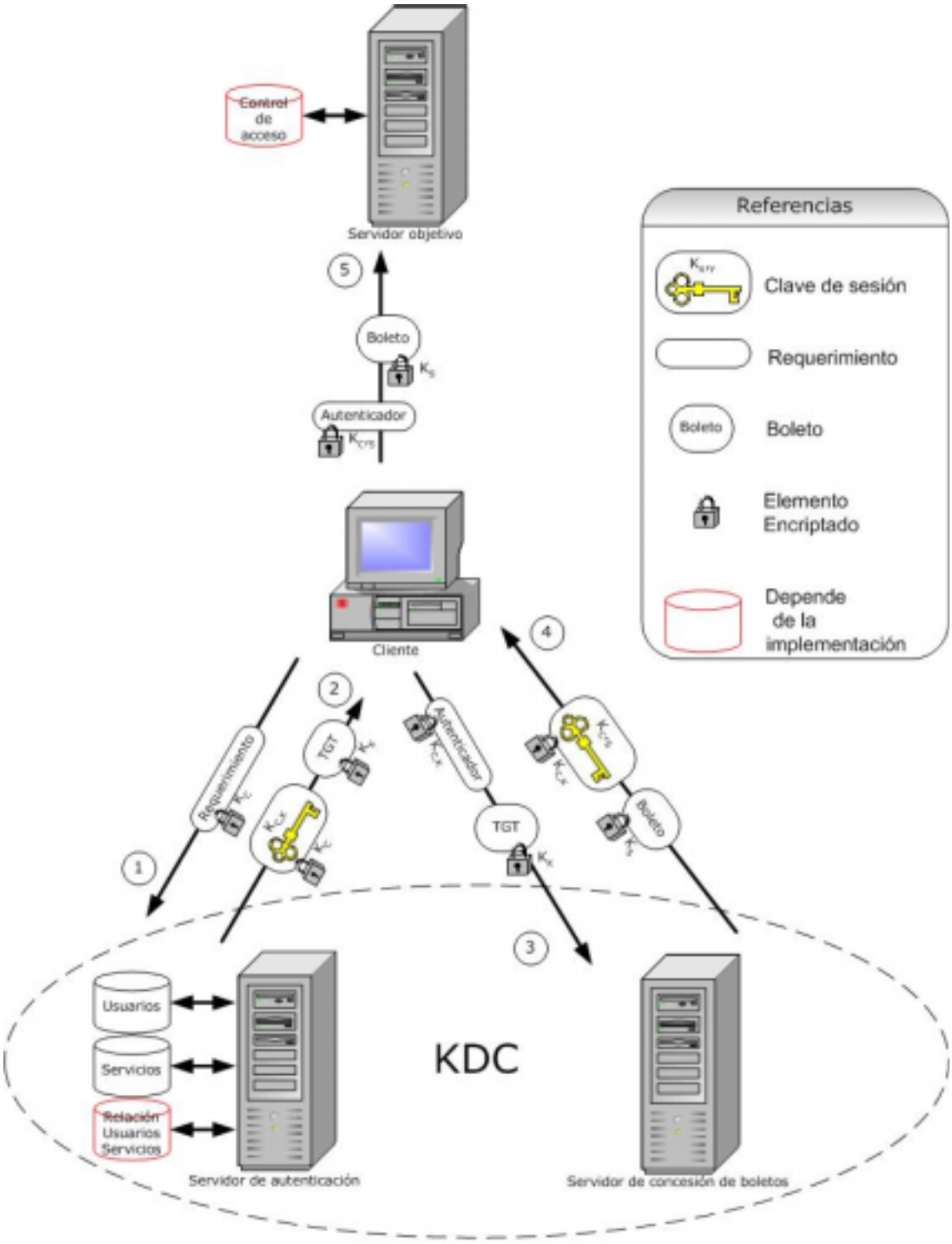

Figura 2.1

Proporcionando integridad y privacidad

Hasta ahora se mostró como Kerberos provee autenticación. Pero Kerberos puede proporcionar, además, integridad y privacidad de datos, otros dos servicios útiles. Debido a los intercambios antes descriptos sabemos que el cliente y el servidor quedaron en posesión de una clave de sesión compartida. Proporcionar estos dos nuevos servicios es sencillo en esta situación.

Para garantizar que ningún atacante pueda modificar los datos transmitidos sin ser detectado, Kerberos en cualquier sistema que esté enviando datos puede calcular una suma de verificación en cada paquete que envía y transmitir esa suma de verificación con el paquete. El valor de la suma de verificación es una función de los datos en los que está basada, así que si los datos cambian, la suma debe cambiar. Pero el enviar sólo un paquete y su suma de comprobación no es suficiente, pues un atacante puede tomar un paquete, modificar los datos, recalcular una nueva suma de verificación basada en los nuevos datos y enviarla a su destino. Para prevenir que esto ocurra, el que envía los datos calcula la suma de verificación no sólo del mensaje en sí mismo, sino del mensaje y otra información, luego encripta el resultado utilizando la clave de sesión. Ahora, ningún atacante puede crear una suma de verificación válida para los datos modificados, dado que ningún atacante conoce esa clave. El resultado es que el receptor del paquete puede detectar cualquier intento de modificación del contenido del paquete.

Proporcionar por último, la privacidad de los datos, es sencillo. Dado que el cliente y el servidor comparten una clave de sesión, Kerberos en cada uno sólo utiliza esta clave para encriptar los datos que se envían uno al otro. Note que la privacidad de datos implica integridad de datos, dado que ningún atacante puede modificar los datos encriptados en su tránsito por la red, sin que esos cambios sean detectados.

2.2.2 El modelo de Passport

Introducción

Hoy en día la solución SSO más popular disponible es Microsoft Passport [MicrosoftTO 2001]. El modelo de Passport consiste de tres entidades:

- El cliente.
- El comerciante.
- El servidor de Passport.

Donde el cliente es generalmente el consumidor con su navegador que se ha registrado previamente al servicio de Passport y el comerciante es generalmente un sitio de comercio electrónico (*e-commerce*). Los sitios de comercio electrónico desean comercializar con el cliente y el servidor de Passport juega el rol de central de autenticación. El servidor de Passport maneja la información de autenticación del usuario, así como los datos de su perfil, lo que le permite interactuar con el sitio de comercio electrónico. Además, el modelo de Passport divide los datos del cliente en información pura del perfil e información de su billetera electrónica (*Passport Wallet*) que contiene datos como los de su tarjeta de crédito. Para ser totalmente correctos, la billetera electrónica es una aplicación basada en Passport. En general, Passport fue desarrollado para permitir SSO en la web y sentar bases para transacciones seguras en el contexto del comercio electrónico.

Arquitectura y funcionalidad

El núcleo de la arquitectura Passport es una base de datos centralizada la cual contiene todos los usuarios registrados, sus datos y credenciales. Para cada usuario Passport genera un identificador único, llamado PUID (*Passport Unique Identifier*). Con la ayuda de éste cada usuario puede ser identificado unívocamente. Passport trata de solucionar los problemas que conciernen al SSO usando tecnologías basadas en web como SSL (*Secure Socket Layer*), *Cookies* y *JavaScript*. Es más, Passport permite a los sitios participantes que elijan entre tres posibles niveles de seguridad en la autenticación según las necesidades:

- Inicio de Sesión Estándar (*Standard Sign On*)
- Inicio de Sesión por Canal Seguro (*Secure Channel Sign On*)
- Inicio de Sesión por Credencial Fuerte (*Strong Credential Sign On*)

El Estándar es preferentemente usado para casos de aplicaciones comunes sin restricciones de seguridad extraordinarias. El de Canal Seguro enriquece el Estándar usando SSL. Y el de Credencial Fuerte permite el máximo nivel de seguridad introduciendo una etapa más al Canal Seguro.

La solución SSO de Microsoft nunca envía la contraseña del usuario a los sitios participantes. La información del perfil de usuario es siempre encriptada para su envío. En ninguno de los tres niveles de seguridad mencionados arriba es necesario instalar una aplicación cliente, sólo basta con un navegador de Internet tal como el Microsoft Internet Explorer, Netscape Navigator u Opera.

Del otro lado, el administrador del sitio participante debe instalar el Microsoft Passport Manager el cual es el único software adicional que Passport requiere. Este software es responsable por la encriptación y desencriptación de las cookies. Es usado para la autenticación, manejo de los datos del perfil de usuario, almacenamiento de la autenticación de usuarios e información de perfiles en las cookies, y finalmente, para tratar las cookies en el proceso de mantenimiento de la sesión de un usuario autenticado.

Supongamos que un usuario entra a un sitio donde la autenticación requerida utiliza Passport, por ejemplo, la página principal de un sitio de comercio electrónico que permite al usuario comprar algo en línea (*on-line*). Si el usuario desea realizar la compra en línea, debe iniciar la sesión (*Sign On*) usando el servidor de Passport. Entonces, el navegador del usuario es redirecionado al servidor de Passport para permitir el inicio de sesión. A partir de este punto, cada uno de los tres niveles posibles de seguridad provistos por Passport, que difieren levemente, podrán utilizarse. En la siguiente sección se describen las bases de Passport usando el procedimiento de Inicio de Sesión Estándar.

Inicio de Sesión Estándar

El Estándar representa el más bajo nivel de seguridad disponible con el modelo de Passport. Con este perfil, SSL sólo es usado cuando el nombre de usuario y la contraseña son transferidos al servidor de Passport. Por lo tanto, los sitios y servicios con necesidades de seguridad mayor, donde los usuarios ganan acceso a datos sensibles como, por ejemplo, información de una cuenta bancaria requerirán un nivel de protección mayor. A pesar de esto, para mostrar los principios detrás de Passport esta sección describe el proceso de autenticación Estándar solamente, dejándose para más adelante la descripción de los dos restantes.

(1). Redireccionando al servidor de Passport
Primero, el cliente sigue el enlace de inicio de sesión (comúnmente representado por el símbolo estándar) del sitio participante al servidor de Passport.

(2). Accediendo al servidor de Passport para iniciar la sesión
El usuario sigue el enlace y su navegador es redireccionado a la página de inicio de sesión del servidor de Passport dedicado. Con esta petición (*request*) se adjunta un identificador único, el cual identifica al sitio participante desde donde el usuario es direccionado. Este identificador, así como también, una clave de encriptación única es inicialmente asignado al sitio cuando éste se registra como un sito participante. El URL de regreso de la petición al sitio participante es embebido en la petición, a fin de redirecionar el navegador del cliente a la página deseada cuando el inicio de sesión sea exitoso.

(3). Ingresando las credenciales de usuario
Después de corroborarse si el identificador del sitio, que fue enviado con la petición, corresponde a un sitio registrado, el servidor de Passport pide al usuario que ingrese su nombre de usuario y contraseña para iniciar la sesión. No hay restricciones para mostrar la página de inicio de sesión del servidor de Passport permitiéndose HTTP o HTTPS, pero el formulario para ingresar las credenciales de usuario utiliza siempre el método POST y se usa HTTPS para enviar las credenciales al servidor. Consecuentemente, cuando el usuario hace el envío (*submit*) de sus datos al servidor de inicio de sesión, ellos son siempre transferidos usando el protocolo SSL como se mencionó antes.

(4). Obteniendo las cookies
El servidor de Passport verifica la información de autenticación recibida. Si se corresponde con la información almacenada en la base de datos del servidor, el usuario se considera autenticado. Luego, el Passport Manager del lado del servidor del sitio participante recupera el PUID e información adicional del perfil si es necesario. Con la ayuda de estos datos el Passport Manager tiene la capacidad de crear cookies representando el estado del usuario. Según la documentación de Passport, hay cinco cookies escritas en el dominio de autoridad y dos más escritas en el dominio del sitio participante. Para explicar los principios de Passport es suficiente tratar sólo con tres:

- La cookie boleto (*the ticket cookie*): que contiene el PUID y la marca de tiempo (*time-stamp*).
- La cookie perfil (*the profile cookie*): que contiene la información del perfil de usuario.
- La cookie de sitios visitados (*the visited sites cookie*): que contiene la lista de sitios donde el usuario ha iniciado la sesión.

Cada una de estas tres cookies importantes está encriptada usando el algoritmo 3DES. La clave usada para la encripción fue inicialmente creada y asignada cuando el sitio se registró como participante del sistema de Passport. El servidor de Passport encripta el contenido de las cookies y retorna el boleto y los datos del perfil agregándolas como parámetro (*query string*) del URL de regreso al sitio participante al cual el usuario quiere acceder como manifiesta dentro del pedido de autenticación. Finalmente, el navegador del cliente es redireccionado de acuerdo con el URL de regreso y el servidor de Passport almacena en la cookie de sitios visitados al participante en cuestión.

(5). Accediendo al sitio participante
El navegador del usuario es redireccionado nuevamente al sitio que el usuario desea siguiendo el URL de retorno provisto por el servidor de Passport. El Passport Manager corriendo en el servidor del sitio participante extrae de los parámetros del URL, el boleto contenido y la información del perfil. Luego de desencriptar la información, el Passport Manager obtiene el PUID, la marca de tiempo y los datos del perfil. De esta forma el sitio participante reconoce que el usuario está autenticado en el servidor de Passport. Si lo desea el sitio puede ser configurado para forzar una reautentición después de un período de tiempo específico. Cuando el tiempo expire, el servidor de Passport mostrará la pantalla de inicio de sesión nuevamente y el usuario deberá reingresar su contraseña para renovar las cookies. Como resultado, el sitio participante almacena la cookie boleto y la cookie perfil contenidas en el URL recibido por el cliente.

(6). Usando el sitio participante
El sitio participante está ahora preparado para mostrar la página personalizada para el cliente usando la información del perfil que viene con la cookie perfil. Además, el sitio puede usar esa información para agregar a su propia base de datos de perfiles, o para crear sus propias cookies. Mientras muestran los recursos deseados al cliente, el sitio participante siempre tiene un enlace para cerrar la sesión (comúnmente representado por el símbolo estándar).

Los pasos del (*1*) al (*6*) pueden observarse en la figura 2.2 para mayor claridad.

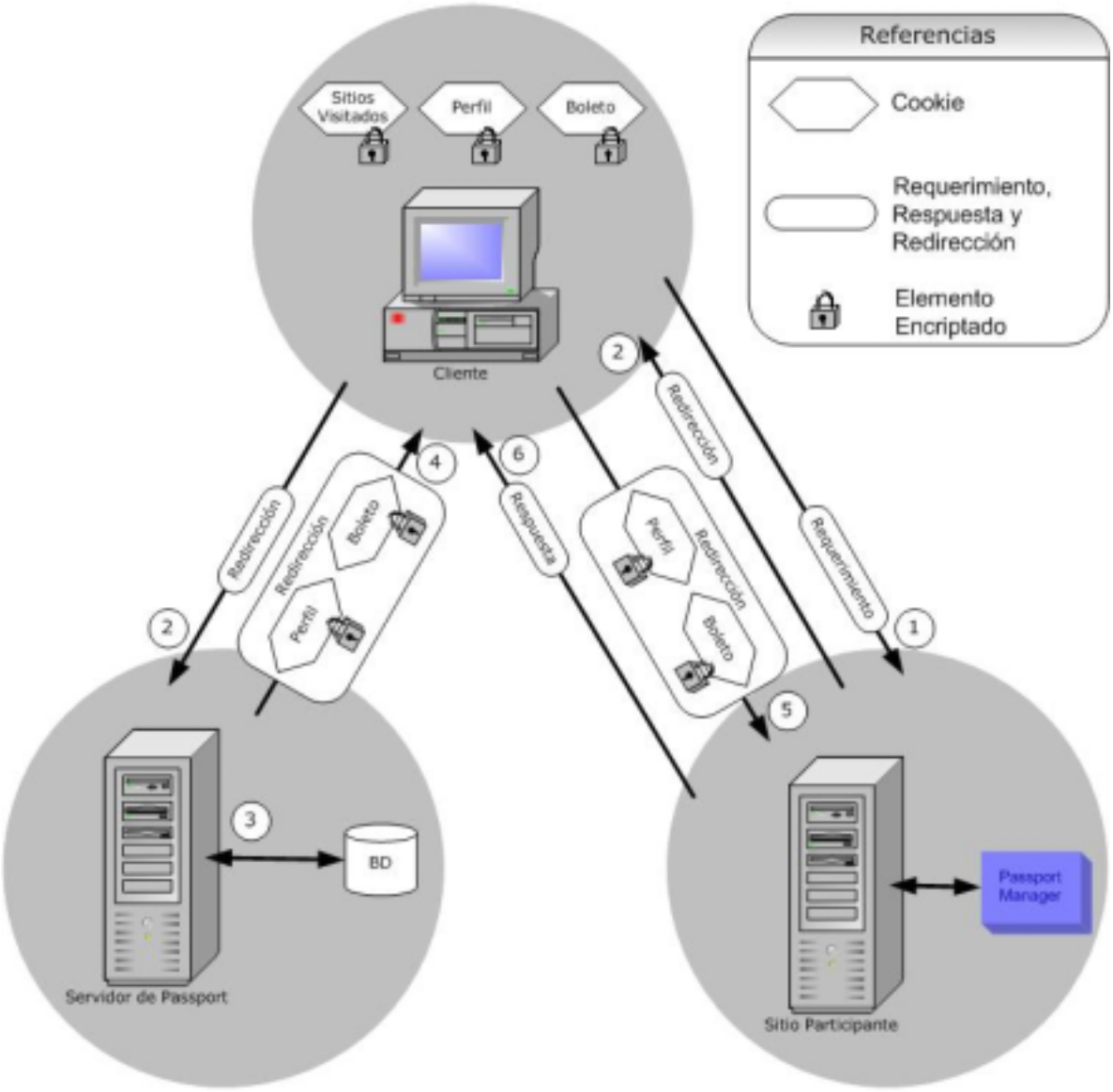

Figura 2.2

(7). Cerrando Sesión

Como se mencionó antes, el usuario puede cerrar la sesión en cualquier momento usando el símbolo de cerrar sesión mostrado en todos lo sitios involucrados. En este caso el navegador del cliente sigue el enlace detrás del símbolo y es redireccionado al servidor de Passport. Después de comprobar el identificador del sitio que originó el pedido de cierre de sesión, el servidor usa la cookie de sitios visitados para borrar todas la cookies creadas al inicio de sesión por los sitios visitados. Para cada cookie a borrar el servidor ejecuta un script en el sitio participante correspondiente, el cuál remueve las cookies. Por el hecho de que sólo el sitio que creó la cookie puede borrarla, cada sitio participante tiene que proveer la URL del script que tiene que ser llamado para borrar sus cookies. La registración de esas URLs sucede durante el proceso inicial de participación. Además, todas la cookies de Passport son temporales. Esto significa que esas cookies son borradas de todas formas, cuando la sesión del navegador es cerrada sin haber cerrado la sesión desde el sistema de Passport previamente. Es más, esas cookies son sensibles al paso del tiempo y expiran después de un cierto período de tiempo especificado por el servidor de Passport o por el sitio participante. Después de que

el contador expira, el usuario es forzado a revalidar su sesión en el servidor de Passport reingresando su contraseña.

El modelo de Passport no usa comunicaciones directas entre el servidor de Passport y el sitio participante. En lugar de ello, la comunicación se realiza mediante el uso de cookies y redireciones a través del cliente. Sólo el Passport Manager instalado en el servidor del sitio participante tiene que recuperar un archivo de configuración (presentado en un documento XML) periódicamente, el cual es almacenado en el servidor de Passport. Este documento XML contiene información actualizada de los URLs del servidor de Passport disponibles y la configuración actual del perfil de Passport.

En el caso de que el usuario ya haya iniciado la sesión en el servidor de Passport y entre a otro sitio participante, será redireccionado una vez más al servidor de Passport cuando haga clic en el símbolo de inicio de sesión. En ese momento el navegador del cliente es redireccionado al servidor de Passport junto con el identificador del sitio participante y el URL de retorno al recurso que el usuario quiere acceder. Nuevamente, el servidor de Passport tiene que comprobar que el identificador del sitio que hace el pedido sea correcto, y también comprueba la validez del boleto de usuario que contiene el PUID y la marca de tiempo. Luego, el servidor de Passport crea nuevas cookies y regresa el boleto y los datos del perfil encriptados al sitio que hizo el requerimiento, agregados como parámetro del URL. De esta manera, es posible entrar a otro sitio después de realizada la autenticación en el servidor de Passport sin requerirle las credenciales al cliente. Es posible, sin embargo, que el sitio participante requiera comprobar una autenticación reciente para mayor seguridad. En este caso, el servidor de Passport es forzado a reautenticar al usuario pidiéndole sus credenciales nuevamente.

Inicio de Sesión por Canal Seguro y Credencial Fuerte

Como se mencionó antes, Passport soporta otros dos mecanismos para el inicio de sesión de acuerdo a las necesidades en el nivel de seguridad. Estos son los Inicio de Sesión por Canal Seguro y Credencial Fuerte que extienden el perfil Estándar.

(1). Inicio de Sesión por Canal Seguro
Como el nombre de este perfil ya dice, lo que hace, es extender el modo Estándar con el uso de un canal seguro SSL de extremo a extremo. Este canal SSL es usado durante todo el proceso de autenticación. Si tal canal no es usado, un atacante escuchando la comunicación entre el navegador del cliente y el servidor de Passport podría realizar un ataque. Por supuesto que el contenido de las cookies está encriptado, pero, robando esa cookie encriptada un atacante puede hacerse pasar por el dueño replicándola para afectar al sitio participante. El perfil Estándar es vulnerable a tales ataques de réplica porque transporta las cookies sin protección usando HTTP. Mientras el atacante monitorea el tráfico de la red puede reconocer cuando una cookie encriptada es enviada. El contenido de la cookie no está en peligro gracias a la encriptación, pero, el atacante puede capturar la cookie y usarla para mandarla al sitio participante en representación del usuario real. Como resultado, el atacante gana acceso al sitio protegido por el tiempo de vida de la cookie robada. Usando SSL como canal no es posible detectar la transmisión de la cookie porque todo el tráfico está encriptado.

Además, este perfil define un nuevo formato para el boleto, con el fin de prevenir que clientes maliciosos puedan modificar sus cookies para hacerse pasar por otro usuario, asegurando la integridad. En otras palabras, el formato usado en este perfil asegura que las cookies recibidas por el servidor de Passport no pueden ser manipuladas sin que la manipulación sea detectada por el sitio participante. Para

resumir, el Canal Seguro es muy similar al procedimiento Estándar excepto que usa SSL para hacer más segura la comunicación.

Cabe aclararse que la seguridad que este mecanismo, protege solamente al canal de comunicación y no a los extremos. La seguridad de los extremos puede verse afectada a pesar de utilizar un canal seguro. Como ejemplo de lo anteriormente expuesto, se puede decir que si una computadora se comunica a través de un canal seguro y a su vez tiene un troyano que roba información confidencial, la falla de seguridad no puede ser resuelta a pesar del mecanismo de seguridad utilizado para encriptar la comunicación.

(2). Inicio de Sesión por Credencial Fuerte
Tanto el perfil de Canal Seguro como el Estándar están limitados en el número de intentos incorrectos que un cliente puede ingresar la contraseña durante el inicio de sesión. En otras palabras, si un atacante trata de adivinar la contraseña de una cuenta de usuario, el servidor de Passport bloqueará la cuenta afectada por unos minutos después de un cierto número de intentos incorrectos. Sin embargo, este mecanismo de seguridad, aún tiene el riesgo de que la contraseña sea vulnerada.

Para reducir el riesgo, Passport extiende el perfil de Canal Seguro introduciendo una segunda etapa donde el usuario tiene que autenticarse a sí mismo otra vez. Si el usuario quiere acceder a un recurso o sitio que está protegido por el perfil de Credencial Fuerte, tiene primero que atravesar un mecanismo de autenticación similar al de Canal Seguro. Si el usuario es autenticado correctamente, se le permite pasar a una segunda etapa de autenticación. Aquí, el usuario requiere ingresar un código adicional de cuatro dígitos, el cuál fue elegido durante el proceso de registración original de Passport. Todas las comunicaciones durante la autenticación son realizadas a través de un canal seguro SSL. Contrariamente a la primera etapa de autenticación donde la cuenta del usuario nunca es bloqueada definitivamente, el número de intentos incorrectos para agregar el código de seguridad adicional se limita a cinco. Después de los cinco intentos incorrectos, el nivel de Credencial Fuerte es bloqueado hasta que el usuario genere una nueva clave de seguridad respondiendo a tres preguntas secretas las cuales fueron elegidas entre diez posibles durante su registración. El mecanismo que es responsable de bloquear la cuenta es reiniciado después de cada inicio de sesión correcto. Es importante mencionar, que sólo la clave de seguridad necesaria para Credencial Fuerte es bloqueada en el caso de cinco intentos incorrectos. El perfil Estándar todavía puede ser usado con las credenciales estándares. De esta forma sitios y recursos que requieren los dos niveles más bajos de seguridad son todavía accesibles por el usuario. Como resultado, este perfil soporta el máximo nivel de seguridad que un sitio participante puede exigir de Passport. Con esto, el sistema no es vulnerable a los ataques de diccionario.

2.3 SSO Federados

Contrariamente a los sistemas de SSO centralizados que utilizan una entidad de autorización central, la versión federada trata con fragmentos de identidad mantenidos por varios servicios, que se los conoce como proveedores de identidad (*identity providers*). Tal proveedor de identidad puede ser un servicio de Internet de cualquier organización como podría ser una empresa que tuviera un portal en la red de redes. La identidad de un usuario consiste no sólo de una cuenta de usuario, la cual contiene toda la información necesaria para todas las posibles tareas de autenticación. Un usuario de hoy en día, posee muchas identidades representadas por diferentes cuentas fragmentadas sobre varios proveedores de identidad. Este método se acerca más a las necesidades y la filosofía de las redes de hoy, especialmente de Internet.

La información de un usuario tal como los datos de su perfil, sus hábitos de compra, etc. son considerados como muy sensibles. Es más, los usuarios quieren decidir ellos mismos cómo y qué información compartir con las organizaciones de su elección.

La identidad de los usuarios de hoy ha sido considerada como consistente de varias identidades independientes representadas por un número de cuentas de usuario localizadas en diferentes proveedores de identidad. De esta forma el usuario puede decidir almacenar información crítica solamente en los proveedores apropiados y de confianza, y no tiene que confiar en una entidad central única de autorización.

Un sistema de SSO federado, el Proyecto Liberty Alliance [Hodges 2002], se describe a continuación. Este sistema es usado para combinar estas identidades distribuidas. Más, el usuario tiene la oportunidad de decidir que fragmento de identidad es usada con un dado servicio o sitio de comercio para iniciar la sesión.

2.3.1 El modelo de Liberty Alliance

Introducción

Liberty Alliance surge del esfuerzo y la cooperación de varias organizaciones y compañías cuyo propósito es alcanzar un nuevo enfoque para introducir confianza en Internet utilizando la infraestructura existente. El Proyecto Liberty Alliance tiene en cuenta que debe permitir interacciones de negocio basadas en relaciones de confianza en medio de varios servicios, los cuales hace tiempo que ya están trabajando independientemente.

Hoy en día, los sitios y servicios en Internet suelen personalizar su apariencia para de esta forma poder satisfacer las necesidades individuales de cada usuario. Es decir, los usuarios crean una cuenta, con un nombre de usuario, con el que luego se identifican frente al sitio o servicio y brindan información sobre su perfil. Después y suponiendo que hablamos de un sitio de comercio electrónico, éste puede convenientemente almacenar los datos que el usuario le brinda, como podrían ser, los datos de la tarjeta de crédito, la dirección de envío de los productos, el número de teléfono y más. Pronto, el conjunto global de atributos distribuidos por varias cuentas sobre la red representa la "identidad de red" del usuario.

A continuación se listan los principales objetivos de este sistema:

- Permitir a los consumidores proteger la privacidad y la seguridad de la información de su identidad de red y permitirles que puedan elegir con que organizaciones desean compartir esa información.
- Permitir a los comerciantes mantener y manejar su relación con los clientes sin la participación de terceros.
- Proveer un sistema de SSO estándar que incluya autenticación y autorización descentralizada desde múltiples proveedores.
- Crear una infraestructura de identidad de red que soporte todos los dispositivos actuales y emergentes de acceso a redes.

Círculo de confianza

La especificación de Liberty Alliance define el círculo de confianza como un modelo que le permite al consumidor o comerciante ingresar sus credenciales una vez y luego compartir esas credenciales entre los miembros. De esta forma el consumidor o comerciante puede moverse libremente entre los sitios de confianza sin la necesidad de identificarse nuevamente, una y otra vez, como lo tendría que hacer si sólo tuviera varias cuentas aisladas. Uno o tal vez más servicios dentro del circulo de confianza actúan como un tipo de autoridad central y los otros servicios tienen relaciones de confianza establecidas con ellos. Las autoridades alojadas en tal círculo son llamadas proveedores de identidad. Los otros servicios restantes son llamados proveedores de servicio. La idea de este proyecto es combinar y afiliar las normalmente varias cuentas de usuario independientes (conocidas como identidades locales de usuario) dentro del círculo y permitir el uso de servicios dentro del círculo después de tener iniciada la sesión en el proveedor de identidad. En otras palabras, un proveedor de identidad es una autoridad que provee la identidad de los usuarios a los servicios afiliados.

La arquitectura de Liberty Alliance consiste de tres componentes principales:
- Los proveedores de servicio.
- Los proveedores de identidad.
- Los usuarios.

Los proveedores de servicio son entidades que simplemente ofrecen servicios. Prácticamente todos los servicios ofrecidos hoy en la web pueden ser considerados como proveedores de servicio, tal como los portales de Internet, los sitios de comercio electrónico, los sitios de servicios bancarios, los servicios gubernamentales, etc. Los proveedores de identidad son proveedores de servicio especiales. Un proveedor de identidad y los otros proveedores afiliados constituyen un círculo de confianza basado en relaciones de confianza establecidas. La tarea de tal proveedor es recoger los fragmentos de identidad de los usuarios en el círculo de confianza para construir una identidad de red única.

Arquitectura y funcionalidad

El Proyecto Liberty Alliance tiene dos facetas principales:

- Federación de identidades
- SSO

La federación de identidades permite a los usuarios afiliar sus cuentas aisladas en los diversos proveedores de servicio a su identidad. Esta capacidad puede ser utilizada cuando existe un acuerdo previo de los proveedores para formar un círculo de confianza. El usuario de identidades federadas tiene que consentir este acuerdo antes de que su cuenta sea federada con la de otros proveedores. Más aún, el consentimiento del usuario tiene que ser registrado y tiene que ser auditable.

La segunda faceta de este proyecto es permitir SSO iniciando la sesión en un proveedor situado dentro del círculo de confianza y subsecuentemente usar varios proveedores a lo largo del círculo sin la necesidad de iniciar la sesión nuevamente.

Con la siguiente ejemplificación se dará una mirada más profunda en las interacciones y en la arquitectura de las identidades federadas del proyecto Liberty Alliance y la solución SSO que propone. En este ejemplo, se utilizan las siguientes entidades y actores:

- Juan Pérez: el usuario.
- Portal del Banco Nacional: el portal de Internet de una institución bancaria (que actúa como proveedor de identidad).
- Sitio de comercio electrónico: un sitio de comercio electrónico (que actúa como proveedor de servicio).

Al principio en este escenario de ejemplo, el usuario Juan Pérez tiene dos cuentas separadas. Una del portal del Banco Nacional y la otra del sitio de comercio electrónico. En cada una de estas cuentas tiene su propio nombre de usuario y contraseña. El ejemplo se separará en dos partes, la primera en donde estas dos cuentas serán utilizadas para federar la identidad de Juan Pérez, y la segunda donde Juan hará uso del SSO.

Federación de identidades (primera parte del ejemplo)

Para la primera parte del ejemplo asumimos que el usuario ya tiene creadas cuentas locales tanto en el proveedor de identidad, como en el proveedor de servicio (el sitio del banco y el sitio de comercio electrónico, respectivamente). Además, asumimos que ambos proveedores se encuentran dentro del mismo círculo de confianza.

El usuario entra al portal del Banco Nacional usando su nombre de usuario local, como por ejemplo: jpe2308. Después de que sus credenciales han sido aprobadas, el usuario consigue el acceso a este sitio. El proveedor de identidad podría pedir al usuario si desea federar su identidad local con la de otro proveedor de servicio, por ejemplo, el sitio de comercio electrónico. Luego, de un breve anuncio sobre la federación de identidades, el usuario tiene que explícitamente estar de acuerdo para federar su cuenta local con la del otro proveedor. El proveedor de identidad tiene que registrar este acuerdo. Unos instantes después, el usuario sigue el enlace que lo lleva al sitio de un proveedor afiliado, por ejemplo, el sitio de comercio electrónico. Éste reconoce que el usuario fue redireccionado por el portal del Banco Nacional y que el usuario tiene iniciada su sesión en el proveedor de identidad. El proveedor de servicio ofrece al usuario federar la identidad del sitio de comercio electrónico con la del portal del Banco Nacional. Asumiendo que el usuario está de acuerdo, el sitio de comercio electrónico le pide que inicie la sesión utilizando su identidad local a éste (por ejemplo: con el nombre de usuario local jperez).

Como resultado, el servidor del sitio de comercio electrónico y el del portal del Banco Nacional federan la identidad del usuario obteniendo el resultado que se puede observar en la figura 2.4. Esto no sucede por el intercambio de nombres de usuario locales, sino intercambiando manejadores únicos generados localmente por ambos proveedores. El sitio de comercio electrónico genera djkjiwu2we2 como un identificador local para el usuario. Mientras, x4dsdic2 es el nombre determinado por el proveedor de identidad (en este caso el portal del Banco Nacional). Si el proveedor de servicio y el proveedor identidades se comunican, siempre tienen que usar el identificador foráneo que utiliza el proveedor de servicio afiliado para identificar la identidad federada. Por supuesto, si ocurre un error durante el proceso de federación los proveedores tienen que informarle al usuario.

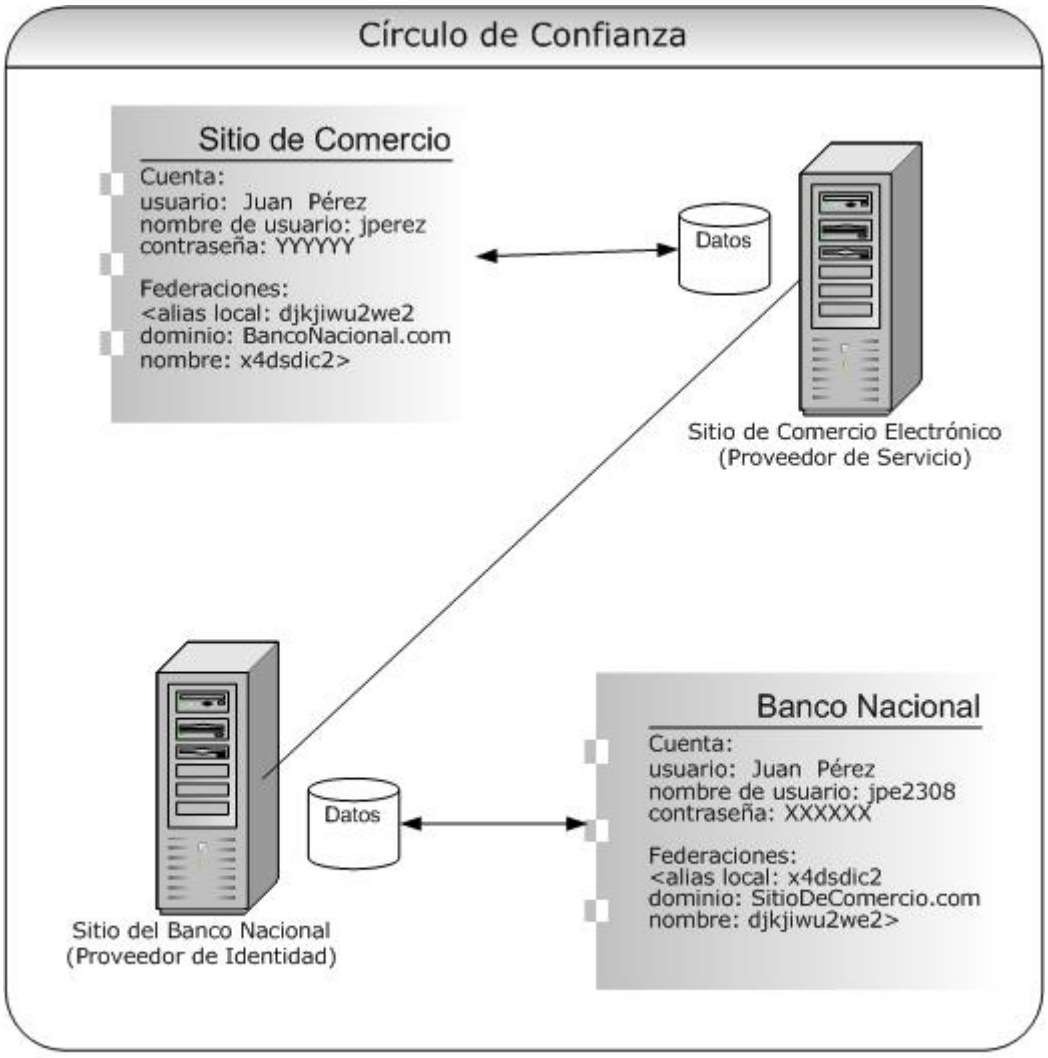

Figura 2.4

SSO (segunda parte del ejemplo)

Para esta segunda parte se asume que el usuario se encuentra en el proveedor de servicio (el sitio de comercio electrónico). Se asume además, que (como se vio en la primera parte del ejemplo) el usuario ya ha federado su identidad entre los dos proveedores (el sitio del banco y el sitio de comercio electrónico) y que no necesariamente se ha autenticado a él mismo en el proveedor de identidad. Por último, se asume que todos los requerimientos son vía HTTP GET.

El usuario está en el proveedor de servicio y decide que requiere algún tipo de servio de los ofrecidos que requiere autenticación (como podría ser la compra de un producto). El usuario informa de alguna manera (con un enlace o botón, por ejemplo) que quiere iniciar la sesión en el proveedor de identidad. Sin importar con que mecanismo el usuario dice "quiero autenticarme a mí mismo". A partir de ese momento se suceden los siguientes pasos:

1. En el sitio del proveedor de servicio, el usuario selecciona un proveedor de identidad para su autenticación (o el proveedor de servicio determina que proveedor de identidad usar, dependiendo de su configuración). Una vez realizada la selección, el navegador del usuario realiza un requerimiento HTTP.

2. El proveedor de servicio determina la dirección del proveedor de identidad y elabora un URI que apunta al proveedor de identidad.
3. El proveedor de servicio responde al navegador del usuario con una repuesta HTTP de redirección (código 302 del protocolo HTTP) y un URI que se encuentra en el campo ubicación (*location*) del encabezado HTTP. Además, ese URI incluye un segundo URI embebido que apunta nuevamente al proveedor de servicio.
4. El navegador del usuario ejecuta un requerimiento HTTP GET al proveedor de identidad cuyo URI estaba en el campo ubicación recibido en el paso 3 como argumento del requerimiento.
5. El proveedor de identidad procesa el requerimiento de HTTP GET. Si el usuario aún no se ha autenticado en el proveedor de identidad lo hace en este momento (usando su cuenta local, lo que sería en el ejemplo jpe2308 y su correspondiente contraseña).
6. El proveedor de identidad responde con una declaración de autenticación. La respuesta es transportada usando una respuesta HTTP de redirección cuyo campo ubicación del encabezado contiene el URI que apunta al proveedor de servicio.
7. El navegador del usuario obtiene la declaración de autenticación. El usuario envía un requerimiento HTTP GET al proveedor de servicio usando el URI que se encuentra en el campo ubicación de la respuesta recibida en el paso 6.
8. El proveedor de servicio, ahora, necesita conseguir una afirmación de autenticación correspondiente a la declaración recibida. El proveedor de servicio manda un mensaje SOAP al proveedor de identidad, pidiendo la afirmación.
9. El proveedor de identidad procesa el requerimiento y responde con la afirmación correspondiente.
10. El proveedor de servicio envía una respuesta HTTP completando el requerimiento original realizado por el usuario en el paso 1.

En la figura 2.3 podemos apreciar como es la interacción entre el usuario, el proveedor de servicio y el proveedor de identidad recién descripta, indicada con la misma numeración.

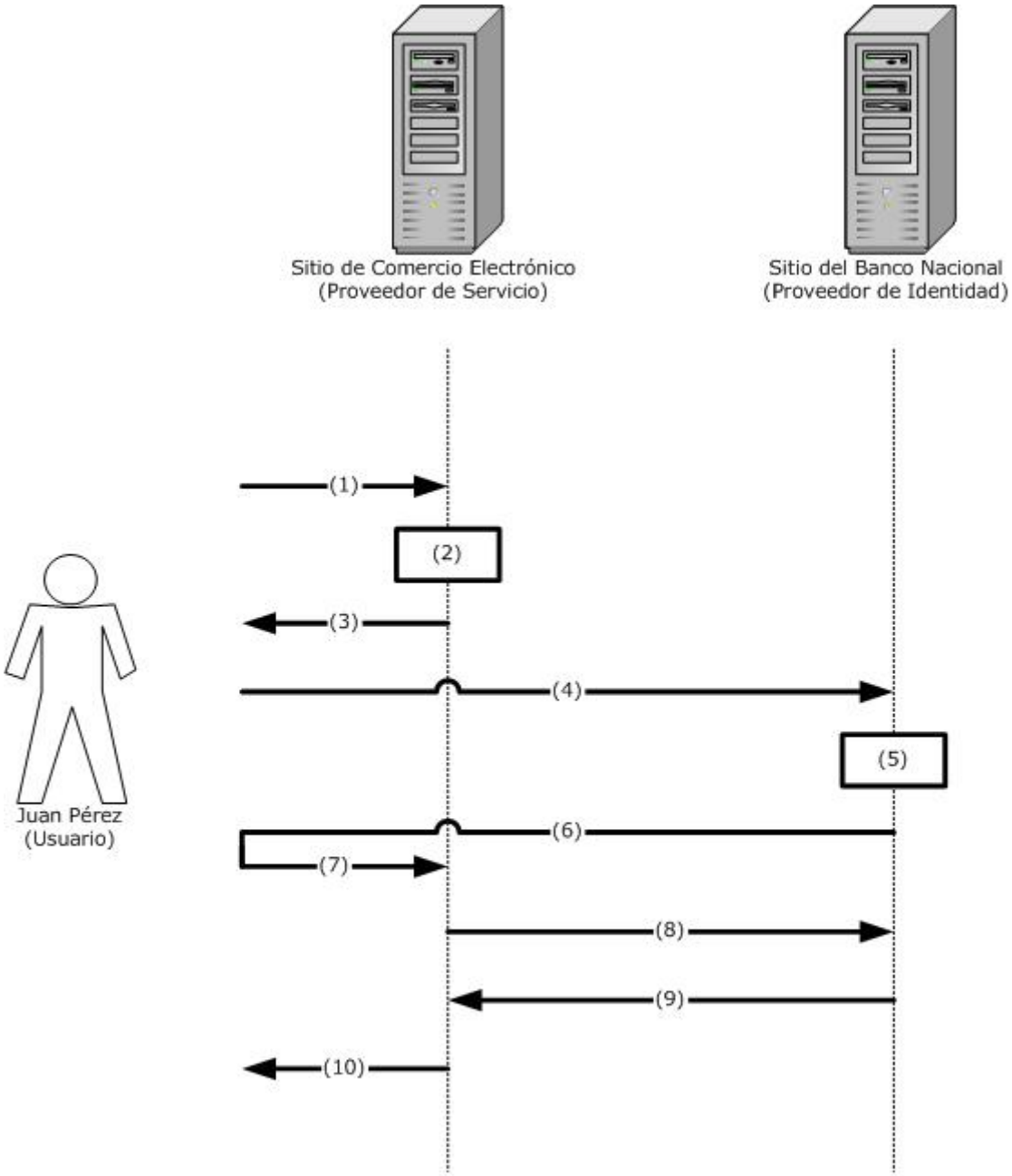

Figura 2.3

Mecanismo de Finalización de Sesión Global

Inclusive para el mecanismo de finalización de sesión, la especificación de Liberty Alliance describe el protocolo, el cual se necesita para la comunicación entre los proveedores afectados. Este proceso puede comenzar tanto en el proveedor de identidad, como en cualquiera de los proveedores de servicio que hayan recibido una autenticación afirmativa de un proveedor de identidad particular. Como resultado, después de comenzado el proceso de finalización de sesión el proveedor de identidad debe contactar a cada proveedor de servicio con el cual haya establecido una sesión para el usuario. Para cada uno de ellos, el proveedor de identidad tiene que mandar una notificación de finalización la cual contenga el nombre identificatorio del usuario que desea finalizar su sesión y el identificador del

proveedor que recibió la notificación. A continuación se verá cual es la secuencia de acciones que se sucede cuando el usuario realiza un pedido de finalización de sesión iniciándola en el proveedor de identidad:

1. El usuario indica que quiere terminar la sesión. Esto lo realiza mediante un requerimiento HTTP GET al URI único de finalización de sesión del proveedor de identidad.
2. Para cada proveedor de servicio para el cual el proveedor de identidad haya provisto de una declaración de autenticación durante la sesión actual de usuario, el proveedor de identidad enviará un HTTP de redirección al usuario. Para cada redirección la ubicación en el encabezado es puesta al URI de finalización de sesión de cada proveedor de servicio.
3. Para cada uno de esos proveedores de servicio, el usuario cumple con el redireccionamiento.
4. Cada proveedor de servicio procesa el redireccionamiento.
5. Cada proveedor de servicio responde al usuario con un HTTP de redirección que apunta al proveedor de identidad.
6. El proveedor de identidad repite el proceso comenzando en el paso 2 si es que hay más de un proveedor de servicio para contactar y sino sigue en el paso 7.
7. El proveedor de identidad responde con un mensaje HTTP al usuario.

Si la finalización de sesión se produce en un proveedor de servicio, sólo cambia el inicio de este proceso. El usuario envía un requerimiento HTTP el proveedor de servicio indicando que desea finalizar la sesión en el proveedor de identidad asociado. El proveedor de servicio responde con un mensaje HTTP de redirección al proveedor de identidad y luego los sucesos son como los ya mencionados.

Federación de proveedores de identidad

El ejemplo previo trata sólo con la federación de las identidades de usuario entre un proveedor de identidad y varios proveedores de servicio. Porque el núcleo de un proveedor de identidad y un proveedor de servicio es similar, la federación de identidad entre varios proveedores de identidad es posible también. Por esa razón es posible combinar identidades a través de varios círculos de confianza.

En otras palabras, es posible combinar y federar identidades entre todos los proveedores sin importar si es un proveedor de servicio o de identidad. Además, es posible conectar varios proveedores de identidad a un proveedor de servicio. Por ejemplo, en el caso que el usuario está registrado en el proveedor de identidad soportado por su compañía y ese mismo usuario es miembro de un segundo proveedor de identidad el cual es usado principalmente para propósitos privados. De esta forma, si el usuario cambia de su oficina a su casa, cambia su proveedor de identidad también. Los proveedores de servicio que él sigue usando son los mismos.

Las especificaciones del Proyecto Liberty Alliance no ponen ningún límite a la federación de identidades. Las bases para todas las federaciones son los acuerdos y relaciones de negocio, y las políticas del círculo de confianza. Basado en esto y en la federación de identidades de usuario locales los proveedores de un círculo de confianza pueden comunicarse con todos los otros acerca de o en beneficio del usurario. No hay límites para la creación de largas cadenas de proveedores de identidad y servicio. A cuenta de esto, inclusive si dos círculos de confianza o especialmente dos proveedores de identidad no tienen relación directa basada en federación, puede existir un camino entre ellos que involucre a otros proveedores. A lo largo de este camino, la identidad de usuario es federada en cada eslabón de la cadena. No hay necesidad para un identificador global único del tipo que, por

ejemplo, Passport requiere. Esta es una sustancial ventaja de la arquitectura federada en comparación con la centralizada.

Capítulo 3

Servicio de Directorio

3.1 ¿Qué es un Servicio de Directorio?

3.1.1 Introducción

Hoy en día, la gente y los negocios dependen de los sistemas de computación distribuidos para soportar aplicaciones distribuidas. Estas aplicaciones distribuidas pueden interactuar con computadoras en la misma red local, con una intranet corporativa, con una extranet que une socios y/o proveedores, o con cualquiera en la Internet global. Para mejorar funcionalidad y facilidad de uso, y para hacer posible una administración efectiva en cuanto a costos de aplicaciones distribuidas, la información acerca de los servicios, los recursos, los usuarios, y otros objetos accesibles desde las aplicaciones necesitan estar organizados de manera clara y consistente. Mucha de esta información puede ser compartida entre muchas aplicaciones, pero debe también estar protegida con el propósito de prevenir modificaciones no autorizadas o el descubrimiento de información privada.

La información que describe a los usuarios, aplicaciones, archivos, impresoras, y otros recursos accesibles desde una red es frecuentemente albergada en una base de datos especial que a veces llamamos directorio. Como el número de redes diferentes y aplicaciones ha crecido, el número de directorios especializados de información ha crecido también, resultando en islas de información que son difíciles de compartir y manejar. Si toda esa información puede ser mantenida y accedida de forma consistente y controlada, podríamos proveer una punto de foco para integrar un ambiente distribuido en un sistema consistente y sin costuras.

El Protocolo de Acceso Ligero a Directorio (*Lightweight Directory Access Protocol - LDAP*) [Johner 1998] es un estándar abierto de la industria que ha evolucionado para encontrar las necesidades de hoy en día. LDAP define un método estándar para acceder y actualizar la información en un directorio. LDAP ha ganado amplia aceptación como el método de acceso a directorio de la Internet y está, por este motivo, transformándose en estratégico para las intranets corporativas. Está siendo incorporado en muchas aplicaciones, por ejemplo los dos navegadores más populares de Internet, Netscape Navigator y Microsoft Internet Explorer, soportan la funcionalidad de LDAP como característica básica.

Este capítulo introduce los conceptos fundamentales de los directorios y el protocolo de uso más común de acceso a directorios, LDAP. También se estudia acerca de cómo están hechos varios componentes de un directorio.

3.1.2 Características de Directorios

Un directorio [Barlen 2002] es una lista de información sobre objetos dispuestos de alguna forma que da detalles acerca de cada objeto. Ejemplos clásicos son, la guía telefónica y los catálogos de fichas de una biblioteca. Para la guía telefónica, los objetos listados son personas. Los nombres son ordenados alfabéticamente y los detalles acerca de cada persona son la dirección y el número de teléfono. Los libros en los catálogos de fichas de una biblioteca son ordenados por autor o por título, y la información dada es el número ISBN del libro y otras cosas referentes a la publicación.

En términos computacionales, un directorio es una base de datos especializada, también llamada repositorio, que almacena información estructurada (respetando tipos de datos) acerca de objetos. Un directorio particular podría listar información sobre impresoras (los objetos) consistente de información tipada tal como la ubicación (una cadena de caracteres con un formato particular), la velocidad expresada en páginas por minuto (un número), los formatos de impresión soportados (por ejemplo PostScript o ASCII), y así siguiendo.

Los directorios le permiten a los usuarios y a las aplicaciones encontrar recursos que tienen las características necesarias para una tarea particular. Por ejemplo, un directorio de usuarios puede ser usado para buscar las direcciones de correo electrónico de las personas o los números de fax. Un directorio, también puede ser explorado para encontrar la impresora color de PostScript más cercana. O un directorio de aplicaciones servidor puede explorarse para encontrar un servicio que pueda acceder a la información anunciada por un cliente.

Los términos "páginas blancas" y "páginas amarillas" son a veces usados para describir como se usa un directorio. Si el nombre de un objeto (persona, impresora) es conocido, sus características (número de teléfono, páginas por minuto) pueden ser recuperadas. Esto es similar a buscar un nombre en las páginas blancas de la guía telefónica. Si el nombre de un objeto particular no es conocido, el directorio puede ser explorado para obtener una lista de objetos que cumplan cierto requerimiento. Esto sería como buscar una lista de peluquerías en las páginas amarillas de la guía telefónica. Sin embargo, los directorios almacenados en una computadora son mucho más flexibles que las páginas amarillas de la guía telefónica, porque pueden ser explorados por un criterio específico, y no solamente por un conjunto predefinido de categorías.

Un directorio es descripto con frecuencia como una base de datos, pero como una base de datos especializada que tiene características que la distinguen de una base de datos de propósito general. Una de estas características especiales de los directorios es que son accedidas (para lectura o búsqueda) mucho más frecuentemente de lo que son actualizadas (escritura). Cientos de personas podrían buscar el número de teléfono de un individuo o las características particulares de una impresora, pero el número de teléfono o las características de la impresora rara vez cambian.

Como los directorios deben estar capacitados para soportar grandes volúmenes de requerimientos de lectura, son por lo general optimizados para accesos de lectura. El acceso a escritura puede estar limitado a los administradores de sistema o a los dueños de cada pieza de información. Una base de datos de propósito general, en cambio, necesita soportar aplicaciones, tales como la reserva de pasajes aéreos o aplicaciones bancarias, con relativamente grandes volúmenes de información a ser actualizada.

Como los directorios están diseñados para almacenar información relativamente estática y son optimizados para tal propósito, no son apropiados para almacenar información que cambia rápidamente. Por ejemplo, el número actual de trabajos en una cola de impresión no debe ser guardado en la entrada del directorio para la impresora porque la información tiene que ser actualizada frecuentemente para ser correcta. En cambio, la entrada en el directorio para la impresora puede contener la dirección de red de un servidor de impresión. El servidor de impresión puede ser consultado para obtener la longitud actual de la cola si se lo desea. La información en el directorio (la dirección de red del servidor de impresión) es estática, mientras que el número de trabajos en la cola de impresión es dinámico.

Otra diferencia, entre los directorios y las bases de datos de propósito general es que los directorios no implementan complicados esquemas para la realización de transacciones que las bases de datos utilizan para llevar a cabo actualizaciones complejas de grandes volúmenes de datos. Por el contrario, las actualizaciones en un directorio son usualmente cambios sencillos.

Las transacciones son operaciones que deben realizarse completamente o no deben realizarse, o sea, deben realizarse por completo o no deben realizarse nada. Por ejemplo, cuando transferimos dinero de una cuenta bancaria a otra, el dinero debe ser debitado de una cuenta y acreditado en otra, en una única transacción. Si sólo la mitad de esta transacción se completa o alguien accede a las cuentas mientras el dinero está en tránsito, las cuentas no estarán balanceadas. Las bases de datos de propósito general habitualmente soportan tales transacciones, las cuales complican su implementación.

Como las bases de datos de propósito general deben soportar aplicaciones arbitrarias, tales como transacciones bancarias y control de inventario, permiten almacenar colecciones de datos arbitrarias. Los directorios podrían estar limitados en los tipos de datos que permiten almacenar (sin embargo la arquitectura no impone tal limitación). Por ejemplo, un directorio especializado para información de contacto de los clientes podría estar limitado para almacenar únicamente información personal tal como nombre, dirección y números de teléfono. Si un directorio es extensible, podría ser configurado para almacenar una variedad de tipos de información haciendo esto más útil a una diversidad de programas.

Otra diferencia importante entre un directorio y una base de datos de propósito general es la forma en que la información puede ser accedida. Muchas bases de datos soportan un estándar, un método de acceso muy poderoso llamado Lenguaje de Consultas Estructurado (*Structured Query Language - SQL*). SQL permite actualizaciones complejas y funciones de consulta. La complejidad de las consultas dependerá del tamaño del programa y la complejidad de la aplicación. Los directorios, tal como un directorio LDAP, por otra parte, usan un protocolo de acceso simple y optimizado que puede ser programado en aplicaciones de forma relativamente simple.

Como los directorios no proveen tantas funciones como las bases de datos de propósito general, pueden ser optimizados para proveer sin demasiado gasto, acceso rápido de las aplicaciones a los datos del directorio en grandes ambientes distribuidos. Si el deseo es utilizar el directorio para lectura, en la mayoría de los casos en ambientes no transaccionales, ambos, el cliente de directorio y el servidor de directorio, pueden ser simplificados y optimizados.

Un requerimiento es realizado típicamente por el cliente de directorio y el proceso que busca información en el directorio lo realiza el servidor de directorio. En general, los servidores proveen un servicio específico para los clientes. A veces, un servidor puede convertirse en cliente de otro servidor con el fin de recolectar la

información necesaria para procesar un requerimiento. Además, los directorios pueden tener la capacidad de replicar información con el fin de aumentar la disponibilidad y la fiabilidad, y a la vez reducir el tiempo de respuesta. Cuando se duplica (o se replica) la información del directorio, pueden aceptarse inconsistencias temporales entre la información que hay en las réplicas, siempre que finalmente exista una sincronización.

3.1.3 Ventajas de un Servicio de Directorio único

Un directorio de aplicación específica almacena solamente la información necesitada por una aplicación particular y no es accesible por otras aplicaciones. Dado que un Servicio de Directorio de funcionalidad completa es complejo de construir, los directorios de aplicación específica son típicamente muy limitados. En general, almacenan solamente un tipo específico de información, no tienen capacidades de búsqueda generales, no soportan replicación ni particionamiento, y probablemente no tengan un conjunto completo de herramientas de administración. Un directorio de aplicación específica podría ser tan simple como un conjunto de archivos de texto editables, o podría ser almacenado y accedido en forma indocumentada y propietaria.

En un ambiente como el descripto, cada aplicación crea y maneja su propio directorio de aplicación específica, lo cual puede convertirse en un caos administrativo. La misma dirección de correo electrónico almacenada por una aplicación de calendario puede también ser almacenada por una aplicación de correo electrónico y por otra que notifique a los administradores del sistema de problemas sobre los equipos. Mantener copias múltiples de la información actualizada y sincronizada resulta dificultoso, en especial cuando coexisten diferentes interfaces de usuario y administradores de sistemas diferentes.

Lo que se necesita es un directorio común independiente de las aplicaciones. Si los desarrolladores de una aplicación pudieran asegurarse de la existencia de un Servicio de Directorio, entonces los directorios de aplicación específica no serian necesarios. Sin embargo, un directorio común trae aparejado una serie de problemas que mencionaremos a continuación. Debe estar basado en un estándar abierto soportado por la mayoría de los vendedores y plataformas. Debe ser accesible a través de un estándar API. Debe ser extensible de manera tal que pueda soportar los tipos de datos necesitados por aplicaciones arbitrarias. Y debe proveer funcionalidad completa sin requerir recursos excesivos en sistemas pequeños. Dado que la mayoría de los usuarios y aplicaciones accederán y dependerán del directorio común, también se requiere que sea robusto, seguro, y escalable.

Cuando tal infraestructura de directorio es instalada, los desarrolladores pueden dedicar su tiempo al desarrollo de la aplicación en lugar de a los directorios de aplicación específica. Así como los desarrolladores descansan sobre la infraestructura de comunicaciones de TCP/IP y en el Llamado a Procedimientos Remotos (*Remote Procedure Call - RPC*) para liberarse de los problemas de comunicación de bajo nivel, podrán descansar en servicios de directorio de funcionalidad completa. LDAP es el protocolo a ser utilizado para acceder a la infraestructura de directorio común. Así como, HTTP (*Hypertext Transfer Protocol*) y FTP (*File Transfer Protocol*), LDAP se ha vuelto una parte indispensable del conjunto de protocolos de Internet.

Cuando las aplicaciones acceden al directorio común estándar que fue diseñado en forma adecuada, se elimina la administración costosa y redundante, y los riesgos de seguridad se vuelven más controlables. Por ejemplo, una agenda, un cliente de

correo electrónico, o un servidor web, pueden acceder al mismo directorio y recuperar información almacenada en una entrada, como podemos ver en la figura 3.1. La ventaja es que los datos son almacenados y mantenidos en un sólo lugar. Varias aplicaciones pueden utilizar atributos individuales de una entrada para diferentes propósitos. Nuevos usos de la información del directorio irán surgiendo a medida que se incremente el número de aplicaciones que utilicen el directorio.

Como conclusión, podemos decir que el almacenamiento de los datos en un directorio y la posibilidad de compartirlo entre varias aplicaciones permite ahorrar tiempo y dinero, disminuyendo los esfuerzos de administración y mantenimiento de los recursos del sistema, así como también, evitando la inconsistencia y redundancia de los datos.

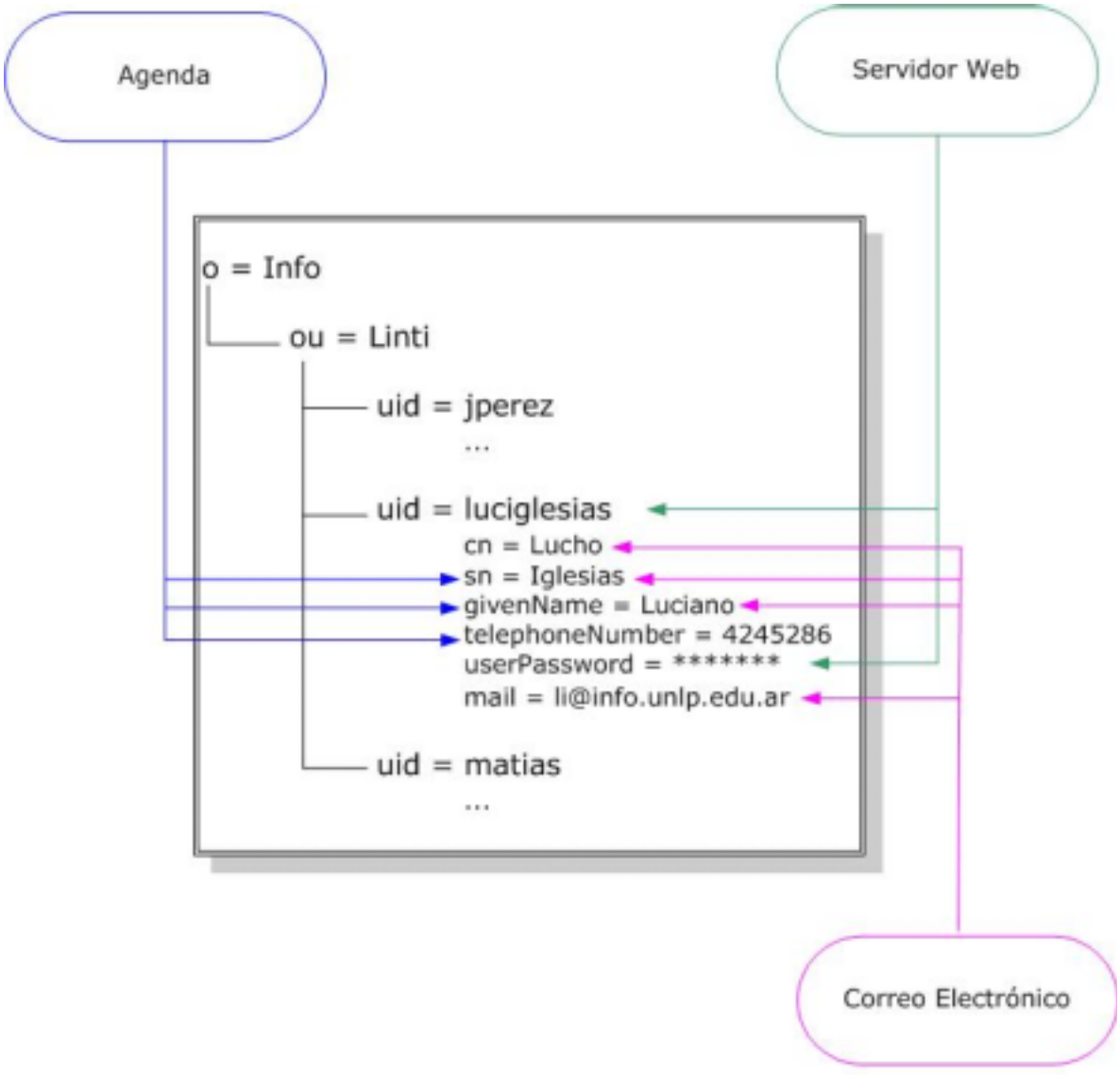

Figura 3.1

3.1.4 Estructura del Directorio

Un directorio contiene una colección de objetos organizados en una estructura de árbol. El modelo LDAP define como las entradas son identificadas y organizadas. Las entradas son organizadas en una estructura tipo árbol denominada Árbol de Información de Directorio (*Directory Information Tree - DIT*). Las entradas están organizadas dentro del DIT basándose en su Nombre Distinguido (*Distinguished Name - DN*). Un DN es un nombre único que identifica en forma no ambigua a una sola entrada. Los DNs están conformados por una secuencia de Nombres Distinguidos Relativos (*Relative Distinguished Name - RDNs*). Cada RDN en un DN corresponde a una rama en el DIT que va desde la raíz del DIT a la entrada del directorio. Un DN está compuesto de una secuencia de RDNs separados por coma, tal como uid=luciglesias,ou=linti,o=info.

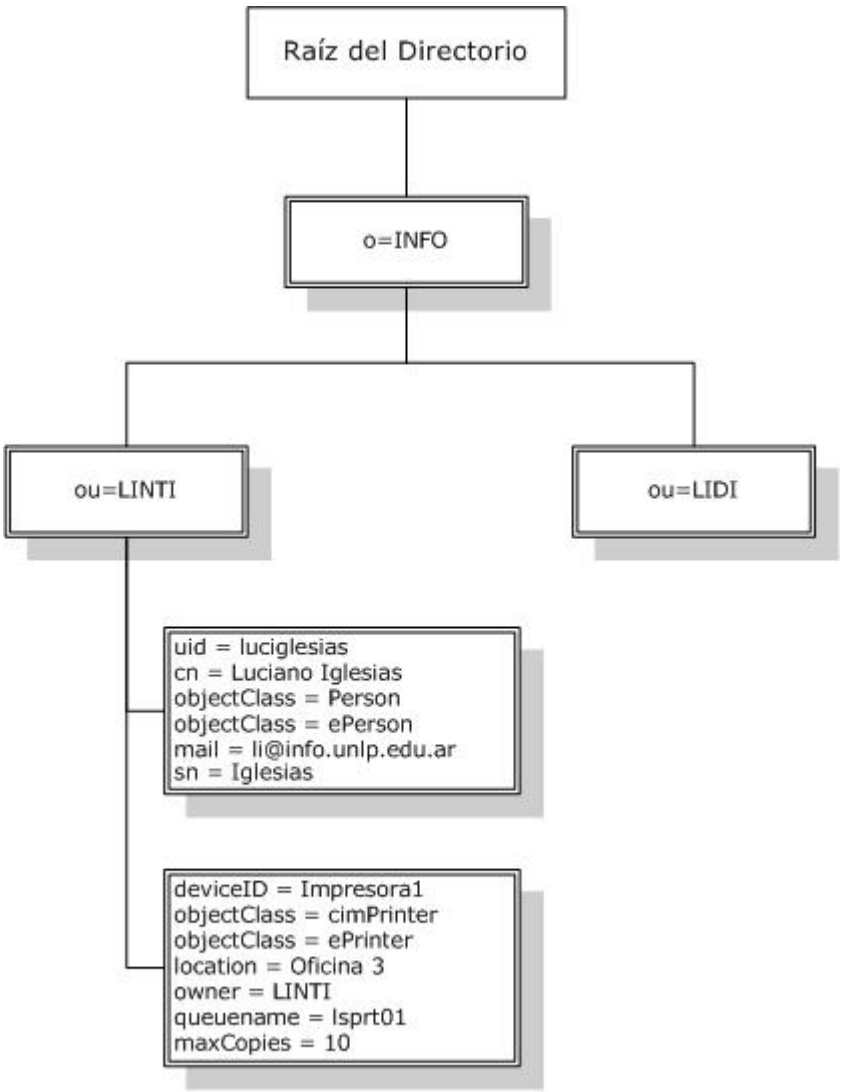

Figura 3.2

Se puede definir el propio DIT basándose en las necesidades organizacionales de cada organización. Si por ejemplo se tiene una compañía con diferentes divisiones, se puede comenzar con el nombre de la compañía debajo de la raíz, como la organización (o), y luego bifurcar en unidades organizacionales (ou) para las divisiones individuales. En el caso que se almacenen datos para múltiples

organizaciones en un país, se podría empezar con un país (c) y luego bifurcar en organizaciones.

Cada objeto es referido como una entrada en un directorio que pertenece a una o más clases de objetos. Una clase de objetos describe el contenido y el propósito del objeto. También contiene una lista de atributos tales como el número de teléfono o el apellido, que pueden ser definidos en un objeto de dicha clase. Se pueden publicar entradas de diferentes clases de objetos debajo de otro objeto. Por ejemplo, los objetos ePrinter y Person, son publicados debajo de la unidad organizacional LINTI como se ve en la figura 3.2.

La clase de un objeto también define cuales de los atributos son requeridos cuando se crea un objeto de esa clase y cuando los atributos son opcionales. Como se muestra en la figura 3.3, la clase de objeto con el nombre ePrinter tiene un atributo requerido y tres atributos opcionales que pueden o no ser llenados cuando se crea un objeto ePrinter. Las clases de objetos pueden también heredar características, tal como atributos de otras clases de objetos. En el ejemplo de ePrinter, la clase hereda todos los atributos que están definidos en la clase cimPrinter. Esto significa que cuando se crea un objeto ePrinter se tiene que definir el identificador del dispositivo (*deviceID*) y opcionalmente la ubicación (*location*), el propietario (*owner*), y la cola de impresión (*queuePtr*) de ePrinter y todos los atributos de cimPrinter.

También los atributos por si mismo tienen ciertas características. Por ejemplo, el nombre del atributo apellido es definido como sn y surName, y describe el apellido de una persona. La definición de atributos también especifica las reglas de sintaxis para el valor del atributo. Un número de teléfono puede solamente contener números y guiones mientras que el apellido consiste de caracteres alfabéticos. Otras especificaciones incluyen si el atributo puede contener solamente uno o varios valores, las reglas de correspondencia, el identificador del objeto (OID), etc. Algunas compañías tales como IBM y Microsoft han agregado algunas extensiones. Por ejemplo, las extensiones de IBM en el servidor iSeries incluyen una clase "access", la cual es usada en combinación con las Listas de Control de Acceso (*Access Control Lists - ACLs*) para controlar quien puede realizar una determinada acción sobre el valor de un atributo, tal como leer, escribir, buscar o utilizar operaciones de comparación.

Todos los objetos y atributos con sus características son definidos en esquemas. El esquema especifica qué puede ser almacenado en el directorio y qué no. La validación de esquemas asegura que todos los atributos requeridos para una entrada estén presentes antes de que la entrada sea almacenada. La validación de esquemas también asegura que los atributos que no están en el esquema no sean almacenados en la entrada. Los atributos opcionales pueden ser completados en cualquier momento. Un esquema también define la herencia y las subclases de objetos y dónde pueden aparecer los objetos dentro de la estructura del DIT.

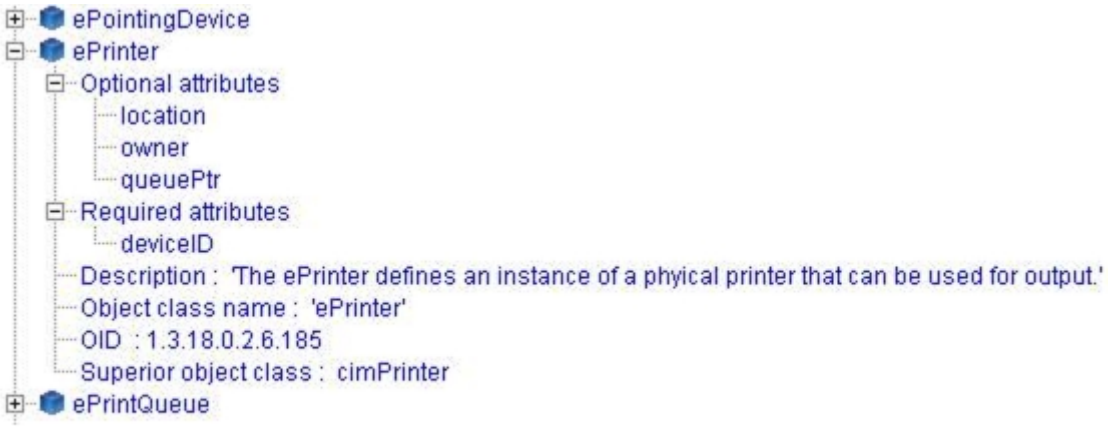
ePointingDevice
ePrinter
Optional attributes
location
owner
queuePtr
Required attributes
deviceID
Description : 'The ePrinter defines an instance of a phyical printer that can be used for output.'
Object class name : 'ePrinter'
OID : 1.3.18.0.2.6.185
Superior object class : cimPrinter
ePrintQueue

Figura 3.3

3.2 Descripción de Ldap

El Protocolo de Acceso Ligero a Directorio (*Lightweight Directory Access Protocol - LDAP*) es un protocolo de tipo cliente-servidor para acceder a un Servicio de Directorio. Surgió en 1993 en la Universidad de Michigan.

LDAP define un protocolo de mensajes usado por los clientes y servidores de directorio. El protocolo LDAP usa diferentes mensajes. Por ejemplo, un *bindRequest* puede ser enviado desde un cliente a un servidor LDAP para iniciar una conexión. Un *searchRequest* se utiliza para buscar una entrada específica en el directorio.

Una API define la interfaz de programación que un lenguaje de programación particular utiliza para acceder a un servicio. El formato y los contenidos de los mensajes intercambiados entre el cliente y el servidor deben cumplir con un protocolo previamente acordado. Hay APIs asociadas con LDAP para distintos lenguajes de programación como pueden ser C y JAVA. El cliente no depende de la implementación particular del servidor, y el servidor puede implementar el directorio como sea que él elija.

LDAP es un estándar abierto de la industria que define un método estándar para el acceso y la actualización de información en un directorio. LDIF (LDAP Data Interchange Format) es un estándar que representa las entradas de un directorio LDAP en formato textual. LDAP ha ganado amplia aceptación como el método de acceso a directorios de Internet, y también se ha convertido en un componente estratégico para las intranets corporativas. Está siendo soportado por un creciente número de vendedores de software que lo están incorporado en numerosas aplicaciones.

LDAP define un protocolo de comunicación, esto es, define la forma de transporte y el formato de los mensajes utilizados por un cliente para acceder a datos en un directorio del estilo X.500. LDAP no define al Servicio de Directorio en sí mismo. Cuando se habla sobre un directorio LDAP, se hace referencia a la información que es almacenada y que puede ser recuperada por el protocolo LDAP.

X.500 organiza las entradas de directorio en un espacio de nombres jerárquicos capaz de soportar grandes volúmenes de información. También define potentes capacidades de búsqueda para que resulte fácil la recuperación de dicha información. Debido a su funcionalidad y escalabilidad, X.500 es utilizado con frecuencia en módulos agregados para la interacción entre servicios de directorio incompatibles.

X.500 especifica que la comunicación entre el cliente de directorio y el servidor de directorio utiliza el Protocolo de Acceso a Directorio (*Directory Access Protocol - DAP*). Sin embargo, como un protocolo de capa de aplicación, DAP requiere de toda la pila de protocolos OSI (*Open System Interconnection*) para operar. Soportar la pila OSI requiere más recursos de los que están disponibles generalmente, en ambientes pequeños. Por lo tanto, se buscó una interfaz para un servidor de directorio X.500 que resultase más liviana en cuanto al uso de recursos y así surgió LDAP (Lightweight DAP), que funciona sobre la pila TCP/IP.

Capítulo 4

Esquema de Single Sign On propuesto

4.1 Introducción

Como se mencionó antes, las organizaciones padecen un legado tecnológico de años de evolución. Entre la diversidad de aplicaciones de una organización existen muchas que requieren algún tipo de inicio de sesión por parte del usuario que le permite a las aplicaciones verificar la identidad de ese usuario (autenticación) y determinar si tiene o no tiene permisos para acceder a dichas aplicaciones (autorización).

Para lograr que la autenticación y la autorización no sean realizadas por cada aplicación particular debemos hallar un mecanismo que nos permita extraer esa tarea de las aplicaciones existentes, y realizarla contra un repositorio único de datos que almacene la información de los usuarios y que permita acceder a ella mediante la utilización de un estándar abierto. Para llevar esto a cabo, se eligió un servidor de directorio como repositorio de datos, que respete el estándar LDAP aprovechando su creciente aceptación.

Utilizar un esquema de SSO evita que cada aplicación realice la autenticación por sí misma, y permite que esta tarea quede a cargo de un nuevo módulo de software. Este nuevo módulo debe tener algún mecanismo que realice la verificación de las credenciales de un usuario en el servidor de directorio, y también que establezca la comunicación con las aplicaciones de manera tal que se pueda realizar el inicio de sesión en ellas, así como también, la finalización.

En este trabajo de grado se implementó un sistema de SSO y se probó con las aplicaciones web, Koha (Sistema de gestión de bibliotecas) [Www Koha] y Guaraní (Sistema de gestión de alumnos para unidades académicas) [Www SIU]. Para ello se desarrolló una aplicación que permite realizar el inicio de sesión por parte del usuario, validando sus credenciales en un servidor de directorio. Una vez autenticado el usuario, éste tiene la posibilidad de ingresar a las aplicaciones, para las cuales tiene permiso, sin la necesidad de identificarse nuevamente.

La incorporación del esquema de SSO propuesto trae aparejadas algunas ventajas como evitar la redundancia e inconsistencia de información (a través del repositorio único de datos), centralizar la administración de las credenciales de acceso, hacer transparente para el usuario la entrada y salida a las aplicaciones, permitir un único punto de acceso, disminuyendo de manera significativa la vulnerabilidad del sistema, etc.

4.2 Esquemas actuales utilizados por las aplicaciones involucradas

4.2.1 Koha

Koha es un sistema de código abierto (*open source*) para la gestión de bibliotecas. Fue realizado en Nueva Zelandia por la Horowhenua Library Trust y Katipo Communications Ltd, en 1999 y entró en producción en enero del 2000. Es un sistema web, desarrollado en Perl, que brinda toda la funcionalidad requerida por una biblioteca, como por ejemplo, catálogos, manejo de miembros, adquisiciones, etc.

Con el esquema actual de Koha la autenticación se realiza mediante una consulta al servidor de la base de datos (*MySql*) que permite verificar la existencia de un usuario con el identificador y contraseña ingresados. De acuerdo con la información almacenada en la base de datos se determina si el usuario está o no autenticado para acceder a los módulos del sistema.

Cuando un usuario dispara el mecanismo de autenticación, el servidor web, ejecuta el script pertinente realizando una consulta a la base de datos. Si como resultado de la consulta se obtienen los datos del usuario entonces se lo considera autenticado, de lo contrario se lo considera rechazado. Luego, si la autenticación fue aceptada se realiza una nueva consulta a la base de datos para recuperar su perfil.

4.2.2 Guaraní

Guaraní es un sistema para la gestión académica desarrollado por el SIU (Sistema de Información Universitaria). Este sistema se ha concebido con la finalidad de brindarle a las universidades, y específicamente dentro de estas a sus unidades académicas una herramienta que les permita administrar la gestión de alumnos de forma segura, con la finalidad de obtener información consistente para los niveles operativos (sector alumnos) y directivos (decano, secretario académico, etc.). Entre sus módulos, cuenta con una aplicación web, desarrollada en php, para la interacción con el alumnado.

La autenticación del módulo web de Guaraní, es muy similar a la recién mencionada para Koha. Se realiza una consulta en el servidor de base de datos (*Informix*) utilizando el identificador y la contraseña ingresados por el usuario. Luego, de acuerdo con la información almacenada en la base de datos se determina si existe tal usuario y tiene dicha contraseña, y de ser así se lo considera autenticado. Al igual que en el caso anterior, si la autenticación es satisfactoria se realiza una nueva consulta para recuperar el perfil del usuario.

4.3 Descripción de la propuesta de SSO

4.3.1 Descripción general

En este trabajo de grado se desarrolló un sistema de SSO que para aplicaciones web. La arquitectura del sistema permite la inclusión de cualquier aplicación web que cumpla ciertos requerimientos. Koha y Guaraní son utilizados sólo a modo de ejemplo en el desarrollo.

Los requerimientos para que una aplicación web pueda formar parte de este esquema de SSO son, que tenga un modelo de autenticación basado en nombre de usuario y contraseña, que pueda ser invocado vía un requerimiento HTTP, y que utilice cookies para mantener la sesión que se establece con el cliente. Gran parte de las aplicaciones web que hacen autenticación de usuarios, cumplen las características recién mencionadas.

La arquitectura aquí propuesta requiere la utilización de un servidor de directorio LDAP y el desarrollo de una aplicación web. El directorio se utiliza para almacenar los datos de los usuarios. Mientras que la aplicación web, permite hacer el inicio de sesión, la finalización de sesión y el acceso a las aplicaciones a las que un usuario dado tiene permiso.

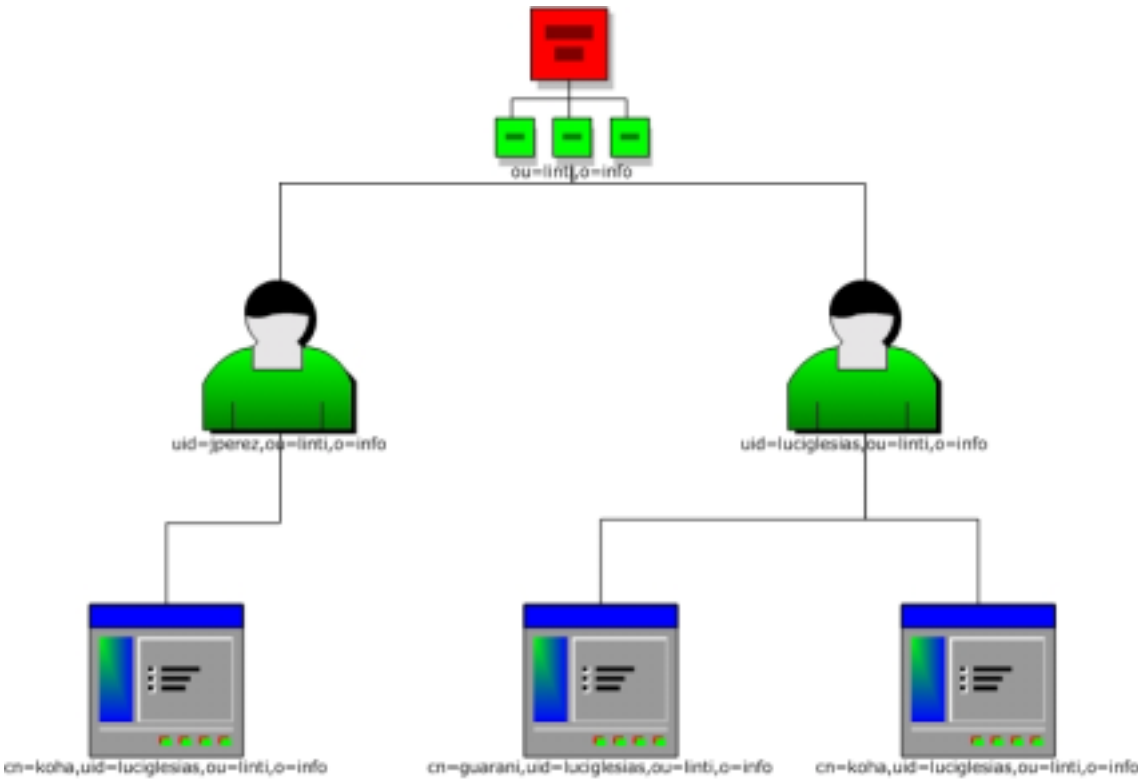

Figura 4.1

La estructura de árbol del directorio incluye una entrada por cada usuario, donde se almacenan todos sus datos (como podemos ver en la figura 4.1 con los usuarios jperez y luciglesias). Entre esos datos de usuario tienen que estar, por lo menos, el identificador y la contraseña. Como hijos de cada usuario se agregan al directorio entradas que almacenan la información de las aplicaciones a las que tiene acceso (como también podemos ver en la figura 4.1). Esto tiene dos propósitos, el primero es almacenar la información necesaria para poder acceder a las aplicaciones y el segundo es hacer las veces de Lista de Control de Acceso (*Access Control List* -

ACL) para indicar a qué aplicaciones un determinado usuario tiene acceso. En el ejemplo de la figura 4.1, el usuario luciglesias puede acceder a dos aplicaciones (Koha y Guaraní), mientras que el usuario jperez sólo lo puede acceder a una (Koha). Entre los datos que se guardan en cada entrada correspondiente a una aplicación web están el nombre de la aplicación, el nombre de usuario y la contraseña particulares de dicha aplicación para ese usuario, el URL de la página que hace el inicio de sesión, el URL de la página que hace la finalización de sesión, los nombres de los inputs que almacenan el nombre de usuario y contraseña, y la forma en que se envían los datos a cada aplicación (POST o GET).

Se creó a modo de prueba la aplicación web (realizada en php) que permite hacer la autenticación. Cuando el usuario ingresa su nombre de usuario y contraseña, éstas son utilizadas para hacer la autenticación (*binding*) en el servidor de directorio. Si tiene éxito, entonces se crea una página que tiene los enlaces a cada una de las aplicaciones a las que ese usuario tiene acceso. Cuando el usuario sigue alguno de los enlaces a las aplicaciones, se accede a las mismas con la sesión iniciada en dicha aplicación para ese usuario. Además, cuenta con la posibilidad de finalizar la sesión global, o sea, para todas las aplicaciones a las que accedió.

4.3.2 Inicio de sesión

El inicio de sesión es la primer tarea que el usuario debe realizar para poder acceder a las aplicaciones que necesita para trabajar. Consta de una primera página donde se escriben el nombre de usuario y la contraseña (únicos identificadores que el usuario requiere recordar). Es un típico cuadro de inicio de sesión, como el que tienen muchos sitios web que requieren autenticación. Es la página de login único de la figura 4.2.

Luego de que el usuario completa los datos y manda a ejecutar el proceso de autenticación, se ejecuta una segunda página (script de inicio de sesión en el directorio de la figura 4.2). Esta segunda página realiza una serie de actividades. Primero intenta conectarse al servidor LDAP. Si la conexión tiene éxito continúa en el próximo párrafo y sino se tiene que optar entre algunas opciones, como intentar conectarse contra un servidor secundario (aunque esto no se analizó, ni se implementó) o terminar con un error.

Si la conexión fue exitosa, se utiliza el nombre de usuario ingresado para conformar el DN, del árbol del directorio, de la entrada correspondiente al usuario que intenta iniciar la sesión. Siguiendo el ejemplo utilizado en la figura 4.1, donde el RDN es ou=linti,o=info y suponiendo que el usuario ingresó como nombre de usuario luciglesias, se compone el DN siguiente uid=luciglesias,ou=linti,o=info. Después, se utiliza el DN recién creado junto con la contraseña ingresada para hacer la autenticación en el directorio (*binding*). Si la autenticación tiene éxito, se deduce que el usuario conoce un nombre de usuario y contraseña válidos para hacer el inicio de sesión. A partir de ese momento se considera al usuario autenticado.

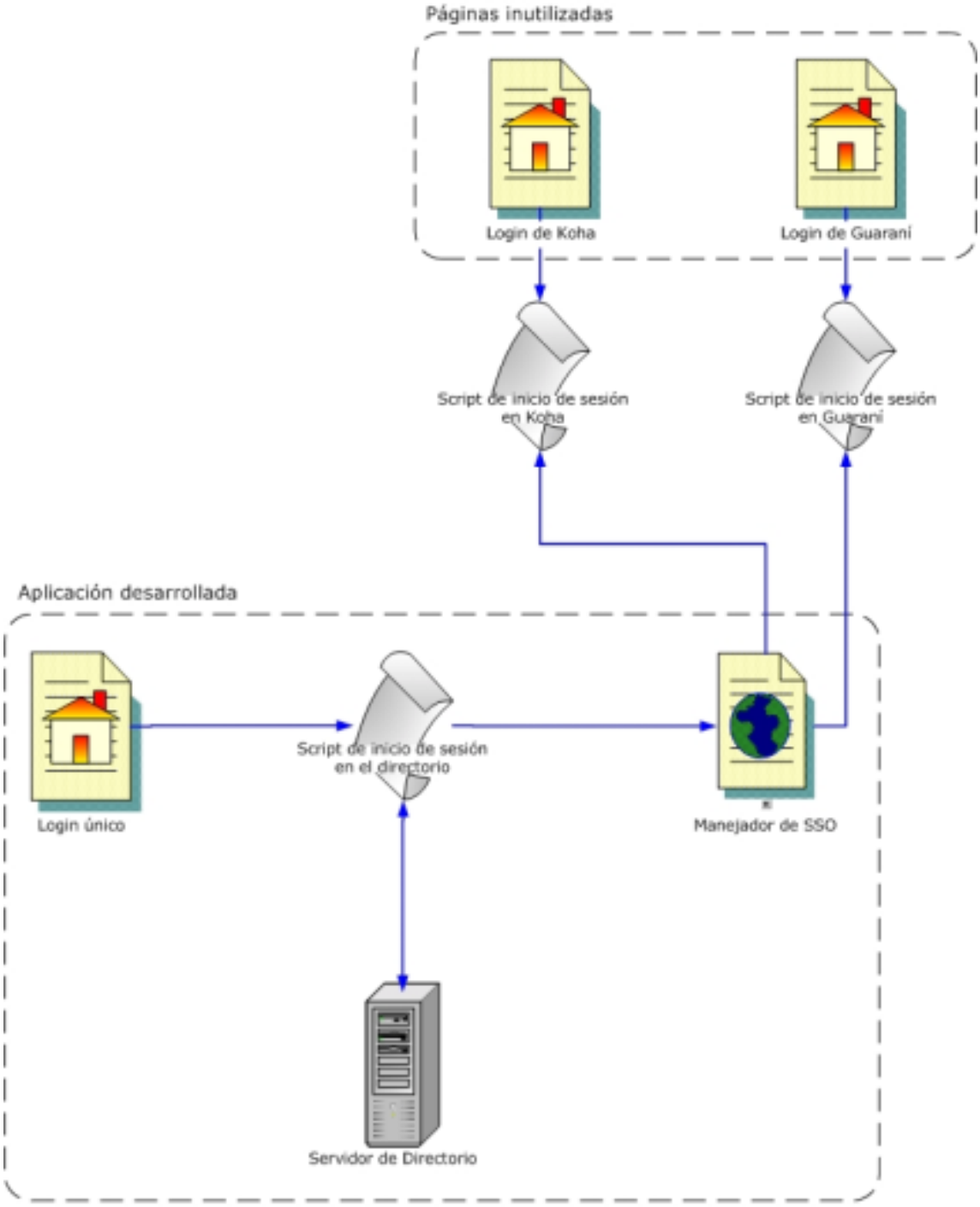

Figura 4.2

4.3.3 Acceso a una aplicación por demanda

Una vez que el usuario logró iniciar la sesión se abre una ventana extra del navegador. Una de ellas, la primera, se usa para cargar las aplicaciones a las que el usuario puede acceder. Mientras que la otra, la última en abrirse, se usa para poder cambiar entre las aplicaciones y hacer la finalización de sesión global.

Para armar esta segunda ventana, a la que llamaremos Manejador de SSO, se recuperan todos los hijos de ese usuario en el directorio. Dichos hijos contienen la

información de las aplicaciones a las que tiene acceso. Para recuperar los hijos se realiza una consulta en el directorio, pidiendo todas las entradas cuyo RDN sea uid=luciglesias,ou=linti,o=info. Luego se genera dinámicamente basado en esa información todos los enlaces a las aplicaciones. Por ejemplo, podemos ver como queda el código generado dinámicamente para el caso de Koha en la figura 4.3.

```
<form name="redireccionar" method="POST" action="http://opac-koha.linti.unlp.edu.ar/cgi-bin/koha/opac-user.pl"
target="VentanaPadre">
  <input type="hidden" name="userid" value="V99937959">
  <input type="hidden" name="password" value="Fr373mZz">
  <input type="submit" value="KOHA">
</form>
```

Datos recuperados del directorio

Figura 4.3

En este momento el usuario puede acceder a cualquiera de las aplicaciones sólo haciendo clic en el botón con el nombre de la aplicación (el botón de *submit* KOHA en la figura 4.3), que aparece en el Manejador de SSO. Dicho clic, lo único que hará es un *submit* a la página que hace el inicio de sesión de la aplicación a la que se quiere acceder. Se envía como parámetros a esta página el nombre de usuario y la contraseña (locales a esa aplicación) que se encuentran almacenados en el directorio, permitiendo así que el usuario inicie la sesión cuando lo demanda. Una vez accedida la aplicación, el usuario podrá trabajar con ella como si hubiera iniciado la sesión utilizando la cuenta local. Cuando lo deseé el usuario podrá cambiar la aplicación con la que trabaja, sólo haciendo clic en el botón con el nombre de la nueva aplicación con la que desea trabajar. Se repetirá el proceso antes mencionado para iniciar la sesión en la nueva aplicación. Cuando el usuario termina de trabajar debe cerrar la sesión desde el Manejador de SSO para que todas las aplicaciones que usó en el transcurso de esa sesión, finalicen sus sesiones locales. A continuación vemos como se implementa.

4.3.4 Finalización de sesión global

Para comprender como es el mecanismo de finalización de sesión global tenemos que tener en cuenta que cada aplicación al ser accedida, escribe cookies en la máquina del cliente con las que mantiene la sesión. Una vez que el usuario termina de trabajar con las aplicaciones, utiliza la opción de finalización de sesión global y se procede borrar todas las cookies.

Debemos tener en cuenta que sólo el sitio que escribe una cookie está autorizado a borrarla. Esto es así por una cuestión simplemente de seguridad, pues sino un sitio podría borrar las cookies de otro o modificarlas. Esta restricción presenta un problema para borrar la cookies porque en este esquema intervienen varios sitios (la aplicación encargada de hacer el SSO y las aplicaciones a las que accede) y no se puede borrar las cookies que otro sitio generó.

Por lo tanto, cada sitio se encarga de borrar las cookies que el mismo generó y de esta forma queda finalizada la sesión en cada aplicación participante. Por último, se borran las cookies de la aplicación de SSO. Una vez que esto es realizado podemos decir que se ha finalizado la sesión global.

Ahora, ¿cómo se puede lograr que cada sitio borre sus cookies?. Bueno, como se explicó al principio de este capítulo, entre los datos que se guardan de cada aplicación está el URL que se encarga de cerrar la sesión (es decir, de borrar las cookies).

Cuando el usuario inicia el proceso de finalización de sesión, se invoca secuencialmente a cada una de las aplicaciones para que cierren su sesión, mediante código javascript generado al momento del inicio de sesión. Luego, y por último, se borra la información del Manejador de SSO finalizando todo el proceso.

4.3.5 Alta, baja y modificación de usuarios

Entre las ventajas antes mencionadas de los sistemas de SSO se destacó el hecho de tener la posibilidad de hacer el ABM (altas, bajas y modificaciones) de un usuario de forma que sus datos (eventualmente distribuidos o replicados), sean modificados de forma transparente. O sea, que las modificaciones se realicen en un único punto y que todas las actualizaciones necesarias se hagan automáticamente. Este mecanismo garantiza la integridad de los datos de usuario.

Ahora, como la idea detrás de este sistema es la evitar la necesidad de modificar las aplicaciones que se integran, no es posible que el ABM en el directorio y en las aplicaciones se haga de forma transparente para el usuario y/o el administrador. Esto sucede porque cada aplicación maneja sus usuarios en una base de datos propia (lo que sería un directorio de aplicación específico, según lo explicado en el capítulo 3). Los usuarios son libres de modificar sus datos vía los módulos que cada aplicación provea. Por lo tanto, queda fuera del alcance del sistema la posibilidad de controlar esos cambios y automatizarlos para evitar inconsistencias. Por otro lado no todo es negativo, como veremos a continuación.

Para realizar el alta de un usuario se requiere que los usuarios sean creados previamente en cada aplicación antes de ser ingresados al directorio. Una vez realizado este trabajo aplicación por a aplicación, se puede dar de alta al usuario en el directorio para que éste pueda comenzar a trabajar.

Como ya se vio previamente, una de las ventajas de los sistemas de SSO es la de tener una sola contraseña para acceder a todos los recursos. Cuando se deben modificar el nombre de usuario y la contraseña sólo debe hacerse en el directorio. Por ende, reiniciar una contraseña es una tarea fácil para el administrador, pues sólo debe hacerlo en un lugar. El resto de los datos deben modificarse en todas las aplicaciones que tengan al dato en cuestión.

Con sólo eliminar la entrada en el directorio correspondiente a un usuario, éste pierde inmediatamente el acceso a todas las aplicaciones. Aunque eventualmente el usuario podría conocer las credenciales locales a cada aplicación. Una buena política sería la de no dar a conocerlas y, además, la de eliminar las cuentas locales en cada aplicación manualmente.

Como conclusión, podemos decir que la solución propuesta mantiene los fundamentos de los que sería un sistema de SSO. Pero tiene algunos inconvenientes heredados del hecho de no modificar las aplicaciones participantes en lo más mínimo.

Capítulo 5

Comparación de las soluciones analizadas

5.1 Ventajas y desventajas de cada solución

La comparación de soluciones de software es en general una tarea compleja para la cual no basta con una visión global del funcionamiento de dichas soluciones. Generalmente, y como en el caso de la comparación aquí realizada, hay muchas cuestiones que deben ser individualizadas y analizadas. En la siguiente sección se realiza una explicación de cuáles son los puntos que debemos tener en cuenta a la hora de elegir una solución SSO. Se ejemplifica además, con algunas características de las soluciones mencionadas en este trabajo, y se culmina con la realización de un cuadro comparativo, como resumen del capítulo.

Entre los aspectos que deben ser analizados al momento de evaluar un sistema SSO se encuentran, entre otros, el tipo de modelo (centralizado o federado), el tipo de aplicaciones soportadas (legacy y/o web), las plataformas que soporta, el costo de implantación y puesta a punto, la escalabilidad, la alta disponibilidad, las credenciales de acceso soportadas como tarjeta inteligente (*smart card*) o usuario/contraseña, la personalización y la facilidad de uso, el costo de administración, y los aspectos de seguridad que alcanzan.

Las cuatro soluciones descriptas en este trabajo de grado (Liberty Alliance, Kerberos, Passport y el modelo propuesto), son analizadas y comparadas basándose en el protocolo que utilizan y no en una implementación particular, con la excepción de Passport que no tiene una definición estándar sino que tiene una versión operativa desde la cual se infiere su funcionalidad. Pueden existir implementaciones que agreguen funcionalidad a las definiciones estándares. Cabe aclarar que para la comparación que se realizará a continuación se utilizan los aspectos mencionados en el párrafo anterior.

Tipo de modelo (centralizado o federado)

Lo primero que se puede contrastar en un sistema de SSO es si éste es centralizado o federado, como se mencionó en el capítulo 2. La característica común de todas las soluciones que están basadas en una arquitectura centralizada es la vulnerabilidad que ellas tienen en su punto central. El hecho de delegar todas las decisiones de autenticación a un sistema central, trae aparejado, el inconveniente de que un potencial ataque a esa entidad central dejaría inoperantes a todos los sitios y servicios que delegaron la autenticación en ella.

Un ejemplo, de arquitectura centralizada que padece la vulnerabilidad antes mencionada es el de Passport. El mismo ayudado por el historial de inseguridad de los sistemas de Microsoft, utiliza un protocolo propietario que permite a los usuarios iniciar sesiones en múltiples sitios autenticándose una única vez en un servidor común único. El problema acá es bastante obvio: hay un único punto de falla. Además, el servidor común es mantenido por Microsoft, lo cual requiere que el usuario confíe en la seguridad que éste le brinda.

En contrapartida, los sistemas federados, como Liberty Alliance, aventajan en este sentido a los centralizados. Un sistema federado construye la identidad de un usuario a partir de cuentas dispersas en varios servidores, por lo tanto no existe un único punto de ataque. De esta forma si alguno de los servidores falla, el usuario aún tendrá acceso a algunas de las aplicaciones para las que está autorizado.

Aplicaciones soportadas (legacy y/o web)

Tanto la solución aquí propuesta, como Passport y Liberty Alliance están orientadas al uso de aplicaciones web, que utilizan el protocolo HTTP. En cambio, Kerberos es

una solución más general y puede trabajar con aplicaciones legacy siempre que éstas puedan ser "kerberizadas".

Plataformas soportadas

Este punto hace referencia a la compatibilidad que tienen las distintas soluciones SSO con la infraestructura sobre la que se desea montar. No todas las soluciones SSO funcionan con todas las plataformas de servidores y plataformas de clientes. El grado de compatibilidad que tenga cada solución dependerá de cuánto ésta se apegue a los estándares.

En el caso de Passport, por ejemplo, la plataforma del cliente sólo debe contar con un navegador de Internet que respete los estándares y tenga habilitada la opción para la manipulación de cookies. Contrariamente, del lado de la plataforma del servidor que desea delegar la autenticación de sus usuarios, se debe contar con un módulo de software propietario (*Passport Manager*) que debe ser instalado junto con el IIS (*Internet Information Server*, servidor web de Microsoft) impidiendo la utilización de esta tecnología en plataformas no Windows.

Con Kerberos la situación es diferente. Las aplicaciones deben ser "kerberizadas", lo que significa que debe instalarse un módulo adicional de software que está desarrollado para las plataformas más utilizadas (como Linux, Unix y Windows). Kerberos, de esta forma, puede utilizar distintas plataformas dentro del mismo reino o dominio (generalmente una red de área local).

Para el caso de Liberty Alliance, la plataforma del cliente sólo debe contar con un navegador de Internet al igual que Passport. Tanto los proveedores de servicio como los proveedores de identidad, están definidos para la utilización de protocolos estándares (HTTP, SOAP, SAML) por lo que pueden ser implantados independientemente de la plataforma subyacente.

Para el caso del modelo propuesto en este trabajo, la tecnología usada por el cliente es sólo el navegador como en Passport y Liberty Alliance, mientras que del lado del servidor se requiere tanto de la aplicación web que permite realizar la autenticación, como del servidor LDAP que se usa como repositorio de los datos de usuario. Puede utilizarse cualquier servidor web (con soporte para php), así como cualquier servidor de directorio que respete el estándar LDAP, sobre cualquier sistema operativo.

Costo de implantación y puesta a punto

Este punto se refiere a las dificultades que hay que atravesar para la implantación de un sistema de SSO en una organización que no utiliza el mismo. En resumen, cuáles son las dificultades que aparecen con cada una de las soluciones para poner en funcionamiento el sistema de SSO. Aquí no se analiza el costo económico, ni el tiempo requerido, pues esto dependerá de muchos factores que este trabajo no contempla, como por ejemplo, el tamaño de la organización y los productos utilizados.

Cuando una aplicación Kerberos es agregada al sistema, el administrador debe realizar varias operaciones para dejarla funcional. El servidor "kerberizado" debe ser registrado en la base de datos, y se le debe asignar una clave privada. Esta tarea debe ser realizada para todos los servicios, además de poner operativo el KDC (servidor propio de Kerberos). Por lo tanto, esta tarea es proporcional al tamaño del reino que se quiere crear.

Desarrollar una aplicación de comercio electrónico utilizando Passport no es una tarea compleja. Para formar parte de la autenticación de Passport (además de pagar por el servicio) se requiere la creación de un conjunto de páginas dinámicas que interaccionan con el servidor de Passport permitiéndole hacer el inicio de sesión, el cierre de sesión y la recuperación de algunos datos del perfil del usuario. Un caso más complejo se da cuando un sitio (ya en producción) que utiliza otro mecanismo de autenticación decide delegar esta tarea al servidor de Passport. Aquí se presentan algunas complicaciones, como por ejemplo, migrar las cuentas ya existentes de la base de datos del sitio, a cuentas en el servidor de Passport, entre otras cosas.

En el caso de Liberty Alliance, el costo de implementar una solución de SSO primero requiere de conseguir a los socios de negocios necesarios para establecer un círculo de confianza. Luego, por lo menos uno de los socios dentro del círculo debe armar la infraestructura necesaria para poner operativo un proveedor de identidad (que desde el punto de vista técnico podría verse como un servidor encargado de autenticar a los usuarios, similar a Passport). Después, los sitios que formen parte del círculo, deben poder interactuar con el servidor de identidad para federar y autenticar a los usuarios. Cualquier sitio que desee pasar a formar parte de un círculo de confianza, puede integrarse en cualquier momento sin alterar la funcionalidad del resto de los participantes.

En el caso del modelo propuesto, la implantación requiere el uso de un servidor web, un servidor de directorio, la generación de las entradas apropiadas para el directorio, la instalación de todas las aplicaciones web a las que se desea acceder y la aplicación encargada de hacer el SSO. Una vez puesto en marcha, la instalación de una nueva aplicación sólo requiere la creación de una entrada en el directorio por cada usuario que tiene acceso a ella.

Escalabilidad

La escalabilidad es un punto crucial que se debe tener en cuenta al elegir cual es la mejor solución de SSO que se ajusta a una organización. Es necesario analizar cual es la escalabilidad de la solución aplicada a la organización, especialmente en organizaciones de gran envergadura, o en aquellas para las que se prevé un gran crecimiento.

Si hablamos de escalabilidad, Liberty Alliance lleva la delantera en este tema, porque permite, en contraste con cualquiera de los esquemas centralizados, que nuevos eslabones (los círculos de confianza) surjan, y se vayan relacionando con otros permitiendo un crecimiento perfectamente escalable basado en la transferencia de confianza de un eslabón a otro.

La escalabilidad de Kerberos se limita sólo a su reino y puede ser perfectamente aplicable en organizaciones que cuenten con una red de área local, o con varias, unidas por túneles formando una VPN (*Virtual Private Network*).

Tanto Passport, como la solución aquí *propuesta*, permiten la constante incorporación de nuevos sitios, que deleguen la tarea de realizar la autenticación. Una restricción, en cuanto a la escalabilidad, que tienen ambos es el tipo de aplicaciones con las que pueden interactuar, sólo web. Más, Passport sólo puede hacerlo con aplicaciones web desarrolladas para el IIS.

Alta disponibilidad

Por otro lado, la alta disponibilidad de un sistema de SSO es imprescindible, porque si el sistema está caído ninguno de los usuarios puede hacer uso de las aplicaciones

y/o servicios que necesita, para trabajar, comprar, etc. provocando pérdidas económicas. Por este motivo, la alta disponibilidad es crítica en cualquier sistema de SSO.

La alta disponibilidad está estrechamente relacionada con la seguridad de los sistemas informáticos. Si por ejemplo, en Kerberos, la máquina donde corre el KDC tiene vulnerabilidades que permiten que se realice un ataque y se comprometa su correcto funcionamiento, todo el sistema Kerberos está en riesgo.

Una forma habitual para mejorar la disponibilidad de este tipo de sistemas es manejar réplicas de los servicios críticos, que puedan alivianar la carga de los servidores primarios, así como también, servir de respaldo ante la eventual caída de un servidor. Cuando el servidor primario está fuera de servicio tiene que responder los requerimientos el servidor secundario de forma transparente para el cliente. Este tipo de soluciones debe mantener coherente la información del servidor primario y de las réplicas de forma automática, para que esto no implique la recuperación de información errónea.

Credenciales de acceso soportadas

La fuerza de los mecanismos utilizados para identificar a un usuario (como el uso de nombres de usuario y contraseñas), debe ser proporcional al valor o importancia del sistema o recurso que requiere protección.

El uso inapropiado de soluciones SSO puede debilitar significativamente la seguridad de una organización. Después de todo, si un intruso puede descubrir la contraseña única de un usuario, éste ganará acceso a todos los recursos de ese usuario. Para evitar que un intruso pueda ingresar al sistema haciéndole creer que es un usuario válido, se deben buscar mecanismos de identificación eficientes, sin hacer de estos una carga para el usuario. Algunas alternativas al uso de contraseñas son combinar la tecnología de SSO con tarjetas inteligentes, o tal vez con soporte biométrico (como la exploración de retina o la hulla digital), siempre y cuando la solución de software a implementar soporte estas tecnologías. Kerberos, por ejemplo, puede extender su funcionalidad para soportar tarjetas inteligentes y de esta forma reforzar uno de sus aspectos débiles.

Personalización y facilidad de uso

La personalización es un concepto que está relacionado con la facilidad de uso del esquema utilizado. Se refiere a como el sistema se adapta a las características particulares de un usuario. Por ejemplo, el sistema una vez que el usuario inició la sesión puede mostrarle sólo la lista de recursos a los que puede acceder, en lugar de todos. Esta es la política utilizada en el modelo propuesto, donde sólo se ven los vínculos a las aplicaciones que el usuario puede acceder. Passport utiliza siempre la misma simbología para indicar dónde hacer clic para iniciar la sesión y cerrarla, y de esta forma ayuda al usuario a darse cuenta rápidamente de cómo iniciar la sesión sin importar en que sitio de encuentre.

Costo de administración

Como los usuarios tendrán que recordar una sola contraseña en lugar de varias, la carga para el administrador se reducirá. Es más fácil para el administrador, mantener y proteger una única base de datos que varias. Cualquiera sea el modelo elegido, detrás tiene un repositorio de datos (un directorio, una base de datos) que guarda la información usada para validar los usuarios.

No todo se facilita para la administración. La instalación de nuevas aplicaciones requiere una cuidadosa configuración que les permita interactuar con el sistema de SSO.

Aspectos de seguridad

El tema de la seguridad informática es muy amplio. No es un objetivo de este trabajo analizar las vulnerabilidades de los distinto productos que permiten realizar SSO. Cada implementación particular de un esquema de SSO puede tener vulnerabilidades que son propias de dicha implementación o propias de la arquitectura.

5.2 Cuadro comparativo

Puntos de comparación	**Kerberos**	**Passport**	**Liberty Alliance**	**Propuesta de SSO**
Tipo de modelo	Centralizado	Centralizado	Federado	Centralizado
Escalabilidad y alcance	Su reino, generalmente una red de área local	Internet	Internet	Internet
Disponibilidad y tolerancia a fallos	Existe un KDC maestro y permite réplicas que periódicamente se actualizan de la BD del maestro	A cargo de una empresa	Permite la existencia de más de un Proveedor de Identidad por círculo de confianza	Permite la réplica del directorio
Plataformas soportadas por el servidor	Soportada por Servidores Windows, Unix, Linux	Soportada por Servidores Windows con IIS	Existen algunas implementa-ciones como SourceID que son multi-plataforma (están desarrolladas en Java y .Net)	Necesita de un servidor web con soporte para php y un servidor de directorio LDAP
Plataformas soportadas por el cliente	Cualquier aplicación "kerberizable"	Requiere un navegador de Internet	Requiere un navegador de Internet o algún dispositivo que soporte HTTP y los protocolos superiores	Requiere un navegador de Internet
Implantación	Requiere la instalación de servicios adicionales y la "kerberización" de las aplicaciones	Requiere adaptar algunos scripts (código asp) y definir la forma en que se migrarán los usuarios existentes (si los hubiese)	Requiere de la instalación y configuración de los Proveedores de Identidad, así como también los módulos de software que permiten la federación de identidades	Requiere la instalación y configuración de los servidores, y la carga de datos en el directorio

Autorización	El estándar no define como realizarla. Existen algunas implementa-ciones donde se delega la tarea a los servidores objetivo, mientras que otras extienden su funcionalidad para realizarla en el KDC	Cada sitio participante debe encargarse de autorizar o no el acceso de un usuario	Se maneja con la federación de las identidades a cada Proveedor de Servicio	Lo maneja con información en el directorio
Mecanismos de autenticación aceptados	Nombre de usuario y contraseña, tarjeta inteligente o huella digital	Nombre de usuario y contraseña (con la posibilidad de requerir un segundo código)	Nombre de usuario y contraseña, tarjeta inteligente o huella digital	Nombre de usuario y contraseña

Capítulo 6

Conclusiones

Al principio de este trabajo se analizaron cuales son las dificultades que padece una organización que no hace uso de una solución SSO. Posteriormente, se realizó un análisis de varias de las soluciones existentes hoy en día (Kerberos, Passport y Liberty Alliance) que se clasificaron como centralizadas y federadas.

Luego, se analizó al servidor de directorio como una herramienta estándar para el almacenamiento de información en general y particularmente de usuarios de una red. A partir de ello se definió un esquema propio de SSO, que utiliza un directorio como repositorio de datos, y se implementó a modo de prueba para verificar su funcionamiento, utilizando dos aplicaciones web reales para estudiar la integración.

En el curso de esta implementación, así como en el análisis de las herramientas existentes se obtuvo un mejor entendimiento de las arquitecturas SSO. Se concluyó además, en un balance general, que las soluciones federadas son mejores que las centralizadas pues no tiene un único blanco de ataque y además, la identidad de un usuario se forma a partir de fragmentos de identidad previamente existentes que se van relacionando. En otras palabras, la solución federada, parece adaptarse mejor a las características actuales de Internet. Aunque en general la implementación de una solución federada es más compleja que la de una centralizada.

También se puede decir que cada una de las soluciones vistas se adapta mejor a un escenario que a otro. Cada implantación de un sistema de SSO requiere un cuidadoso análisis de cuáles son las necesidades de los usuarios y cuáles son los servicios que se necesitan acceder.

La solución presentada en este trabajo puede utilizarse, por ejemplo, como el módulo de autenticación en un portal de Internet que puede llegar a hacer uso de otras aplicaciones. Por ejemplo, en el caso concreto de Koha y Guaraní, sería un conjunto de servicios que se le podrían ofrecer a alumnos de una facultad, cuando ingresan al portal de la misma.

Entre las ventajas de la solución desarrollada para este trabajo se encuentran, la facilidad de integración con nuevas aplicaciones, la posibilidad de incorporarse a un directorio corporativo previamente existente con sólo algunas extensiones, la facilidad de uso para el usuario final pues el inicio de sesión es similar el de la mayoría de los sitios actuales de Internet y, la transparencia que todo mecanismo de SSO debe ofrecer. Sin embargo, existen algunos aspectos que pueden ser mejorados tales como la utilización del Servidor de Directorio más cercano, en lugar de un único servidor centralizado, la utilización del mismo directorio para enriquecer el perfil del usuario, la posibilidad de incorporar aplicaciones legacy, etc.

Existe una tendencia de Microsoft a complementar Passport con otros proveedores de identidad para conectar sus mecanismos de autorización y autenticación a una infraestructura federada. De forma similar al Proyecto Liberty Alliance, Microsoft ya anunció una nueva tecnología llamada *TrustBridge*.

Como el Proyecto Liberty Allience es todavía nuevo, si tendrá un resultado exitoso o no es una incógnita. La franqueza de este sistema insinúa una autenticación segura y fiable a través de plataformas de software y hardware, inclusive de dispositivos móviles.

La reflexión final es que no hay una definición muy marcada hacia un producto. Si bien, Passport ha tenido gran aceptación en la Internet de hoy, parece existir una tendencia hacia las identidades federadas.

Glosario

3DES (Triple Data Encoding Standard): Es un algoritmo de encriptación con clave simétrica.

ACL (Access Control Lists): Es un método que define en un sistema de computación quién tiene acceso a qué. Tiene la capacidad de otorgar o limitar el acceso a un recurso por parte de un usuario.

API (Application Program Interface): Es un conjunto de funciones y procedimientos que los programadores pueden utilizar en sus propios programas, para acceder a los diferentes recursos de un sistema.

Aplicación legacy: Es una aplicación en la que una empresa hizo un gran gasto de tiempo o dinero. También se conoce de esta forma a las componentes de la infraestructura que puede tener una empresa u organización que está entrando en desuso.

Aplicación web: Es una aplicación que corre sobre Internet, donde el usuario accede a la lógica del negocio a través de un navegador.

Círculo de confianza: Un grupo de servicios y proveedores de identidad que tiene relaciones de negocio construidas sobre las bases de la tecnología de Liberty Alliance.

Cookie: Una cookie es un pequeño archivo de texto que los servidores de sitios web almacenan en el navegador del cliente. Cada vez que el cliente hace un requerimiento de una página a un sitio, el navegador envía la cookie al servidor. La cookie contiene información provista por el usuario para el sitio, típicamente durante la registración, que identifica al cliente ante el sitio para personalizar las páginas que ve (por ejemplo, mostrando el nombre del cliente en la página).

DES (Data Encoding Standard): Es un algoritmo de encriptación con clave simétrica.

DIT (Directory Information Tree): Se conoce así a la estructura con forma de árbol de un servicio de directorio.

DN (Distinguished Name): Es un nombre único que identifica en forma no ambigua, una entrada en un servicio directorio.

FTP (File Transfer Protocol): Es un protocolo de capa de aplicación que sirve para transferir archivos de una máquina a otra.

GET: Método usado para enviar datos codificados dentro de una URL.

Guaraní: Sistema para la gestión de alumnos en unidades académicas. Más información puede ser encontrada en el sitio http://www.siu.edu.ar/.

Hash: Es un número generado a partir de una cadena de texto. El hash es por lo general substancialmente más corto que el texto en sí mismo, y es generado por una fórmula, de forma tal que es extremadamente difícil que alguna otra cadena de texto produzca el mismo valor. El propósito del uso de funciones de hash puede ser, por ejemplo, la verificación de contraseñas. Md5 es un ejemplo de un algoritmo de hash.

HTTP (Hypertext Transfer Protocol): Es un protocolo de capa de aplicación que se utiliza principalmente para la transferencia de hipertexto.

HTTPS (Hypertext Transfer Protocol over Secure Socket Layer): Es un protocolo de capa de aplicación que encripta y desencripta los datos transmitidos entre el navegador del cliente y el servidor web usando SSL.

IDEA (International Data Encryption Algorithm): Es un algoritmo de encriptación con clave simétrica.

Identidad de red: Es el conjunto global de atributos compuesto de varias cuentas de usuario individuales.

Identidad federada: Es un sistema que relaciona múltiples cuentas de usuario que pertenecen a una única persona física.

IIS (Internet Information Server): Servidor web de Microsoft.

Informix: Es un servidor de bases de datos relacionales.

IP (Internet Protocol): Protocolo estándar que define a los datagramas IP como unidad de información que pasa a través de una red de redes y proporciona las bases para el servicio de entrega de paquetes sin conexión y con el mejor esfuerzo. El conjunto de protocolos completo se conoce frecuentemente como TCP/IP pues el TCP y el IP son los dos protocolos más importantes.

JAVA: Lenguaje de programación multiplataforma.

JavaScript: Es un lenguaje de scripting que puede interactuar con código HTML para hacer a las páginas web más robustas y dinámicas agregandoles lógica, a la aplicación, del lado del cliente. Casi todos los navegadores actuales soportan JavaScript.

KDC (Key Distribution Center): Es el servidor de Kerberos en sí mismo.

Koha: Es un sistema open source para la administración de una biblioteca. Más información puede ser encontrada en el sitio http://www.koha.org/.

LDAP (Lightweight Directory Access Protocol): Es un protocolo de capa de aplicación y a su vez es una arquitectura estándar para organizar los datos en un directorio. LDAP es una versión simplificada de DAP (Directory Access Protocol) usado en el estándar anterior X.500.

LDIF (LDAP Data Interchange Format): Es un estándar que representa las entradas de un directorio LDAP en formato textual.

MD5 (Message Digest 5): Es un algoritmo de firma digital. Es usado para verificar la integridad de los datos a través de la creación de un mensaje de 128 bits, derivado de los datos de entrada. A partir de una cierta entrada siempre retorna lo mismo. Es extremadamente difícil encontrar dos bloques que deriven en exactamente el mismo mensaje.

MySql: Es un servidor de bases de datos relacionales.

Open Source: Hace referencia a los programas de computación que están distribuidos con una licencia que obliga a adjuntar el código fuente. Su definición es mucho más amplia y puede ser encontrada en http://www.opensource.org/.

OSI (Open System Interconnection): Es un modelo de referencia sobre como los mensajes deben ser transmitidos entre los distintos integrantes de una red de telecomunicaciones.

Passport Manager: Módulo de software adicional para el IIS, que permite hacer la autenticación de usuarios a través de Passport.

PHP: Lenguaje de scripting para el desarrollo de páginas web dinámicas.

POST: Método usado para enviar datos codificados dentro del cuerpo de un mensaje HTTP.

Principal: Usuario o servicio que se puede autentificar mediante el uso de Kerberos. Todos los principales de un dominio tienen su propia clave, que se deriva de su contraseña, para el caso de los usuarios, o se genera aleatoriamente, para el caso de los servicios.

Proveedor de identidad: Provee autenticación de usuarios en un círculo de confianza. Un círculo de confianza puede tener múltiples proveedores, autenticando diferentes grupos de usuarios. Esta terminología es usada principalmente en el estándar de Liberty Alliance.

Proveedor de servicio: En una arquitectura federada es un servicio que hace uso de los proveedores de identidad para hacer la autenticación de usuarios y tomar las decisiones de control de acceso. Esta terminología es principalmente usada en el estándar de Liberty Alliance.

PUID (Passport Unique Identifier): Identificador de usuario único generado por Passport.

RC4 (Rivest Cipher 4): Es un algoritmo de encriptación con clave simétrica.

RPC (Remote Procedure Call): Es un modelo de programación en red que permite comunicar aplicaciones punto a punto. El proceso llamador hace un requerimiento en forma de procedimiento, función o método, que es ejecutado en la máquina remota.

SAML (Security Assertion Markup Language): Es el estándar más popular de identidades federadas con una creciente aceptación. Especifica los mecanismos para la autenticación de usuarios en un sistema federado, el intercambio de atributos y las decisiones de autorización sobre esos usuarios.

SOAP (Simple Object Access Protocol): Es un protocolo liviano para el intercambio de información en un ambiente descentralizado y distribuido. Es un protocolo basado en XML que permite hacer llamados a procedimientos remotos (RPC).

SQL (Structured Query Language): Es un lenguaje de manipulación de datos y definición de estructuras de datos estándar, para el manejo de bases de datos relacionales.

SSL (Secure Sockets Layer): Es un protocolo seguro que provee encriptación de datos, autenticación del servidor, e integridad de los mensajes. Casi todos los navegadores actuales soportan SSL y muchos sitios web y servicios (incluyendo a Passport) usan este protocolo cuando transmiten y reciben información confidencial de usuarios (por ejemplo, contraseñas o números de tarjeta de crédito). Por

convención, las direcciones de páginas web que requieren usar una conexión SSL comienzan con HTTPS en lugar de HTTP.

TCP/IP (Internet Protocol Suite): Nombre oficial de los protocolos TCP/IP.

TGS (Ticket Granting Service): Es el encargado de emitir boletos para un servidor específico cuando el cliente quiere accederlo, en el contexto de Kerberos.

TGT (Ticket Granting Ticket): Boleto especial que permite al cliente obtener boletos adicionales para aplicaciones de servidor específicas, en el contexto de Kerberos.

Ticket: Es un boleto o credencial electrónica, en el contexto de Kerberos.

Troyano: Programas que, enmascarados de alguna forma como un juego o similar, buscan hacer creer al usuario que son inofensivos, para realizar acciones maliciosas en su equipo. Estos troyanos no son virus ni gusanos dado que no tienen capacidad para replicarse por si mismos, pero en muchos casos, los virus y gusanos liberan troyanos en los sistemas que infectan para que cumplan funciones específicas, como, por ejemplo, capturar todo lo que el usuario ingresa por teclado. La principal utilización de los troyanos es para obtener acceso remoto a un sistema infectado a través de una puerta trasera. Este tipo de troyano es conocido como Backdoor.

URI (Uniform Resource Identifier): Es un término genérico para todos los tipos de nombres y direcciones que hacen referencias a objetos en la World Wide Web. Un URL es un tipo de URI.

URL (Uniform Resource Locator): Cadena que proporciona la localización de una parte de información. La cadena comienza con el protocolo (por ejemplo, HTTP) seguido por la identificación de la información específica (por ejemplo, el nombre de dominio de un servidor y el camino hacia un archivo en el servidor).

VPN (Virtual Private Network): Es una red privada conectada por túneles. Transporta tráfico perteneciente a una red privada desde un nodo final hasta otro ubicado en otra red privada remota sobre una red pública (como por ejemplo, Internet).

X.500: Es un método estándar para el desarrollo de un servicio de directorio. Es la base para la definición de LDAP.

XML (Extensible Markup Language): Es un lenguaje para la definición de otros lenguajes. Utiliza archivos de texto plano y es autodescriptivo.

Referencias

[Abdul-Rahman 1997] A. Abdul-Rahman and S. Hailes. A distributed trust model. In Proceedings of the New Security Paradigms 97, 1997.

[Abdul-Rahman 2000] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In Proceedings of the Hawaii Int. Conference on System Sciences 33, Maui,Hawaii, 2000.

[Barlen 2002] Thomas Barlen, Wolfgang Eckert, John Taylor, Klaus Tebbe, Wendy Thomson, Marc Willems. Implementation and Practical Use of LDAP, 2002. (http://www.redbooks.ibm.com/).

[Beatty 2002] J.D. Beatty et al. Liberty Protocols and Schemas Specification 1.0. Liberty Alliance, 2002.

[Box 2000] D. Box et al. Simple Object Access Protocol (SOAP) 1.1. World Wide Web Consortium Note, May 2000.

[Brown 1999] A. Brown et al. SOAP Security Extensions: Digital Signature. World Wide Web Consortium Note, February 2001.

[Comer 1996] Douglas E. Comer. Redes globales de información con Internet y TCP/IP, 1996.

[Evaluation 1999] International Standardization Organisation (ISO). Evaluation criteria for IT security (ISO/IEC 15408:1999), 1999.

[Hodges 2002] J. Hodges et al. Liberty Architecture Overview 1.0. Liberty Alliance, 2002.

[Jakarta 2002] The Apache Software Foundation. The Jakarta Project-Apache Tomcat,2002. (http://jakarta.apache.org/tomcat/index.html).

[Johner 1998] Heinz Johner, Larry Brown, Franz-Stefan Hinner, Wolfgang Reis, Johan Westman. Understanding LDAP, 1998. (http://www.redbooks.ibm.com/).

[Kannappan 2002] L. Kannappan et al. Liberty Architecture Implementation Guidelines 1.0. Liberty Alliance, 2002.

[Kohl 1993] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5), 1993.

[Kormann 2000] D. Kormann and A. Rubin. Risk of the passport single signon protocol. In Computer Networks, Elsevier Science Press, volume 33, 2000.

[Maler 2002] E. Maler, P. Hallam-Baker, et al. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) 1.0. OASIS, May 2002.

[Microsoft Dev 2002] Microsoft Corporation. Microsoft Passport Software Development Kit Documentation, 2002. (http://www.dcs.napier.ac.uk/~bill/PROJECTS/paul.pdf)

[MicrosoftTO 2001] Microsoft Corporation. Microsoft .NET Passport – Technical Overview, 2001. (http://www.student.chula.ac.th/~43322080/wp_engl_net_passport.pdf)

[Mishra Bindings 2002] P. Mishra et al. Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) 1.0. OASIS, May 2002.

[Mishra SOAP 2002] P. Mishra et al. The SOAP Profile of the OASIS Security Assertion Markup Language (SAML) 1.0. OASIS, March 2002.

[Open Group 2002] Open Group. Introduction to Single Sign-On, 2002. (http://www.opengroup.org/security/sso/).

[Rouault 2002] J. Rouault et al. Liberty Bindings and Profiles Specification 1.0. Liberty Alliance, 2002.

[Slemko 2001] M. Slemko. Microsoft Passport to Trouble, 2001. (http://alive.znep.com/marcs/passport/).

[Stallings 2002] William Stallings. Cryptography and network security, 2002.

[Steiner 1988]	J. Steiner, C. Neuman, and J. Schiller. Kerberos: An authentication service for open networks systems. In Usenix Conference Proceedings, 1988.
[Tung 1996]	B. Tung. The Moron's Guide to Kerberos, 1996. (http://www.isi.edu/gost/brian/security/kerberos.html).
[Www Koha]	Koha. (http://www.koha.org/).
[Www KohaSiu]	Implementación del Software Koha en la Argentina. (http://koha.siu.edu.ar/)
[Www Liberty]	Liberty Alliance Project. (http://www.projectliberty.org/)
[Www Open]	OpenLdap. (http://www.openldap.org/)
[Www Security]	Security. (http://www-912.ibm.com/as400/v5r2to/pdfs/j02secur.pdf).
[Www SIU]	Sistema de Información Universitaria. (http://www.siu.edu.ar/).
[XML 2002]	The Apache Software Foundation. XML Security, 2002. (http://xml.apache.org/security/index.html).