UNIVERSIDAD DE ALCALÁ

Escuela Técnica Superior de Ingeniería Informática

Ingeniería Informática

Trabajo Fin de Carrera

Integración de un metasistema de identidad en la arquitectura eduroam para proporcionar un servicio de inicio de sesión único unificado

Samuel Muñoz Hidalgo

2010

UNIVERSIDAD DE ALCALÁ

Escuela Técnica Superior de Ingeniería Informática

Ingeniería Informática

Trabajo Fin de Carrera

Integración de un metasistema de identidad en la arquitectura eduroam para proporcionar un servicio de inicio de sesión único unificado

Autor: Samuel Muñoz Hidalgo

Director: D. Enrique de la Hoz de la Hoz

TRIBUNAL:

Presidente: D. Antonio García Herráiz

Vocal 1º: D. Bernardo Alarcos Alcázar

Vocal 2º: D. Enrique de la Hoz de la Hoz

CALIFICACIÓN: .....

FECHA: .....

Quiero dedicar este trabajo principalmente a mi familia, sobre todo a mi madre por hacerme la comida y la cena todos los días, ya que sin energía no funciona el cuerpo.

También quiero agradecer la colaboración de mi tutor, Enrique, siempre dispuesto a responder a mi desvaríos; de mis amigos ("bboys" incluídos) por evitar que esos desvaríos fueran a más; y dar especial mención a todos los que hacen "softwware" libre por razones más que obvias.

Lo que sobre va ponderadamente dedicado al resto del mundo según el número de veces que me preguntó por el estado del proyecto. Los días malos cuentan doble.

The more is given the less the people will work for themselves,
and the less they work the more their poverty will increase.

Leo Tolstoy - Help for the Starving, Pt. III (January 1892)

A quien dices tu secreto das tu libertad.

Fernando de Rojas - La Celestina

If liberty means anything at all, it means the right to tell people
what they do not want to hear.

George Orwell - Preface to Animal Farm (1945)

Work consists of whatever a body is OBLIGED to do, and...
Play consists of whatever a body is not obliged to do.

Mark Twain - The Adventures of Tom Sawyer (1876)

Sí señor, desgraciadamente soy una persona maleducada.
No soy como usted. ¡Haga el favor de dejarme en paz!
¡Déjeme en paz! ¡Pues déjeme de admirar!
¡Váyase usted a la mierda, he dicho a la mierda!

Fernando Fernán Gómez - A un lector ...

Índice general

Índice de figuras

Resumen

Este proyecto de final de carrera versa sobre cómo se podrían combinar un metasistema de identidad basado en la tecnología de "Information Cards" y la red universitaria "eduroam" para ofrecer una solución conjunta al manejo de la identidad digital, la identidad centrada en el usuario y la federación de servicios en el marco de la educación superior.

Además de las herramientas propias para tales efectos, se desarrollará piezas de código y modificaciones para extender sus funcionalidades. Se parte de la base actual ya desarrollada (simpleSAMLphp) sobre inicio de sesión único (SSO) para extenderlo y ofrecer un inicio de sesión único unificado (uSSO).

Palabras clave: uSSO, SSO, metasistema, identidad digital, identidad centrada en el usuario, eduroam, simpleSAMLphp, infocard, federación de servicios, RADIUS, suplicante, servicios Web, seguridad, autenticación.

Parte I

El problema

Capítulo 1

El manejo de la identidad

NOTA: Este capítulo es una síntesis del primer capítulo del libro "Understanding Windows CardSpace: An Introduction to the Concepts and Challenges of Digital Identities" [Bertocci et˜al., 2008] que encarecidamente recomiendo leer si se desea profundizar en aspectos tanto técnicos como sociales y de mercado en el tema de las Information Cards. También he aportado experiencias propias en forma de anécdotas para afianzar conceptos.

1.1. Internet VS Realidad

Me gustaría situar el contexto hace unos años, en ese momento en el que se estaba dando el paso hacia la interconectividad globlal, justo cuando Internet empezó a ser una red de masas. Por aquel entonces, las empresas empezaban a ofrecer sus recursos en la red.

Pero... ¿qué es un recurso? Según la R.A.E. un **recurso** es el conjunto de elementos disponibles para resolver una necesidad o llevar a cabo una empresa. En este caso, datos, información almacenada.

La forma en que las empresas permiten acceder a sus recursos es a través de **servicios**, que son herramientas que nos permiten interactuar con la información almacenada, esto es: añadirla, borrarla, modificarla o consultarla. Ejemplos de servicios son el correo electrónico, la banca online, redes sociales, páginas de búsqueda de trabajo, hacienda pública, etc. Asusta pensar que gran parte de nuestro patrimonio económico puede estar representada por no más que un número en algún ordenador cuya localización desconocemos, por esto es de vital importancia proteger adecuadamente esta información y garantizar el servicio a esta a su legítimo propietario. Empieza a tener sentido el dicho de que "vivimos en la era de la información y la información es poder".

Obviamente las personas que hacen uso de los servicios son seres que existen en un mundo físico, luego es lógico pensar que tendrán una representación abstracta en el mundo digital. Esta representación se llama **identidad digital**. Sirve para varios menesteres, pero ahora mismo nos centraremos en el de la autenticación, garantizar que alguien es quien dice ser, establecer la correlación entre una persona y su identidad digital. Podemos ver esta analogía resumida en la figura 1.1

Debido a esta masificación de la adopción de servicios en Internet, surgen nuevos modelos de negocios y toma cierto protagonismo un concepto, la ubicuidad. La ubicuidad es un término utópico que viene a significar omnipresencia. Se puede resumir en que no importa donde estemos, podremos acceder a nuestros datos desde cualquier ordenador, refinando el término podría agregar que hasta

podríamos trabajar en un entorno personalizado con las herramientas a las que estamos acostumbrados. Es la deslocalización de la información. Esto tiene mucho que ver con la confianza en terceras partes, ya que delegamos la responsabilidad de almacenar nuestros datos en empresas que nos garantizan el acceso a ellos en la mayoría de condiciones. Esperamos que los guarden celosamente, ya que ante una exposición de estos al mundo, su reputación empeoraría, la confianza de sus usuarios disminuiría y el negocio quebraría.

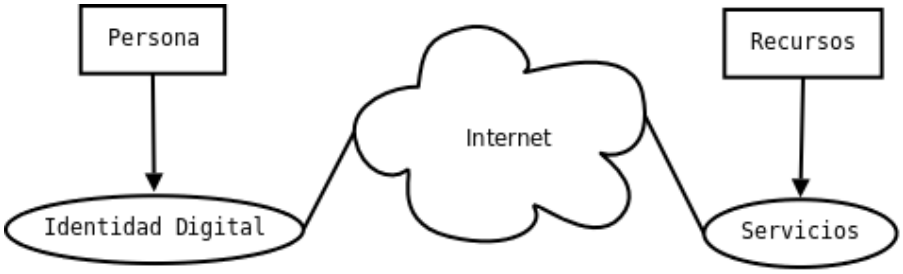

Figura 1.1: Abstracción del mundo real a Internet

1.2. Criminales

La breve introducción anterior nos lleva a la siguiente consideración. Internet es la proyección del mundo real, hay negocios, personas y... delincuentes. En el mundo físico, reconocer a una persona es más sencillo, tiene una cara, un ADN, documentos difíciles de falsificar emitidos por su gobierno, etc. En la red, el problema de la autenticación radica en la falsificación de los credenciales de esa identidad o incluso en el robo de la identidad digital, algo que en la realidad se asemejaría a poseer a alguien (vudú, hipnotismo) o cambiar la apariencia (estilo Mística de los X-Men) o cualquier otro poder digno de novela de ciencia-ficción. No estamos hablando de garantizarle a alguien el acceder a una máquina (era pre-conectividad), sino de asegurar su identidad ante un sinfín de servicios ofrecidos por un gran grupo de empresas que poco tienen que ver entre sí, ni métodos de autenticación, ni políticas de seguridad o de datos, ni siquiera podemos asegurar que son quien dicen ser (Internet es una red neutral bastante anárquica y que tiende a la pluralidad).

Es precisamente por esta debilidad en la seguridad que los ciberdelincuentes tienen maneras de hacer dinero de forma rápida. Veamos una una tabla en la figura 1.2 en la que se explica el cambio de delitos que ha habido en el crimen digital con la conectivad.

NOTA: *el precursor más conocido de la ingeniería social fue Kevin Mitnick (Cóndor) en la primera mitad de los 90, pero no se puede considerar un método propio de esa época en compraración a la cantidad de crackers que se dedicaban a saltar protecciones de software. La ingeniería social se puede definir como el arte de manipular a las personas para conseguir información (un concepto atemporal aplicado al campo de la seguridad informática).*

	ERA PREINTERNET	ERA INTERNET
Cambio de roles de los atacantes	Hacker,Cracker,Script Kiddie	Spammer, estafador, timador
Cambio de botines	Programas pirateados (venta), reconocimiento social	Cuentas de usuario, poder de computación, ancho de banda, información sensible, DINERO
Métodos	Vulnerabilidades en el código, escalada de privilegios, niveles muy técnicos	Ingeniería social, engaños, mucha automatización, rapidez, SUPLANTACIÓN DE IDENTIDAD

Figura 1.2: Progresión del cibercrimen

Los principios de la **ingeniería social** son [Wikipedia, g]:

1. Todos queremos ayudar.
2. El primer movimiento es siempre de confianza hacia el otro.
3. No nos gusta decir NO.
4. A todos nos gusta que nos alaben.

De todos los métodos delictivos actuales, el que atañe al usuario de forma más directa es el de la suplantación de identidad (phishing), alguien se hace con los credenciales que desbloquean una identidad digital y hace un uso para beneficio propio de los recursos a los que tiene acceso. El caso más común es el robo de dinero a través de la banca digital. Por no decir siempre, el culpable de que este ataque tenga éxito, suele ser el propio usuario, ya que expone de forma explícita sus credenciales ante cualquiera que simule ser alguien en quien tiene confianza (un servicio).

1.2.1. El ciclo de la información

Para saber cómo nos pueden atacar, tenemos que comprender el ciclo de la información, desde que se crean los datos hasta que dejan de ser utiles o se destruyen. Pasan por las siguientes fases: introducción, transporte, procesado y almacenaje.

Fase de introducción

Es la fase en la que el usuario introduce los datos en la aplicación y estos todavía no han sido enviados. Es la fase favorita de los atacantes, ya que produce los mayores resultados y más exitosos. Existen dos tipos de ataques:

- Grabar o monitorizar al usuario sin que este lo sepa. Forma pasiva. Esto se puede hacer desde con programas espía estilo keylogger (almacenador de pulsaciones) hasta con métodos menos informáticos como puede ser con cámaras o mirando diréctamente cómo lo escribe (como los timos de los cajeros).
- Engañar al usuario para que introduzca la información en un sitio incorrecto (controlado por el atacante). Es quizá la forma más usual de robo de información ya que se basa mucho en la debilidad humana y poco en la técnica (como la mayoría de los timos).

> **NOTA:** *El primer contacto que tuve con este tipo de técnicas, mucho antes de conocer la denominación "phishing"fue como "víctima"de un ejecutable que simulaba ser la ventana de inicio de sesión del popular Messenger de Microsoft, obviamente y debido a que en cada versión cambiaban el diseño de esta ventana, no piqué en la trampa. Pero esta experiencia me sirvió para darme cuenta en lo refinado que tiene que ser el cebo que se pone ante el usuario para que no perciba la inconsistencia.*

Las tres fases de un ataque de robo de identidad mediante phishing son:

1. Nos llega un mensaje (a través de e-mail, mensajería instantánea, teléfono, SMS o como sea) de alguien de çonfianza"(¡no lo es!) que capta nuestra atención. Probablemente trate un tema de actualización de datos (si viene del banco) o de ver alguna foto (algún conocido). Todo es un pretexto para derribar la primera barrera, la desconfianza en el mensajero.

2. Hay un enlace que supuestamente nos permite hacer lo que dice el mensaje. Lo seguimos y llegamos a una web.

3. Esta web simula ser la entrada al servicio que queremos usar. Una vez introducidos los credenciales, llegan al atacante.

Está claro que el primer mensaje no venía sino de alguien que se hacía pasar por otro y el enlace era a una web falsa que nada tiene que ver con la entidad en la que confiamos.
Al igual que el SPAM, este tipo de ataque suelen ser indiscriminado, se mandan mensajes aleatoriamente a infinidad de destinatarios, y si pican, pican. No obstante se pueden refinar muchísimo. Si yo sé la relación contractual entre una empresa y una persona (más información), puedo refinar la interfaz, el mensaje, es decir, personalizar todo para que la experiencia del usuario sea casi igual a veces anteriores y aumentar mis posibilidades de éxito.

Fase de transporte

Es la fase en la que los datos van del emisor al receptor. Es una fase difícil de controlar tanto para atacante como para los comunicadores ya que Internet se caracteriza por tener un modelo de conexión un tanto anárquico en el que no hay caminos fijos y los nodos por los que pasa la información pueden pertenecer a cualquiera (multitud de intermediarios). Un ataque en esta fase se conoce como el ataque del intermediario (man in the middle) y consiste en que prácticamente cualquier nodo (computadora) por el que pase la información es susceptible de analizarla y extraer los datos de interés. Es más, no hace falta que haya intermediarios, con que alguien se conecte (pinche) el medio de transmisión, ya tendrá acceso a la información.
La solución en esta fase es la criptografía, es decir, que el emisor mande un mensaje que sólo sea inteligible para el receptor, así aunque sea interceptado, la información no tendrá lógica alguna y parecerán datos aleatorios.

En algunos ambientes militares, para evitar pinchazos en sus líneas, las meten dentro de conductos presurizados, de forma que cualquier intento de acceder al cable, libera gas y hace saltar una alarma.

[Tanenbaum, 2003]

NOTA: *Como alternativa a la criptografía tradicional, existe la cuántica (basada en la mecánica cuántica). Se basa en el principio de incertidumbre de Heissemberg para en la práctica decir que no es posible analizar/estudiar una transmisión sin alterarla (algo reservado exclusivamente al receptor). Es un buen método para un intercambio de claves seguro, aunque su fuerza reside en la física en vez de en la lógica.*

NOTA: *Cuando alguien me pregunta sobre la seguridad WiFi, le respondo que es como si un caco tuviera una oreja el patio de luces de un bloque de apartamentos. Una WiFi sin proteger transmite en claro, como si las vecinas se pusieran a chismorrear, con cifrado WEP el simil es con vecinas hablando en lenguas romances (por la similitud a mi lengua materna el castellano), con WPA estaríamos tratando de un lenguaje desconocido. Aun cuando no podamos sacar algo en claro de la comunicación, siempre podemos sospechar por qué una vecina lleva casi una semana sin chillar. ¿Será buen momento para entrar en su casa porque se ha ido de vacaciones? Quizá el dar voces no sea la forma más segura de comunicarse, aunque puede que la más cómoda.*

Fases de procesado y almacenaje

En esta fase, la información ha llegado al destinatario, es hora de analizarla y posiblemente almacenarla. Aquí surgen las dudas.

¿Qué información personal guarda el sitio?

Imaginemos que hemos hecho un pago mediante tarjeta de crédito, aunque hemos suministrado datos necesarios para la transacción, el sitio no necesita para nada almacenar nuestro número de tarjeta, fecha de caducidad y mucho menos el código de seguridad (algo prohibido por los bancos en los términos de uso de los pagos con tarjeta).

¿Qué seguridad tiene?

Qué pasa si la información que posee queda expuesta ante cualquiera. ¿Es una máquina segura? ¿Es vulnerable a exploits conocidos?

¿Cuáles son sus politicas de seguridad y privacidad?

A nadie le hace gracia que compartan sus datos personales con terceras empresas (aunque sean del mismo grupo/multinacional) para en un futuro despertarle de madrugada para ofrecerle una conexión ADSL "al mejor precio".

Sinceramente es un tema escabroso, ya que el usuario ha expuesto su identidad y ya no la controla. La única solución es para los diseñadores de estos sistemas, que suponiendo que sean profesionales y el negocio serio, se tomarán las molestias de asegurar sus máquinas, no almacenar información que no necesitan y no guardar la información personal sin cifrar/hashear.

1.2.2. Conclusión

Hasta ahora podemos sacar dos conclusiones:

- Los ataques son rentables, mucho. Es difícil rastrear al delincuente y el grado de automatización actual permite ataques a gran escala.
- La parte más débil del ciclo de la información es el usuario. Casi todos los ataques se centran en la fase de introducción.

1.3. El primer credencial, la contraseña

La forma más común de controlar el acceso a un recurso es mediante una contraseña, un secreto compartido (palabra) entre el usuario y el servicio. A primera vista parece una buena idea.

1.3.1. Apogeo

Las contraseñas se adoptaron rápidamente como método de autenticación por las siguientes razones:

- Es la forma más simple y universal de autenticación. Es la proyección de la llave del mundo físico (algo que se posee) al mundo virtual (algo que se sabe).
- Está implementado sobre la parte lógica, por lo que es el método más barato y rápido.
- Se supone que es más difícil olvidar algo que perderlo.
- Sirven como protección de todo tipo de recursos: acceso a servicios, encriptación de ficheros, instalación de programas, etc.

1.3.2. Autenticación

Autenticación es la operación en la que se verifica una aserción sobre la identidad, es decir, comprobamos que alguien es quien dice ser. Existe un caso de uso el que se permite a un usuario anónimo acceder al sistema, realizar una instalación... , este tipo de credenciales usados para tener acceso aunque no se verifique su identida se llaman çredenciales ciegos".

1.3.3. Autorización

Autorización es la operación en la que una vez se tiene la identidad del usuario, se establecen los permisos de acceso a los recursos. Quién puede acceder a qué. No es lo mismo decir quién soy a reclamar mis privilegios. Este concepto tiene su razón de ser en la conectividad, imaginemos una red local de una empresa en la que yo me autentico una vez para establecer mi identidad contra un servidor, y esta máquina establece si tengo permisos (me autoriza) para usar ciertas impresoras, acceder a servicios de directorio, de compartición de ficheros, de correo, de voz sobre IP...

1.3.4. Sesión

La idea proviene del ejemplo anterior, establezo mi identidad una vez para acceder al sistema y me tienen que autorizar cada vez que quiero hacer uso de un servicio. En todo momento saben quién soy.

1.3.5. Peligros

El perder una contraseña (entendamos como perder que una tercera la conoce) no sólo afecta a su legítimo propietario, sino a todos los propietarios de su red de trabajo. Puede ser el comienzo de una escalada de privilegios para comprometer todo el sistema o simplemente un robo de identidad (que ya es bastante malo). Malas prácticas:

- Apuntarlas. Desvirtua todo propósito de protección, ya que si quiero acceder a un ordenador al que no tendría que poder hacerlo y la contraseña está apuntada en un papel pegado en la pantalla...
- Contraseñas débiles: comunes (1234, contraseña), fáciles de adivinar (datos personales, palabras de un diccionario o de longitud insuficiente).
- Reusarlas en varios sitios. Alguien podría crear un servicio de poco valor con el único fin de obligar a sus usuarios a registrarse con su email y una contraseña, y seguramente la mayoría de los usuarios usaría la misma contraseña que la de su correo. El delincuente ya tiene todos los datos para empezar a atacar.

1.3.6. Declive

Y hasta aquí es la parte bonita, porque todo cambia cuando se aplica este método a Internet. Con multitud de servicios no interrelacionados, la teoría dice que debemos usar una contraseña diferente por servicio. Hagamos números, tengo cuenta en: hotmail, gmail, messenger, flickr, tuenti, facebook, youtube, wordpress, twitter, skype, jabber y cuatro foros. Lo que hacen ¡diceiséis! contraseñas para mi identidad digital (en el caso de que solo tenga una identidad digital). No es descabellado empezar a hablar de una "fatiga"[Wikipedia, i] o caos en el manejo de mis credenciales. Veamos qué métodos se han usado hasta ahora para mitigar este efecto:

- Usar la misma contraseña para todos los servicios. Me expongo a un robo de identidad en toda regla, en cuanto se sepa esa contraseña, caerán todas mis cuentas.
- Cookies para mantenerme autenticado. Me evitan el tener que introducir mis credenciales tan a menudo, pero no solucionan mucho más.
- Rellenadores automáticos de formularios. Útiles en un principio pero limitan la movilidad, si cambio de máquina tengo que seguir acordándome de las contraseñas. Al ser un proceso automatizado puedo estar descuidando a quién le doy mis datos.

1.3.7. Las preguntas

En este punto surgen varias cuestiones

- ¿Soy yo mi clave? Estamos reduciendo la identidad de alguien a la posesión de este credencial y no es así, el credencial desbloquea la identidad, no es la identidad. En la práctica, se tiene que pensar en la identidad como en un concepto operacional, por lo que a veces surgen este tipo de reducciones tan radicales. Es cierto que alguien siempre es él mismo, pero las máquinas operan con abstracciones para reducir la complejidad del mundo real.
- ¿Con quién estoy negociando? ¿Por qué tengo que autenticarme ante alguien que es posible que no sea quien dice ser? Este es el primer paso para solucionar el phishing.

1.3.8. Pluralidad en la autenticación

Además de las contraseñas, se han desarrollado otros sistemas de autenticación, y actualmente existe una gran multitud en el que cada uno satisface las necesidades de su propia casa. En el mundo empresarial, donde la seguridad puede ser crítica (imaginemos un banco), se suelen usar sistemas de autenticación no basados en contraseña. Existen dos tipos principales:

- Basados en certificado: autenticación de cliente mediante SSL, tarjetas inteligentes (smartcards),etc.
- Basados en certificado: autenticación de cliente mediante SSL, tarjetas inteligentes (smartcards),etc.

No existen razones para que un sistema se imponga sobre el resto y se adopte como un estándar, así que el principal problema es que cada empresa es como una isla con su propia tecnología, lo que causa un aislamiento. ¿Cómo establecer relaciones empresariales para compartir datos o servicios? Este problema de interconexión, en un futuro lo llamaremos federación, se agrava (si por casualidad dos empresas se hubieran puesto de acuerdo para adoptar el mismo sistema) aún más cuando lo hacemos extensible a Internet, donde prima el individualismo y lo mejor no es cerrarse en métodos propios.

1.3.9. Usabilidad

Y al final de toda elucubración viene el usuario. Podemos reducir este factor a la navegación web y reformular preguntas.

¿Con quién estoy negociando? Barra de direcciones

La identificación de la otra parte, la que ofrece el servicio, viene determinada por la barra de direcciones (suponiendo que todo vaya bien). El problema surge que a veces esas direcciones no son todo lo aclarativas que debieran, o existen marcadores para automatizar el proceso o redirecciones o pop ups para ofuscar todo el proceso, por lo que en la mayoría de las ocasiones se le presta poca atención.

¿Cómo sé cuándo algo va mal? Experiencia y aspecto

La diferencia entre la experiencia de autenticación en un sistema operativo y los servicios de internet es que en el sistema operativo existe una consistencia entre las interfaces, mismo aspecto, mismo funcionamiento, ayudas, etc; y en los servicios de internet esto no pasa porque cada empresa tiene su imagen corporativa, lo que hace difícil recordar una experiencia satisfactoria para cada uno. Es cierto que existe una especie de consenso no escrito en el que lo más común es encontrar un formulario con dos campos, uno para el nombre de usuario y otro para la contraseña, pero no con el mismo aspecto en todos los casos, ni la misma disposición ni nada más allá de esto. Partiendo de que es fácil replicar el aspecto de una web y que incluso puede ser variable, ¿cómo se va a dar cuenta un usuario de que algo va mal basándose en la inconsistencia? ¿Qué señales puede tener de que algo no es como debiera ser?

¿Y los métodos no basados en contraseña?

Ya hemos visto que están relegados al mundo empresarial (hasta que se implante y adopte ampliamente el e-DNI), así que sin un soporte técnico, no es posible para el usuario medio instalar un arsenal de lectores de tarjetas con sus propios controladores y sus propios programas con interfaces no comunes.

La verdad es que el panorama es un poco caótico, ya que la única forma que tenemos ahora mismo de identificar y verifificar al proveedor del servicio es mediante certificados SSL. Confiando en

autoridades de certificación y en el sistema PKI. Funciona. La única pega es que para usarlo hace falta comprender mínimamente cómo funciona, algo que probablemente se le escapa al usuario medio. Al fin y al cabo esta es una sección sobre usabilidad.

Capítulo 2

Conceptos fundamentales

Me gustaría exponer el siguiente ejemplo para explicar posteriormente tres conceptos fundamentales en los que se basa.

Ejemplo(1) Imaginemos que yo, Samuel Muñoz Hidalgo, estudiante de la universidad de Alcalá de Henares (UAH), decido ir a la universidad de Roskilde en Dinamarca (RU) a pedir opinión sobre algunos aspectos de este proyecto.

Una vez allí, recuerdo que justo hoy mismo salía la nota de la última asignatura que me quedaba para acabar la carrera. Ni corto ni perezoso, saco mi portátil, me conecto a la red WIFI de la universidad de Roskilde con mis credenciales de la universidad de Alcalá de Henares y me dispongo a navegar por internet. Me meto en la página de la UAH, en la sección de "mi portal" y debo introducir otra vez los credenciales para ver las notas. - "¡Que pena, un 4,75, casi aprobado¡' Creo que sé cuál puede haber sido el problema. Me meto en mi cuenta del correo de la UAH (otra vez tengo que introducir mis credenciales) y concierto una cita con el profesor para la revisión del examen. Como quiero ir preparado para la revisión, reservo en la página de la biblioteca de la UAH (otra vez tengo que introducir mis credenciales) un libro relacionado con un problema del examen para la vuelta.

Posteriormente reviso mi cuenta de correo de Gmail (tengo que introducir mis credenciales de Gmail) y publico en mi blog de Blogger que estoy teniendo una estancia muy agradable por tierras escandinavas.

Veamos a continuación en qué tres pilares se basa el ejemplo.

2.1. Itinerancia o "Roaming"

El primer servicio que he disfrutado ha sido el de poder conectarme a la red WIFI de una universidad que no era la mía. Este sevicio se denomina itinerancia o "roaming" y permite al usuario acceder a redes inalámbricas distintas a la que le ofrece su proveedor.

El acceso a la red de la universidad de Roskilde está restringido, por eso he tenido que introducir unos credenciales. Que me permita hacer uso del servicio con unos credenciales de la universidade de Alcalá de Henares indica que tiene que haber algún tipo de acuerdo entre ambas entidades.

Otro ejemplo más común de itinerancia se da en la telefonía móvil, un cliente de la compañía A sale de su área de cobertura, encuentra que la compañía B le ofrece cobertura y utiliza los servicios de esta aun no siendo cliente. Existe un acuerdo comercial entre la compañía A y la B para ofrecer este servicio. Obviamente la compañía A le cobrará los servicios al cliente más un añadido por el servicio de itinerancia.

Una ventaja de la itinerancia es que es un proceso transparente para el usuario, es decir, hace uso de la red como si fuera la red de su proveedor.

2.2. Federación

Hemos visto que el servicio de itinerancia requiere una comunicación entre el proveedor original (entidad a la que pertenezco o de la que soy cliente) y el proveedor local (entidad que me ofrece el servicio inalámbrico) para compartir datos como la autenticación, autorización o la tarificación. Esta compartición de datos, normalmente a través de servicios, se llama federación.

La federación surgió como solución a determinadas necesidades comerciales. Retomemos el ejemplo de la itinerancia telefónica.

Vimos que el cliente del operador telefónico A, cuando no estaba bajo su cobertura (por ejemplo al salir del país), tenía acceso a la red del operador B y podía realizar llamadas de forma normal. Lo interesante en esta sección es la relación entre las compañías A y B. Es una relación comercial de federación. Cuando B localiza un terminal en su red con un número que sabe que no gestiona, le pregunta a A y esta se lo confirma. B permite al cliente usar su red. Cuando B tarifica al cliente por el uso de la red, se lo comunica a A que es la que maneja la información financiera del cliente (la que a fin de mes le pasa la factura). Este flujo de información se da porque A y B son socios comerciales y han federado los servicios necesarios (identificación del número telefónico y tarificación).

Lo mismo sucede en el ejemplo inicial, la universidad de Roskilde y la universidad de Alcalá de Henares forman parte de la misma federación (eduroam Europa), cumplen unas políticas comunes y tienen servicios accesibles entre ellas para permitir el acceso a otros alumnos que pertenecen a una entidad de esa federación.

2.2.1. Confederación

Una confederación es una federación de federaciones. Es un caso particular de federación en el que se distinguen varios niveles. Volviendo al caso de UAH-RU, ambas universidades no forman parte de la misma federación de forma directa. Eduroam Europa es una confederación formada por federaciones que son los propios paises europeos que la forman. Es decir, la UAH pertenece a la federación de RedIRIS (España) y la RU pertenece a la UNI-C (Dinamarca) y ambas a Eduroam Europa.

Cuando me conecto en la RU con mis credenciales, mi nombre de usuario tiene la forma de usuario@uah.es. Con esto, el servidor que me autentica en RU pregunta al de Eduroam Europa (confederación) , determina que es de España y pregunta al de RedIRIS (federación) y le pregunta al servidor de autenticación de la UAH quien manda siguiendo la misma ruta la validación o denegación del acceso.

2.3. Inicio de sesión

Se conoce como inicio de sesión el acto de introducir credenciales para acceder a un servicio de forma exitosa.

Con la expansión de Internet, utilizamos cada vez más servicios, con lo que el inicio de sesión es una tarea monótona y repetitiva. A continuación muestro cómo han evolucionado las tecnologías en este campo y cómo se tiende a federar el inicio de sesión en favor de la comodidad de todos (usuario, desarrolador y administrador del servicio).

2.3.1. Inicio de sesión unificado - USO (Unified Sign On)

El inicio de sesión unificado o USO permite al usuario usar los mismos credenciales para accedera a los servicios ofrecidos bajo un mismo dominio.

Ejemplo(2) He hecho uso de servicios que me ofrece la UAH como revisión de notas, correo electrónico y biblioteca. Cada vez que he accedido a uno de ellos he tenido que introducir mis credenciales, pero siempre eran los mismos.

2.3.2. Inicio de sesión único - SSO (Single Sign On)

El inicio de sesión único o SSO permite al usuario iniciar sesión en un dominio de servicios web introduciendo una sola vez sus credenciales. Viéndolo desde un punto de vista más técnico, consiste en autenticarse contra un servicio de SSO y este nos iniciará una sesión por cada servicio del dominio, esas sesiones se guardan en "cookies" temporales (recuerda que el protocolo HTTP por si mismo no guarda estados).

Ejemplo(3) Estar introduciendo mis credenciales para cada nuevo servicio que uso de la UAH es algo demasiado engorroso y no deseable (ver contraseñas). Todos estos servicios pertenecen a la misma entidad (UAH) pero están gestionados por distinto personal. Una buena solución sería federar la autenticación, de forma que al autenticarme en uno de ellos tuviera acceso a todos.

Ejemplo(4) Cuando he introducido mis credenciales para ver mi correo de Gmail, he realizado un inicio de sesión único y desde ese momento he tenido acceso a todos los servicios de Google (como Blogger) introduciendo mi contraseña una vez.

Fin de sesión único - SLO (Single Log Out)

El servicio de SSO tiene una fecha de expiración gracias a las "cookies" temporales y a que los navegadores borran las cookies de sesión cuando son cerrados. Esta manera de proceder sería confiar mucho en el navegador y en que el usuario no va a dejar la máquina sin más y venga otro y le robe la sesión. Por esto, muchos servicios ofrecen un fin o cierre de sesión único o SLO.

- Las sesiones serán cerradas y no se podrán piratear o robar.
- El servicio se cierra de una forma controlada (realizando las transacciones pendientes y reduciendo los cuelgues o problemas de inestabilidad).
- Se liberan recursos más rápidamente.

Recomiendo mirar los siguientes enlaces para profundizar en el tema.

- Single Sign On - Single Log Out (Feide)
  http://docs.feide.no/fs-0034-1.0-en.html

2.3.3. Inicio de sesión único unificado - uSSO (unified Single Sign On)

El inicio de sesión único unificado o uSSO es la extensión natural del servicio de SSO a entornos heterogéneos, esto es, fuera de la web.

Ejemplo(5) Imaginemos que hubiera hecho uSSO al conectarme a la red WiFi de la RU. Solamente en ese momento hubiera tenido que introducir mis credenciales para acceder a servicios relacionados con la universidad de la UAH (ya que es ella quien me reconoce como alumno y quien me ofrece los servicios web).

La principal ventaja es la comodidad extendida que pudiera ofrecer el servicio de SSO, pero sin duda los problemas aparecen exponencialmente en la fase de diseño debido a la naturaleza variada de los propios servicios. No es trivial implementar un método de autenticación común a todos.

La solución que más se está extendiendo es la de contar con un servicio web que haga de intermediario entre las diferentes aplicaciones, ofreciendo un servicio web (valga la redundancia) para ciertas comunicaciones como el inicio de sesión y accediendo a almacenes de datos (bases de datos o servidores LDAP) en donde se intercambian claves y otra información (que consultará el servicio que lo necesite).

Me gustaría ilustrar este enfoque con la figura 2.1 tomada del SSH Federado del proyecto GT-STOR (almacenamiento distribuido federado).

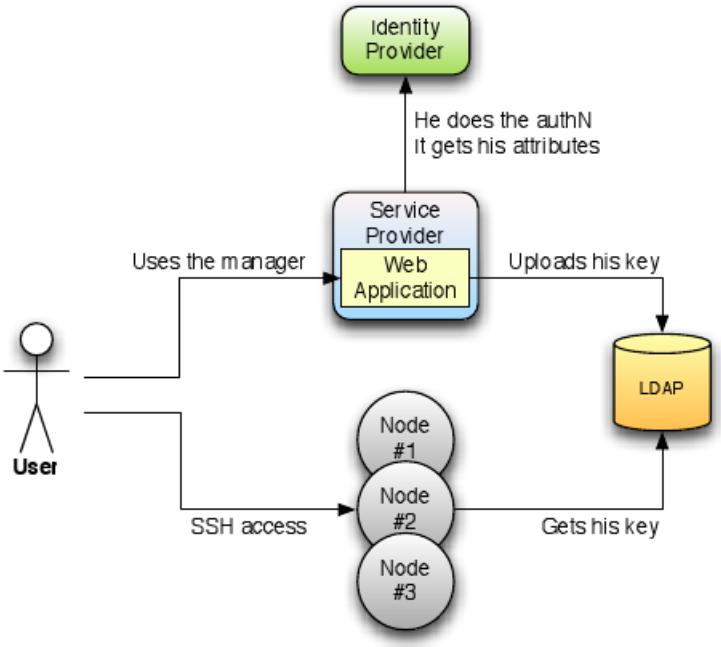

Figura 2.1: SSH federado.

Otro enfoque quizá más elegante pero menos práctico sería la creación de una capa de autenticación que nos permitiera gestionar estas capacidades en las aplicaciones que se conectan a la red. La solución contaría con una librería que manejara las funciones necesarias y luego, y aquí viene el gran problema, que cada aplicación hiciera uso de la librería. Esto conlleva el reescribir cada programa para incorporarle las nuevas características, sin duda algo inviable a corto plazo. Digo que este enfoque es más elegante porque desde un punto de vista estratégico, producirá un software mejor adaptado al escenario, más flexible y cuya estructura (al menos la del módulo de autenticación) se ajusta con una necesidad bien definida, establecida y ordenada, no como una solución ád hoc'. Otro inconveniente nada desdeñable es la ausencia de un estándar para gestionar el servicio de uSSO, lo que imposibilita la creación de dicha librería. En el capítulo 7, sobre SAML, el lector verá que ni si quiera existe una forma totalmente aceptada para hacer SSO (y eso que sólo se reduce a la web), con lo que es de esperar que la estandarización de un método de uSSO sea un problema de varios órdenes de magnitud mayor.

Recomiendo mirar los siguientes enlaces para profundizar en el tema.

- GT-STOR: Almacenamiento distribuido federado
  http://backnoise.com/?gt2009-stor

- Deliverable DJ5.3.1: Documentation on GÉANT2 unified Single Sign-ON (uSSO) Requirements
  http://www.geant2.net/upload/pdf/GN2-06-261v4-DJ5-3-1_Documentation_on_GEANT2_uSSO_requirements_20070201095918.pdf

2.4. Aclaraciones sobre los términos

La unión de la autenticación con la usabilidad (de lo que trata este capítulo) es un tema bastante nuevo dentro del mundillo informático. Es por esto que no existe un consenso a la hora de denominar las distintas técnicas a las que hago referencia. Si además añadimos las técnicas de mercadeo a la hora de buscar nombres a nuevos productos comerciales, es normal que exista una confusión entre los estudiantes novicios e incluso entre los profesionales más curtidos en el tema.
La razón de esta sección es la de respaldar bibliográficamente y con el mayor rigor posible otras acepciones para los términos ya definidos e indagar en nuevas denominaciones.

2.4.1. Sobre USO - (Unified Sign-On)

Hemos visto que es el uso reiterado de los mismos credenciales para hacer inicio de sesión en varios servicios.

El inicio de sesión único (SSO, donde una vez iniciada la sesión, todos los servicios son accesibles) se distingue del inicio de sesión unificado (USO, donde el usuario debe iniciar sesión en cada servicio independientemente, pero con los mismos credenciales). El término "inicio de sesión simplificado" se usa para resaltar una mejora de la experiencia de usuario en los controles e acceso, quizá usando un inicio de sesión único o unificado.

[Davies and Shreeve, 2007, pág 19]

... Se migraron cuatro sistemas con tres identificadores de inicio de sesión separados y sus contraseñas a un proceso de inicio de sesión unificado un único nombre de usuario y contraseña.

[Sentillion, 2005, nota de prensa]

Sin embargo en otros contextos se utiliza como sinónimo de inicio único de sesión unificado o uSSO. Dejo como prueba dos patentes americanas que tratan sobre lo mismo (si es que no son la misma). En ellas no habla de servicios web, sino de dominios y lo escenifica con un ejemplo de inicio de sesión en UNIX y después otro en Windows.

- System and method for unified sign-on - US Patent 7275259
  http://www.freepatentsonline.com/7275259.html
- System and method for unified sign-on - US Patent 7392536
  http://www.freepatentsonline.com/7392536.html

Y para concluir, hay autores que mantienen la ambigüedad. Expongo la siguente cita.

Inicio de sesión unificado: las contraseñas están sincronizadas o la autenticación está centralizada. Piensa en servidores RADIUS con contraseñas, ... la autenticación de de Active Directory, ... servidores centrales de autenticación fuerte o los muchos programas de sincronización de contraseñas que no requieren un servidor central de autenticación (pero que requieren un sistema que las coordine en tu espacio de contraseñas global).

[Scher, 2004]

Insisto en ojear el artículo citado anteriormente, sobre todo la sección de ventajas e inconvenientes.

2.4.2. Sobre el inicio de sesión único empresarial o E-SSO (Enterprise Single Sign-On)

Los sistemas de inicio de sesión único empresarial (E-SSO) está diseñados para minimizar el número de veces que un usuario debe introducir sus credenciales para iniciar sesión en múltiples aplicaciones. La solución de E-SSO inicia la sesión automáticamente y actúa como un rellenador de contraseñas donde no se puede automatizar de otra forma el inicio de sesión.

...

Estas soluciones son normalmente programas que acompañan al usuario y automatizan el proceso de inicio de sesión y cambio de contraseña para ciertas aplicaciones (Win/Java /Ajax/Web/Consola) ... por esto desde la perspectiva de los agentes E-SSO solo son almacenes de credenciales.

[Wikipedia, c]

Es decir, que los productos de inicio de sesión único empresarial son una aproximación a nuestro uSSO. Mezclan varias técnicas, y cuando no queda más remedio, recurren a los gestores de contraseñas.

Para profundizar más en el tema, recomiendo mirar algún producto comercial. Yo he encontrado Tivoli, el gestor de acceso de IBM. Se puede ver una introducción en
http://www.ibm.com/common/ssi/fcgi-bin/ssialias?infotype=SA&subtype=WH&appname=SWGE_TI_SE_USEN&htmlfid=TIW14017USEN&attachment=TIW14017USEN.PDF.

2.4.3. Sobre Universal Sign-On

El ejemplo siguiente demuesta cómo se toma este término como nuestro uSSO (inicio de sesión único unificado). Lo aplica a todo lo cotidiano, así que más bien sería una especie de uSSO para la vida cotidiana. No deja de ser una predicción hecha allá por el 2004.

- Why stop at Single Sign On, why not Universal Sign On?
  http://blogs.zdnet.com/Ou/?p=12

El siguiente ejemplo es la página web de una empresa que ofrece una solución de Universal Sign-On, pero que en realidad es un ESSO no intrusivo.

- AccessMatrix USO - i-Sprint Innovations http://www.i-sprint.com/products_uso.htm

2.4.4. Sobre Universal Single Sign-On

En muchas ocasiones se utiliza este término y Unified Single Sign-On indistintamente, incluso dentro de la misma organización y del mismo enlace.

Pongo el siguiente ejemplo:
http://www.geant2.net/server/show/nav.00d00a005001/targetBlock/5319/viewPage/2
Mira el enlace "Documentation on GÉANT2 Universal Sigle Sign-On (uSSO) Requirements (DJ5.3.1)", ábrelo y verás que el título real es "Documentation on GÉANT2 unified Single Sign-On (uSSO) Requirements".

Capítulo 3

Definición del problema

Este breve capítulo tiene la intención de materializar en algo concreto lo que llevamos visto y culminar esta parte modelando una solución que será el eje de lo que resta de proyecto.

3.1. Situación actual

En los capítulos anteriores hemos visto la problemática de la gestión de la identidad y la de la usabilidad en el inicio de sesión. Haciendo una revisión de la conectividad dentro del mundo académico y de los servicios que ofrece, es posible proponer una mejora en el software que gobierna esa infraestructura (también conocido como "middleware") en pro de la comodidad del usuario y facilidad de gestión del sistema.

Describo los elementos del escenario actual que existe en la UAH.

- Red inalámbrica eduroam, ofrece acceso a internet y la red universitaria y también ofrece un servicio de itinerancia dentro de los centros educativos que forman parte de la confederación.
- Servicios web de la universidad.
  - Correo electrónico.
  - Sevicios de reserva y renovación de libros en la biblioteca.
  - Aula virtual.
  - Portal para la consulta de notas, asignaturas. matriculadas, progreso académico...
  - etc.
- Servicios que ofrecen otras entidades a usuarios universitarios.
  - Consulta de revistas.
  - Descuentos en compra de libros, material, viajes, etc.
  - Gestión de becas.
  - etc.

Describo los principales problemas que existen ahora mismo.

- Es necesario iniciar sesión por cada servicio que se desea utilizar.
  Hay que introducir los credenciales de usuario para acceder a la red eduroam, introducir credenciales para acceder a las notas, introducir credenciales para consultar el correo... ¡introducir credenciales para todo!

- Servicios como la consulta de revistas están limitados a peticiones desde direcciones IP provenientes de la universidad. Si quiero hacer uso desde este servicio y no estoy en la red universitaria (estoy en mi casa, por ejemplo), tengo que acceder a la intranet de la universidad y hacer la consulta desde esa red. Sin duda todo un engorro.

- Los descuentos a universitarios se efectúan presentando el carnet que te acredite como estudiante cuando vas a comprar en persona. ¿Pero qué sucede con las compras en tiendas virtuales? ¿Cómo puede tener la certeza la tienda de mi condición de estudiante? Es más ¿qué pasa si esa tienda solo ofrece descuentos a estudiantes de centros concretos?

- Otro problema derivado de los dos casos anteriores es la información sobre mí que comparte la universidad con el proveedor del servicio. Yo no lo sé y es un dato a tener en cuenta. Si ese proveedor va a usar mi cuenta de correo de la universidad para mandarme publicidad, es posible que yo no quiera aceptar el trato y rehuse a acceder a sus servicios.

- Por último existe un grave problema de automatización en los trámites oficiales como pedir becas o convalidaciones de créditos o subvenciones para una asociación universitaria. El papeleo. Viajes a distintas oficinas del estado para entregar documentación impresa que ellos mismos tienen.

3.2. Características de la solución

Teniendo en cuenta la problemática anterior y los conocimientos que llevamos acumulados es de lógica pensar que se puedan automatizar no sólo los inicios de sesión, sino también la información necesaria para el correcto funcionamiento de los servicios vistos en la sección anterior.

Lo que se pretende es crear una prueba de concepto con la que demostrar que es posible unificar los inicios de sesión de forma que solo haya que introducir los credenciales una vez, cuando se accede a la red WiFi. De esta forma estaríamos logrando un verdadero uSSO. Para los servicios web, requeriré de una herramienta de SSO y para la gestión de la identidad, de un metasistema de identidad (en la parte segunda de este proyecto se explican estas herramientas). Se puede ver un esquema que refleja esta intención en la figura 3.1.

Un caso de uso sería el de un estudiante que se conecta a la red WiFi de la UAH y a partir de entonces está reconocido (tiene la sesión iniciada) en todos los servicios del dominio de la herramienta de SSO. En los servicios que no forman parte del inicio de sesión única (revistas, agencias de viajes, becas del estado) puede identificarse como usuario de la UAH mostrando su identidad digital de forma sencilla y controlar la información que cede a estas terceras partes. A pesar de todo esto, el usuario solo ha introducido una vez sus credenciales, cuando se conectó a la WiFi.

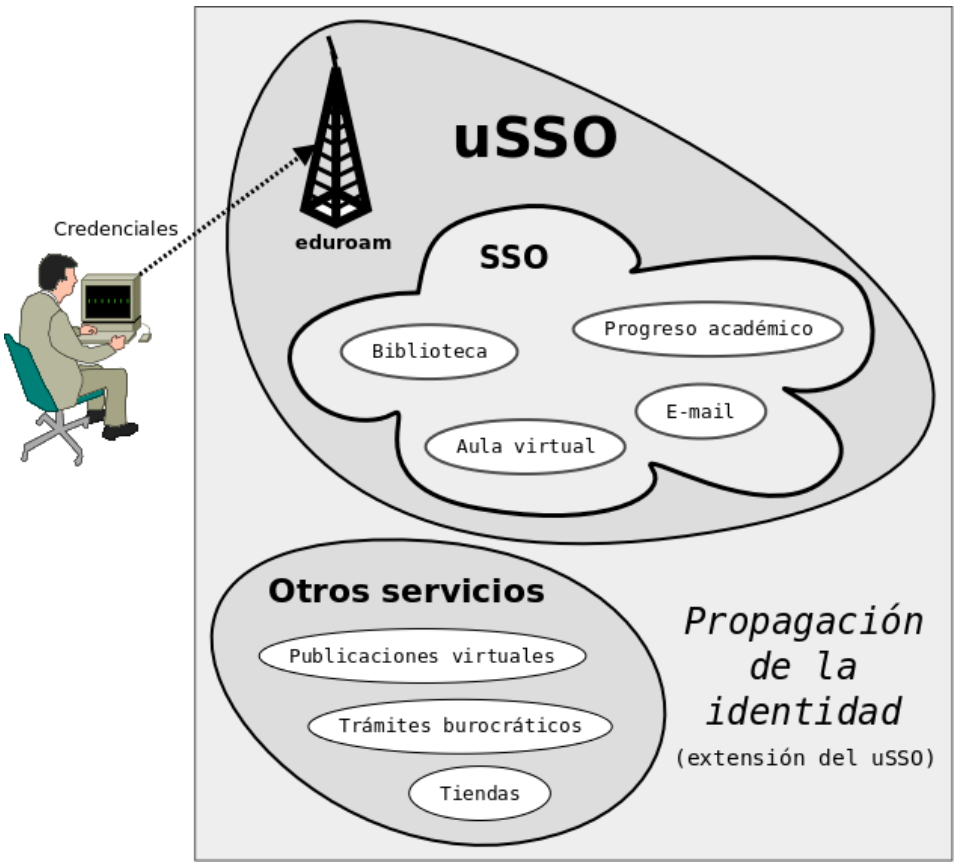

Figura 3.1: Posible resultado

Parte II

Las herramientas

Capítulo 4

Las 7 leyes de la identidad

La necesidad

En la parte anterior vimos cómo Internet es una red a nivel mundial sin una autoridad central que la controle: muchos usuarios que tienen sus propios hábitos, empresas que usan tecnologías dispares, multitud de formas de hacer las cosas... La red no es más que una enorme infraestructura de comunicación en la que proveedores de servicios hacen sus negocios y en base a estos adecúan sus sistemas de autenticación. Proveedores que están sedientos de identidades para adecuar sus objetivos comeriales: fidelizar clientes, analizar hábitos de consumo para mejorar ofertas, mantener perfiles actualizados, permitir el teleempleo, etc. Es decir, cada negocio gestiona las identidades como quiere.

La forma en que se define una identidad, se envía y se gestiona en una transacción, define el contexto. Todo el mundo tiene interés en gestionar el contexto de sus transacciones. Es la falta de un estándar lo que ha favorecido la aparición de múltiples soluciones y simultáneamente un caos diréctamente proporcional al número de estas. Como en el mundo de las comunicaciones se tiende a simplificar por niveles, se está viendo la necesidad de añadir una solución genérica en forma de capa de identidad. Intentaremos buscar una solución que en potencia sea capaz de representar las necesidades presentes y futuras (al menos las básicas) de una forma genérica, ya que si no estaremos cayendo en uno de los métodos que sólo satisfacen uno de los escenarios mencionados anteriormente.
Objetivos de esta capa:

- Que se ajuste a todas las necesidades, que sea universal.

- Que tenga la aceptación de todas las partes y en especial la del usuario: privacidad, seguridad, sencillez de uso, control y una experiencia aceptable.

Visto lo visto, ¿qué solución podemos formular? Basándonos en la experiencia hay que tener en cuenta que un problema similar a otro que ya hemos solucionado, no implica afrontarlo de la misma manera, las dos formas de resolverlos pueden no parecerse en absoluto. Se vio en el primer capítulo que el mayor fiasco ha sido intentar extender el uso de una tecnología para tratar con problemas que exceden en varios órdenes de magnitud a los que originalmente se plantearon. Una solución eficaz requiere de un distanciamiento de las herramientas actuales y una análisis desde cero del problema. No obtendremos instantáneamente un producto, pero sentaremos las bases para un resultado satisfactorio. Entendiendo las propiedades básicas que debería tener la solución, no podemos sino acercarnos al éxito.

Kim Cameron: el dios de la identidad

Kim Cameron (ver figura 4.1) es el arquitecto jefe de Identidad en la división de Identidad y Seguridad de Microsoft [Microsoft, 2009]. Su mayor logro (relevante para este proyecto, sin menosprecio de otros) fue allá por el 2004-2005 el poner de acuerdo a: su jefe (Bill Gates), líderes del movimiento "Open Source", y más pesos pesados del mundo de Internet en un esfuerzo para definir la ciencia del manejo de la identidad y cómo aplicarla a la computación. Con un énfasis principal en entender qué funcionó y qué no funcionó (Microsoft Passport) en el pasado y por qué. Se examinaron hechos desde varios puntos de vista: tecnológico, consideración social, usabilidad, privacidad... Una vez llegado a un consenso, publicó un borrador en el 2005 donde se sintetizaban en forma de siete leyes los principios que todo sistema de manejo de la identidad debía cumplir.

Figura 4.1: Kim Cameron

Para saber más, recomiendo los siguientes enlaces:

- `http://www.identityblog.com`
  Todo el trabajo que llevó a cabo, así como explicaciones y ejemplos se pueden ver en su blog
- `http://www.networkworld.com/power/2005/122605-cameron.html`
  Una pequeña biografía.
- `http://www.microsoft.com/presspass/features/2005/may05/05-12DigitalID.mspx`
  Preguntas y respuestas en un pase de prensa.

Sobre las leyes A continuación explico las leyes de la identidad, no sin antes decir tres cosas a tener en cuenta:

1. No son dogmáticas, provienen de la experiencia y son verificables. Tienen su razón de ser.
2. Provienen del consenso.
3. A pesar de su importancia, no son más que meras recomendaciones. No leyes como tal.

4.1. El usuario consiente y controla

Los sistemas de identidad sólamente deben revelar información del usuario cuando éste lo consienta.

[Cameron, 2005]

Es el principio fundamental de la identidad centrada en el usuario, la ley más importante.
El usuario tiene que saber:

- Qué información va a dar.
- A quién.
- Cuándo.
- Para qué es esa información.
- Qué consecuencias puede tener la transacción.

Este (¿nuevo?) nivel consciencia que se le exige al usuario es la clave para permitir el control.
Como consecuencia directa, se espera solucionar los ataques de phishing al saber con quién se está negociando.
Puede existir un conflicto con los sistemas de S.S.O. ya que estos, al compartir información sin el conocimiento directo del usuario, hacen que pierda el control de sus datos. A favor está su usabilidad, el no tener que hacer inicio de sesión para cada servicio, y como solución se propone el avisar al usuario de a quién y qué información se transmite de forma que pueda cancelar la operación.

4.2. Mínima exposición para un uso restringido

La solución que expone la menor cantidad de información y mejor limita su uso es la solución más estable a largo plazo.

[Cameron, 2005]

Esta ley se puede resumir en que no hay que dar más información de la estrictamente necesaria para la transacción.

Como primer ejemplo me gustaría exponer el caso de las tarjetas de fidelización de clientes de los centros comerciales. Es una tarjeta que nos hace el centro comercial por ser cliente habitual (la verdad es que los operadores de caja tienen la obligación de ofrecerla siempre) que ofrece descuentos en algunos productos, te envían mensualmente alguna revista o te obsequian con algún artículo de poco valor el día de tu cumpleaños. Algo que parece totalmente inocente, se convierte en una herramienta de control de hábitos de consumo un tanto preocupante. No el que hagan estadísticas para mejorar su logística o sus ofertas, sino el que dirijan esa información contra el propio consumidor.

- El primer caso es el de alguien que se va a hacer un seguro médico. ¿Qué pasaría si la aseguradora obtuviera los hábitos de compra de bebidas alcohólicas, tabaco, aperitivos grasos, salados y dulces? Probablemente el negocio de la aseguradora sería aún mayor a expensas de la salud o la economía de los ciudadanos [E.P.I.C.].

- El segundo caso, más particular, se refiere a un juicio que tuvo el supermercado Von's de California en el que se le demandaba por una herida sufrida por un cliente al resbalar sobre un suelo mojado. Von's se intentó defender probando mediante la tarjeta de fidelización que el cliente era comprador habitual de alcohol y que probablemente fuera un alcohólico [E.P.I.C.].

Esto nos lleva a la siguiente pregunta, ¿de verdad es necesario que el vendedor sepa mi nombre y me identifique cuando compro algo que es perfectamente legal? El quiere vender, yo comprar, fin.

El segundo ejemplo es un caso más refinado. Si compro alcohol, el tendero solamente tendría que verificar si soy mayor de edad, no saber mi fecha de nacimiento (dato más personal que un un simple Sí/No soy mayor de dieciocho años).

El tercer ejemplo versa sobre usos indirectos que se derivan de esta mala práctica. En muchos almacenes de información (bases de datos de empresas, administraciones públicas, etc) aparecen los datos que necesitan ligados a mi Número de Identificación Fiscal (número que en muchos casos no es necesario y sólo tiene el fin de identificador). Un ataque a estos almacenes permitiría reconstruir mi identidad de forma casi completa relacionando piezas de identidad a través del N.I.F. cosa que si usara un identificador inventado, solo podrían hacerlo mediante inferencias más complejas a través de otros datos (nombre, apellidos, correo, fecha de nacimiento...), lo que dificultaría los ataques en escala general y podría imposibilitarlos si se hicieran de forma dirigida contra una identidad.
A los diseñadores de bases de datos, el NIF es buena clave primaria, pero mala forma de proteger la identidad, no lo uses/almacenes a no ser que lo necesites.

La conclusión es que este principio proviene de la experiencia y muestra un gran valor estratégico. Sólo con el tiempo nos alegraremos de haberlo tenido en cuenta (aunque nos llamen paranoicos).

4.3. Partes justificables

Los sistemas de identidad digital deben ser diseñados de tal forma que la revelación de la información se limite a partes que tienen un papel necesario y justificable en la relación.

[Cameron, 2005]

Qué fue MS Passport
Esta ley viene tras el fracaso del servicio Passport de Microsoft. MS Passport surgió poco después del año 2000 como un servicio de autenticación, las empresas con presencia en la web, podían subcontratar la autenticación de sus usuarios a Microsoft a través de este ingenio. Tenía su razón de ser, puesto que los medios tecnológicos y la infraestructura necesaria para llevar a cabo la tarea conllevaban un coste y una responsabilidad que en términos económicos superaba la subcontrata. Pero con el tiempo, las empresas vieron que el servicio no se adecuaba a su forma de actuar, aumentaron sus recursos tecnológicos y hubo un gran incremento en el número de personas que accedían a Internet. La información empezó a ser considerada como fuente de riqueza (minería de datos, publicidad, personalización). Existían métodos técnicos viables económicamente para que cada web tuviera su propio servicio de autenticación. Pero el detonante fue la pregunta ¿por qué Microsoft debe tener un registro de quién entra en qué web? El proveedor del servicio de autenticación ya no era necesario, sus condiciones eran abusivas (en alusión a la privacidad), se había convertido en una parte no justificada en las transacciones.
Pero Passport no murió, se adaptó a los nuevos tiempos y exigencias con el nombre de Windows

Live ID, hay una pequeña introducción en la siguiente URL http://msdn.microsoft.com/en-us/library/bb288408.aspx

En el fondo, "Partes justificables" es un refinamiento de la primera ley. El usuario debe entender qué papel toma cada parte en la transacción y que consecuencias tiene. Debe comprender las políticas de privacidad y manejo de datos y tener como última opción el cancelar el proceso.

Como primer ejemplo retomo el caso de las tarjetas de fidelización de la ley anterior. Ya no es el enfoque opcional de revelar nuestros datos, sino si ¿llevaríamos a cabo compras si nos obligaran a tenerlas? ¿Nos someteríamos a ese nivel de control? Si ha contestado que sí, pase al segundo ejemplo.

Mi segundo ejemplo es algo más contundente. Supongamos que cada vez que uso mi Documento Nacional de Identidad (DNI) para probar que soy mayor de edad, el Gobierno de España (entidad que lo avala como documento oficial) tuviera constancia de ello, es decir, relacionara mi persona con un negocio o actividad en una determinada fecha y por supuesto lo almacenara en alguna base de datos. Puede parecer algo "inocente" de primeras, pero veamos un caso no tan descabellado. Pongamos que cuando acabe este proyecto quiero celebrarlo por todo lo alto y me embarco en un viaje con mis amigotes a Amsterdam (Holanda) y entre otros menesteres: compro alcohol, entro a un Cofee Shop y contrato los servicios de alguna señorita. Gracias a mi tez poco maltratada (la luz de una biblioteca causa poco envejecimiento), he tenido que probar en cada caso mi mayoría de edad usando mi DNI, quedando constancia de ello. ¿Qué consecuencias tiene este registro aun sin haber realizado nada ilegal? Esto lo dejo a elección del lector (posición social, oposiciones, política, chantajes, entrevistas de trabajo, seguros, etc). Pero lo más probable es que cualquiera de nosotros, antes de aceptar esta condición, nos dejemos una barba digna de Matusalén y olvidemos el DNI en los otros pantalones.

4.4. Identidad dirigida

Un sistema universal de identidad debe soportar tanto identificadores omnidireccionales para uso de entidades públicas como identificadores unidireccionales para uso de entidades privadas, facilitando el descubrimiento de servicios y evitando la liberación innecesaria de manejadores de relación.

[Cameron, 2005]

Identificador

Un identificador es el dato (o datos) que relaciona unívocamente nuestra identidad digital con nuestra entidad física. Veamos unos ejemplos:

- EJ1: En España hay mucha gente que se llama Juan Carlos, pero si me refiero al DNI 00000010-X ya me estoy refiriendo al rey de España. El número del DNI es un identificador.
- EJ2: ¿Cuánta gente hay que se llame Juan Carlos y sea rey de España? Esos dos datos, nombre y título nobiliario son el identificador. Aunque no vale para el resto de la plebe.
- EJ3: ¿Cuántas personas nacieron el 5 de Enero de 1938 en Roma y se llaman Juan Carlos? Fecha y lugar de nacimiento y nombre también pueden ser identificadores.

Direccionalidad
La direccionalidad de un identificador se establece segun quien vaya a hacer uso de ese identificador. Podemos establecer dos tipos:

- **Omnidireccional**
  Es un identificador que todo el mundo conoce, perfecto para entidades públicas que quieren dar a conocer sus servicios. Ejemplos de estos identificadores son las URLs o los certificados.

- **Unidireccional**
  Es un identificador pensado para el usuario, relaciona al usuario con un servicio, el usuario no tendría ese identificador en otro servicio. La consecuencia directa de este uso es que no se podrían reconstruir identidades a partir de almacenes de datos de dos servicios diferentes, pero hemos visto que también pueden ser identificadores otros conjuntos de datos, así que el problema es más complejo de lo que parece.

4.5. Pluralismo de operadores y tecnologías

Un sistema universal de identidad debe permitir la interoperabilidad de múltiples tecnologías de identidad usadas por múltiples proveedores de identidad.

[Cameron, 2005]

Un sistema de manejo de identidad que aspira a ser una solución universal no puede caer en modas, tiene que ser algo genérico. Puesto que la autenticación tiene un contexto de uso: inversiones, preferencias de los usuarios, niveles de seguridad, modas, formas de expresar la identidad, etc; la solución pasa por dos palabras, incluir y tolerar. No se sabe qué nuevos contextos traerá el futuro, pero está claro que si se quiere prevalecer como opción, habrá que estar preparado. Probablemente por esta ley se ganó la acepctación del mundo "Open Source".

Un sistema universal debe abarcar la diferenciación a la vez que nos reconoce simultáneamente en varios contextos: como ciudadano, empleado, cliente, persona virtual.
...
Necesitamos un protocolo de encapsulación simple...
...
El metasistema universal de identidad debe ser policéntrico y polimórfico, lo que permitirá a la "ecología de la identidad" emerger, evolucionar y autoorganizarse.

[Cameron, 2005]

4.6. Integración humana

El metasistema universal de identidad debe definir al usuario como un componente del sistema distribuido a través de mecanismos de comunicación no ambigua humano-máquina que le protejan de ataques contra la identidad.

[Cameron, 2005]

El usuario es la parte más importante del sistema, el gestiona su identidad. Mientras la comunicación máquina-máquina es rígida y por consiguiente predecible, al extenderla al caso persona-máquina, surgen los problemas. La solución pasa por maximizar la comprensión por parte del usuario y minimizar la ambigüedad para hacer el margen de error lo más pequeño posible. Esto se consigue mendiante la introducción de un protocolo a seguir en la interfaz, será rígida y poco creativa, pero usable, sencilla y segura (que es lo que interesa).

4.7. Experiencia consistente a través de contextos

El metasistema de identidad debe garantizar a sus usuarios una experiencia simple y consistente a la vez que permite la separación de contextos mediante múltiples operadores y tecnologías.

[Cameron, 2005]

Echemos un vistazo a posibles identidades que se usan en internet:

- **Navegar**, identidad autoemitida para navegar por la web (no damos información real).
- **Personal**, identidad autoemitida para sitios con los que tengo relación (apodo y un correo).
- **Comunidad**, identidad pública para colaborar con otros.
- **Profesional**, identidad pública emitida por mi empresa.
- **Tarjeta de crédito**, identidad emitida por mi banco.
- **Ciudadano**, identidad emitida por mi gobierno.

¿No sería maravilloso cosificar las identidades para hacer uso de ellas cuando lo necesitaramos? Convertirlas en algo tangible como los directorios de un ordenador o los ficheros o el sistema de sonido. De esa forma cuando una web nos pidiera una identidad, la selecionaríamos sin más y estaríamos autenticados. Se mejoraría enormemente la usabilidad (sencillez, control de usuario, consciencia de qué identidad usamos, variedad de métodos de autenticación según la seguridad requerida) a la par de que la autenticación sería siempre de la misma manera, proporcionando una experiencia consistente que se rompería ante cualquier imprevisto (¿ataque?).

Capítulo 5

El metasistema de identidad

Ya tenemos las leyes de la identidad para evaluar si una solución es apta. A partir de aquí intentaremos definir de forma abstracta (sin entrar en implementaciones ni tecnologías) cómo debería ser nuestro metasistema de identidad.

El metasistema de identidad es una arquitectura interoperable para el manejo de la identidad digital que asume que el usuario tendrá varias identidades digitales basadas en múltiples tecnologías subyacentes, implementaciones y proveedores. Usando esta aproximación, los clientes serán capaces de seguir usando las infraestructuras de identificación en las que se invirtió, elegir la tecnología de identidad que mejor se les ajuste y migrar de forma más sencilla de viejas a nuevas tecnologías sin sacrificar la interoperabilidad.

[Microsoft]

5.1. Definiciones

5.1.1. Metasistema

Un metasistema es un sistema de sistemas que se comunican entre sí, de esta forma se cumple la ley de "pluralidad de operadores y tecnologías".

Esta construcción tan ambigua viene de una pregunta casi filosófica. "¿Por qué van a abandonar las empresas sus sistemas (en los que hicieron una gran inversión) para adoptar uno nuevo?" La respuesta con la definición que hemos propuesto es que ¡no hace falta! Así todo lo nuevo que se pueda extraer de nuestro modelo se incorporará como un añadido a lo ya existente.

El manejo de la identidad conlleva

- Manipular principios abstractos comunes.
- Realizar acciones específicas.
- Cubrir roles canónicos.

Son conceptos comunes a todos los esquemas de autenticación, sólo varía el que se implementan de forma diferente. Esta visión se correspondería con un T.A.D. (tipo abstracto de datos), que es "un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de

datos para el modelo"[Wikipedia, l] muy usado en informática a la hora de realizar análisis y plantear diseños (nótese la particular afinidad del autor por esta herramienta).

El metasistema define conceptos y operaciones universalmente válidos en el contexto de la identidad sin preocuparse de la implementación. No sustituye a los sistemas actuales, sino que actúa de unificador, es una capa superior que se sirve de ellos. De esta forma da cabida a tecnologías pasadas y futuras, centrándose en el manejo de la identidad y despreocupándose de métodos específicos.

5.1.2. Desacoplamiento

En este contexto, definimos desacoplamiento como la negación del término acoplamiento, que en informática se refiere al "grado de interdependencia que hay entre los distintos módulos de un programa"[Wikipedia, o].

El metasistema debe garantizar el desacoplamiento entre las operaciones relacionadas con la identidad y sus implementaciones, de forma que al cambiar una implementación, su operación asociada no varíe. Esto se consigue añadiendo el nivel de abstracción que es el metasistema en sí, similar a la capa de identificación que requeríamos en la sección anterior.

Es una característica deseable en todo sistema, ya que aunque puede añadir cierto grado de ineficiencia (inherente a toda abstracción), es compensado de sobra por su inteligibilidad, su facilidad de uso y el poder ser extendido, adaptado y mantenido en un futuro.

5.1.3. Claim (aserto)

Un "claim" representa un hecho sobre algo o alguien. En el contexto de la identidad digital, un claim es una aserción sobre la entidad a la que se refiere.
Ejemplos de claims:

- Me llamo Samuel.
- Mi nacionalidad es española.
- Estudio en la U.A.H.
- Vivo en Japón (que aun siendo un dato falso, es un claim perfectamente válido).

5.1.4. Identidad digital

Una identidad digital es un conjunto de claims que provienen de la misma fuente referidos a una entidad.

La confianza en la fuente determinará la validez de esa identidad.

Podemos distinguir dos tipos de identidades:

- **Autoemitidas**, en las que emitimos asertos sobre nosotros mismos. Por ejemplo cuando nos registramos en una web.
- **Emitidas por un tercero**. Por ejemplo el D.N.I. que lo emite el Gobierno de España.

5.1.5. Confianza

La confianza es el deseo del sujeto de creerse las aserciones proporcionadas por otros.

Ejemplo(6) Si A se fía de B, A tomará por válido lo que diga B.

Ejemplo(7) Puedo viajar por la U.E. porque al enseñar mi D.N.I., los guardias de las aduanas se fían del Gobierno de España y me dejan pasar.

La confianza está limitada a un cierto ámbito.

Ejemplo(8) Al comprar alcohol, el tendero se puede creer que me llamo Juan si así se lo digo (identidad autoemitida), pero al intentar yo comprar tiene que verificar que soy mayor de edad y no le vale mi palabra, exigirá un credencial de otro tipo que no sea mi palabra (identidad emitida por un tercero en el que confíe para obtener esa información).

5.2. Roles en el metasistema

Una vez aclaradas las definiciones, tendremos que modelar los actores que participan en la transacción. De esta forma podremos definir los roles esenciales o arquetípicos para entender las relaciones y expectativas de cada parte y comprender su funcionamiento y sus propiedades.

La primera consecuencia es que cada entidad que participe en el proceso de identidad se englobará dentro de uno o varios roles. Esta extensión también se puede aplicar a las relaciones del mundo real.

Definimos tres roles fundamentales:

5.2.1. Relying Party (Parte Confidente) o RP

Suele ser la entidad que proporciona un servicio limitado a cierto público. Antes de proporcionarnos acceso al servicio, necesita que le suministremos una identidad. Es una consumidora de identidades.

Ejemplo(9) Voy a la biblioteca, cojo un libro, me acerco al mostrador y quien me atiende (parte confidente), necesita mi carnet de socio (identidad, conjunto de aserciones sobre mi persona emitido por la biblioteca) para autorizarme a llevarme el libro.

En esencia sigue siendo la puerta de entrada al servicio que queremos usar, solo que en vez de credenciales, acepta identidades.

5.2.2. Subject (Sujeto) o S

Un sujeto es alguien que tiene una identidad digital unidireccional. Otra definición es la de entidad cuya identidad digital es consumida por una RP.
Un sujeto es el usuario de los servicios.

5.2.3. Identity Provider (Proveedor de Identidad) o IP o IdP

Un Proveedor de Identidad es una entidad que emite identidades digitales acerca de un sujeto con el que mantiene algún tipo de relación.

Es un concepto ampliamente extendido en el mundo real pero casi desconocido en la autenticación digital tradicional. Proveedores de Identidad del mundo real:

- Gobierno: DNI, pasaporte.
- Biblioteca: carnet de socio.
- Empresa: carnet de empleado.
- Dirección General de Tráfico: carnet de conducir.
- Médico: certificado de buena salud.
- Perico el de los palotes: cuando le dice al tendero que me fíe la compra porque él responde por mí.
- Yo mismo: cuando relleno una encuesta.

La confianza en un IP se sustenta en que se le considera una **autoridad** en el contexto de la información que maneja. Así un gobierno puede probar mi ciudadanía pero no mi pertenencia a un club de lectores y viceversa con el carnet del club de lectores. Todo se basa con el tipo de relación que tengo con el IP. Una RP deberá tener esto en cuenta si por ejemplo quiere ofrecer contenido +18 o descuentos por pertenencia a un club, y aquí se ve envuelto el usuario con la última ley "Experiencia consistente a través de contextos": ver quién emite mi identidad y qué relación tiene con el destinatario me permite controlar la transacción.

Identidad autoemitida: la controversia

Retomando este caso particular de identidad...
Una identidad autoemitida es la identidad que un sujeto puede emitir/afirmar sobre si mismo. Aunque en un principio parece algo inutil, hagamos memoria. Hasta ahora solo hemos usado identidades emitidas por terceros, que no prueban más que el que mantenemos algún tipo de relación con el IP. ¿Qué pasa con una RP que quiere emitir un servicio no crítico, normal y corriente de tipo: foro, red social, portal...? Lo que le interesa es que el usuario X siempre sea el mismo (misma persona física) en cada proceso de autenticación, no que su identidad esté respaldada por un tercero. Y aquí entra en juego la identidad autoemitida, en la que existe un identificador unidireccional e invariable para mantener la correspondencia identidad-usuario pero cuyos claims (nombre, apellidos, e-mail, país, idioma...) están definidos por el usuario. Este tipo de identidad puede ser útil para navegar y participar en el mundo digital de forma anónima y consistente, ya que permite tener un perfil siempre actualizado e idéntico para los servicios que lo requieran.

5.2.4. Ejemplo Final: el vendedor de alcohol

Como último ejemplo de esta parte, expondré el desgastado caso de ir a comprar alcohol desde la perspectiva de:

- Roles:
  - S: yo, el comprador.
  - RP: el tendero.
  - IP: el Gobierno de España.
- Relaciones:
  - S-RP: relación comercial, comprar alcohol.
  - S-IP: ciudadano del reino de España.
  - RP-IP: relación de confianza, el Gobierno de España es una autoridad para certificar que yo soy mayor de edad.
- Reglas:
  - S-RP: debo probar que soy mayor de edad.
- Medios:
  - IP-¿S-¿RP: mi D.N.I. el elemento expedido por el IP (con sus medidas de seguridad para garantizar la fuente) constituye mi identidad y es la clave para satisfacer la regla y que la relación de S-RP se lleve a cabo.

5.2.5. Conclusiones

Cambio de paradigma

Es evidente que el metasistema representa un auténtico cámbio de paradigma en el manejo de la información personal referida a la identidad digital.
Hemos pasado del **paradigma clásico** en el que la identidad es almacenada por el servicio y se libera con unos credenciales al **nuevo paradigma** en el que el usuario suministra la identidad al servicio.

¿Estamos delegando la autenticación?

Es la pregunta más común al hablar de confianza en identidades emitidas por terceras partes. ¿Otro MS Passport? En absoluto. Lo que hace una RP es verificar que el sujeto (S) tiene una relación con el proveedor de identidad (IP). Relación necesaria para iniciar el servicio, pero los datos propios del servicio hay que mantenerlos, perfiles de usuario, reservas, direcciones de pedidos, etc. La diferencia con MS Passport es que este servicio almacenaba todo. Es como preguntar si estamos delegando la autenticación al hacer federación, pues no, ya que en todo momento podemos verificar y controlar los datos.

5.3. Componentes del metasistema

El documento "Microsoft's Vision for an Identity Metasystem"[Microsoft] establece como claves en un metasistema de identidad los siguientes componentes.

5.3.1. Identidades basadas en claims

Es la forma, a la que hemos llegado de forma intuitiva, de representar una identidad digital.

5.3.2. Negociación

Es el modo de garantizar la interoperabilidad, mediante un protocolo de negociación.

Si un sistema es capaz de usar un conjunto de tecnologías, el metasistema debe proveer de una forma para ponerse de acuerdo con otro sistema, también con su propio conjunto de tecnologías, para iniciar una transacción basada en alguna de las tecnologías que ambos comparten.

Si la intersección de los conjuntos de tecnologías que soportan es nulo, no podrán comunicarse; es un resultado posible de la negociación. Es muy importante que la negociación exista como característica del propio metasistema y no tenga que implementarla las diferentes partes. Así será lo más formal posible la forma que se especifican los requisitos (o políticas) de cada parte.

5.3.3. Protocolo de encapsulación

Es el protocolo de comunicación común que entiende cada participante de forma que no hace falta usar una tecnología específica para establecer una conexión (es una comunicación tecnológicamente neutra). Debe permitir transmitir información según las reglas de las diferentes tecnologías.

Ejemplo(10) El protocolo de encapsulación más común es el correo. Da igual en el lenguaje en el que vaya escrita la carta, o siquiera si es una carta, que mientras lleve el sello y la dirección (reglas del protocolo), llega.

5.3.4. Transformadores de claims

Hasta ahora hemos supuesto que existen unos tipos de claims estándar (nombre, apellidos, fecha de nacimiento, etc) que usan todos los proveedores de identidad, no es así. Volvemos al caos anárquico que es Internet, reflejo del mundo en el que vivimos y lo aplicamos a las terceras partes. Los transformadores de claims son elementos para crear-modificar claims a partir de información, solo eso. Sirven principalmente para dos cosas:

1. **Compatibilidad**. Cada entidad maneja su propio modelo de datos y es un engorro a la hora de hacer que un sistema interoperere con otro.

   Ejemplo(11) Es posible que una entidad tenga tres claims (nombre, primer apellido, segundo apellido) que se correspondan con un solo claim que maneja otra empresa (primer apellido+segundo apellido,+nombre todo como una cadena, no como tres). La comunicación a nivel formal entre máquinas sería imposible ya que para ellas son campos totalmente diferentes, pero con un transformador podríamos convertir uno en otro y solucionar el problema.

2. **Inferencia de datos**. De la información que tenemos podemos crear nueva información para necesidades específicas.

   Ejemplo(12) Imaginemos una web de un grupo político que admite comentarios de ciudadanos del país. Queremos garantizar el anonimato del que comenta a la vez que comprobar que es un ciudadano de pleno derecho (mayor de edad y con la nacionalidad). Deberíamos requerir una identidad del gobierno (IP) con un claim del estilo "mayor de edad Sí/No". El estado no guarda el dato de si somos mayores de edad, pero sí nuestra fecha de nacimiento. Podría emitir el claim basándose en esta última.De esta forma están comprobados los requisitos y garantizado (en cierta medida) el anonimato, ya que la única información personal que podría usar el partido para identificarnos es la dirección IP (y es posible también hacerla anónima). Todos contentos.

   Ejemplo(13) Imaginemos una web de un colegio que restringe el acceso a las fotos de la función escolar por temas de privacidad al salir menores de edad. Se quiere garantizar el acceso a los familiares directos de los niños y además solamente a las fotos en las que salga el chaval. Podríamos requerir un claim del estilo "es familia de" en el que viniera el nombre del niño y además "tipo de familiar" por si queremos llevar un registro. Como IP pondríamos a una autoridad gubernamental que es la que lleva el registro de ciudadanos. Esos dos campos no existen en las bases de datos del gobierno como tales, pero se pueden inferir a través de la relación X es hijo de Y,Z que aparece en el DNI.

Consecuencias secundarias de este elemento son:

- Se potencia el desacoplamiento entre sistemas, tenemos un potente mapeador de campos.
- Se pueden crear grandes escenarios con cadenas de confianza.

  Ejemplo(14) Sólo quiero negociar con gente que esté certificada por cierta empresa (IP), pero esta empresa toma datos de un banco (IP) que admite identidades generadas por el Gobierno de España (IP).

5.3.5. Experiencia de usuario consistente

El metasistema busca implicar al usuario para que las decisiones que tome sobre el manejo de su identidad las haga desde una posición informada y razonable. El medio de realización será una interfaz comprensible, consistente, fácil de aprender, con acciones predecibles, integrada, que proporcione el protocolo a seguir por el humano (protocolo del que se habló en la sexta ley) y funcione siempre de la misma forma sin importar qué tecnologías esté manejando. También debe presentar al usuario la información que va a exponer en cada transacción. De esta forma estará asegurada ante ataques la fase de introducción de datos (vista en el capítulo 1).

La fase de transporte se asegura con criptografía. Encapsulado HTTPS.

La fase de almacenaje escapa al control del usuario, aunque el metasistema ayuda de varias formas:

- Lo más importante es que el usuario decide si suministra o no su identidad.
- El uso de identificadores unidireccionales evita el tener información personal.

- Al suministrar una identidad, no hace falta almacenar cierta información, la podemos obtener con la autenticación en el inicio de sesión. Así ante un ataque a la base de datos, será difícil que relacionen datos con identidades. Lo que no se almacena no se puede robar.

- En última instancia la RP es la responsable de la información que recibe.

5.4. El baile de la identidad

En esta sección describiré dos escenarios posibles en los que se puede usar el metasistema. Se seguirá manteniendo el nivel de abstracción para no entrar en implementaciones que distraigan al lector con detalles no relevantes.

5.4.1. Escenario canónico

La figura 5.1 representa un ejemplo clásico. Cada rol enumerado se representa como una entidad (o instancia) y es perfectamente extrapolable al ejemplo del vendedor de alcohol.

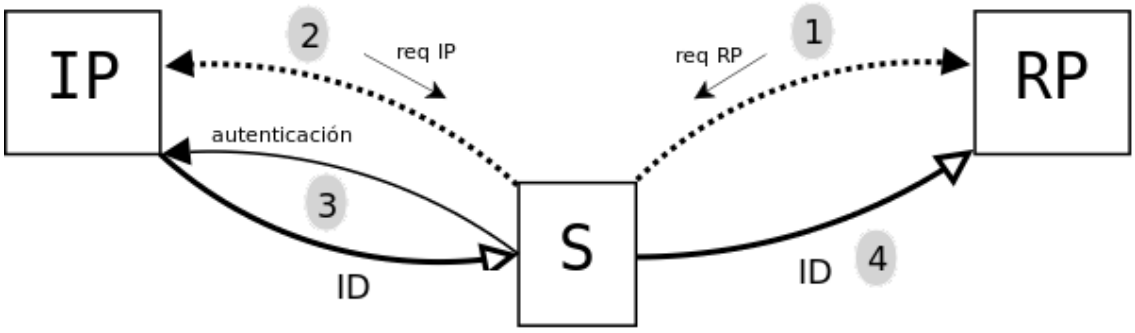

Figura 5.1: Escenario canónico

El flujo de datos es el siguiente:

1. S quiere acceder a un servicio protegido. Para autenticarse tiene que presentar una identidad digital. Negocia con la RP para adquirir sus requisitos y su política.
   S comprueba si maneja alguna identidad que cumpla los requisitos (identidad emitida por el IP con ciertos claims).

2. S negocia con el IP para ver cómo se invoca, obtiene sus requisitos.

3. S se autentica en el IP pide una identidad que satisfaga ciertos requisitos. El IP emite la identidad.

4. S inspecciona la identidad emitida y si está conforme con los datos, se la envía de vuelta a la RP (consumidora de identidades) ganando acceso al servicio.

Se ve que no hay imposiciones tecnológicas más allá de la de que los protocolos de negociación y encapsulación tienen que ser comunes a todas las partes. En ningún momento se ha definido la tecnología con la que se representa una identidad digital ni las formas de autenticarse ante el IP.

5.4.2. Escenario de confianza negociada

Vamos a usar los **transformadores de claims** para modelar un escenario de relaciones empresariales (ver figura 5.2). El caso es el siguiente: un empleado quiere comprar piezas de una empresa (a través de su web) que mantiene un acuerdo con la empresa para la que trabaja, es decir, se fía de esta.

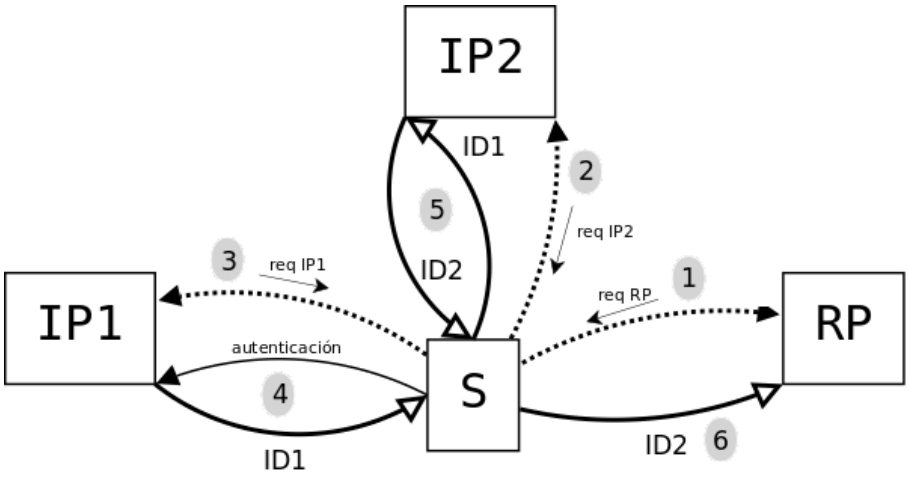

Figura 5.2: Escenario de confianza negociada

Los actores serán:

- S, el empleado.
- RP, el servicio de venta de piezas.
- IP1, el proveedor de identidad de la empresa para la que trabaja S.
- IP2, un transformador de claims que toma identidades de IP1 y emite identidades que satisfacen los requisitos de la RP. Pertenece a la empresa del trabajador.

Como las dos empresas son independientes, el modelo de datos difiere, y las identidades emitidas por una no satisfacen los requisitos de la otra.

Veamos el flujo de datos:

1. S quiere acceder a un servicio protegido y debe presentar una identidad digital. Adquiere los requisitos y la política de la RP.
   Requisitos:
   - Una identidad digital emitida por IP2.
   - Los claims responden al modelo de la empresa de venta de piezas, y además necesita un claim que diga cuál es el crédito de S.

   S comprueba que no tiene ninguna identidad digital que se ajuste a lo pedido, con lo que se procede a inspeccionar los requisitos de IP2.

2. Negociación con IP2 para ver qué exige.
   Requisitos:

- IP2 exige una identidad emitida por IP1.

S comprueba que tiene una identidad que satisface los requisitos de IP2.

3. Negociación con IP1 para ver cómo se invoca, obtiene los requisitos.
4. S se autentica ante IP1 y recibe una identidad digital ID1. A continuación decide si se la presenta al IP2.
5. S presenta la identidad digital ID1 al IP2 y éste le devuelve otra identidad digital ID2.
6. S presenta la identidad digital ID2 a la RP y gana acceso al servicio.

Es una secuencia más larga que el escenario canónico, pero también mucho más potente. Gracias a la automatización, la longitud de la cadena de confianza (en esto caso IP1-IP2) puede ser tan larga como se requiera con retrasos de tiempo mínimos en la autenticación. Se aprecia que la negociación (cuando pedimos requisitos) y la encapsulación (cuando mandamos identidades) son las dos únicas constantes tecnológicas y que a la vez libran a las partes de una excesiva coordinación (en las formas de autenticarse) que desencadenaría en un gran acoplamiento.

Capítulo 6

Servicios Web (Web Services) - WS

En este capítulo perderemos parte de la abstracción del metasistema de identidad para entrar en detalles de implementación.

Hasta ahora hemos visto que las constantes tecnológicas comunes a todas las partes son dos.

1. El **protocolo de encapsulación**.
2. El **protocolo de negociación**.

Deben ser invariables para garantizar la interoperabilidad.

Como no es una decisión baladí el adoptar una tecnología para implementar estos protocolos, más que nada por su trascendencia estratégica, deberemos cuestionarnos los requisitos tecnológicos que debe satisfacer la solución final.

1. Que cumpla su función (lo que llevamos viendo hasta ahora).
2. Que no sea dependiente de una tecnología o plataforma (interoperabilidad).
3. Tiene que haber sido elegida por consenso entre todos lo que tienen voz en este mundillo (para evitar monopolios, patentes, litigios legales y por temas de interoperabilidad).
4. Debe ser accesible desde una amplia gama de plataformas, contextos y tipos de conectividad (requisito estratégico por temas de ampliación, no se sabe lo que depara el futuro).

La opción que más se ajusta a una solución ideal son los **servicios web**. La idea clave reside en que si la industria puede establecer un conjunto de estándares comunes que definan los aspectos clave de la comunicación basada en mensajes, la interoperabilidad directa entre diferentes plataformas es posible.

Según la W3C (World Wide Web Consortium), los servicios web son:
Un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

[W3C, b]

Podemos entender los servicios web como servicios de datos que se ofrecen a través de internet y que utilizan la Web para transmitir su información. El uso de de la Web los condiciona a comunicarse mediante mensajes (unidad mínima de transmisión) y a que estos estén en formato de texto (legibles por un humano). De estos dos últimos datos se deduce que es fácil transmitir-recibir e interpretar mensajes (interoperabilidad garantizada). Puesto que estos servicios están pensados para comunicaciones entre máquinas, es de suponer que la sintaxis de esos mensajes será no ambigua y todo estará formalizado de acuerdo a unas normas.

6.1. Fundamentos

Veamos las tecnologías en las que se basan los servicios web.

6.1.1. XML

XML (Extensible Markup Languaje) o lenguaje extensible de marcas es un metalenguaje que sirve para describir datos.

XML no es más que una sintaxis estricta que le permite a las máquinas interpretar los documentos de forma no ambígua.

En la mayoría de documentos XML se pueden distinguir ciertos elementos.

- Prólogo: líneas opcionales con las que comienza el documento e indican la versión XML, el tipo de documento, la codificación, etc

  Ejemplo(15) `<?xml version="1.0" encoding="ISO-8859-1" ?>`

- Cuerpo: información del documento, lo que no es el prólogo. La característica indispensable es que tiene un elemento raíz.
- Elemento: cada una de las marcas que utiliza para englobar datos. Un elemento se define mediante etiquetas.

  Ejemplo(16) En el lenguaje XHTML definimos el elemento cuerpo de una web con la etiqueta body de la siguiente forma: `<body>Cuerpo de la web </body>`. Los elementos sienen una serie de propiedades o atributos, se pueden anidar, etc.

Existe una multitud de tipos de documentos que basan su sintaxis en el XML para describir:

- XHTML: páginas web.
- SVG: gráficos vectoriales.
- RSS: subscripción a noticias.
- ODF(Open Document Format): documentos ofimáticos.
- Documentos de configuración de programas.

Otra de las cualidades del XML es que es modular, lo que permite definir etiquetas propias asociadas a un esquema ("namespace") para evitar ambigüedades.

Ejemplo(17) La etiqueta <p>en XHTML significa un cambio de párrafo, pero en mi aplicación significa persona. Para evitar esta indeterminación, en mi documento XML habría una propiedad (xmlns) que determinara a qué esquema de datos pertenece la etiqueta.

```
<body xmlns:SMH="samuelmh_esquema">
<p>Esto es un párrafo.</p>
<SMH:p>Hola, soy Samuel Muñoz Hidalgo</SMH:p>
<SMH:div>¡Soy divertido!, no un bloque CSS</SMH:div>
<body>
```

<SMH:p>hace referencia a la etiqueta p del esquema "samuel_mh". He especificado y usado el prefijo "SMH". Si tuviera más etiquetas en ese esquema, podría usarlas con el mismo prefijo, sin tener que redefinirlo, siempre y cuando esté en un nivel anidado a la definición.

Recomiendo mirar los siguientes enlaces.

- Una explicación del XML en 10 puntos.
  http://www.w3.org/XML/1999/XML-in-10-points
- Tutorial sobre XML de la w3schools.
  http://www.w3schools.com/xml/default.asp
- Guía breve sobre XML.
  http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML
- XML en la Wikipedia.
  http://en.wikipedia.org/wiki/Xml

6.1.2. HTTP

HTTP(HyperText Transfer Protocol) o Protocolo de Transferencia de HiperTexto, es un protocolo de la capa de aplicación que se usa para transmitir cada mensaje de la Web (como una página web).

Datos:

- Viene especificado en la RFC 2616.
  http://tools.ietf.org/html/rfc2616
- Utiliza el puerto 80 del protocolo TCP.
- Es un protocolo sin estado.

Lo utilizaremos para enviar nuestros mensajes de servicio web por su simplicidad y casi universalidad. Es el protocolo que mejor se adapta a nuestras necesidades.

Seguridad

Podemos tener la necesidad asegurar el envío de mensajes para evitar ataques en la fase de transporte (como vimos en el primer capítulo, evitar que alguien ojee lo que mandamos). El protocolo HTTP, nos proporciona una seguridad en el transporte mediante una extensión, el protocolo HTTPS (HyperText Transfer Protocol Secure) o Protocolo de Transferencia de HiperTexto Seguro. No es más que HTTP sobre TLS para garantizar un canal seguro extremo a extremo de la comunicación [RFC, 2000b].

Otras alternativas
Es posible utilizar otros protocolos para ofrecer servicios web, por ejemplo a través de correo electrónico (protocolo SMTP).

6.1.3. SOAP

SOAP (Simple Object Access Protocol) o Protocolo de Acceso Simple a Objeto es una especificación para intercambiar mensajes a través de un servicio web. Podemos considerarlo como el servicio web canónico.

La comunicación es sencilla, se basa en el modelo petición-respuesta. Se manda un mensaje, se recibe otro.

Si XML es la sintaxis, el esquema que define SOAP es la semántica. La figura 6.1 muestra la estructura básica de un mensaje SOAP.

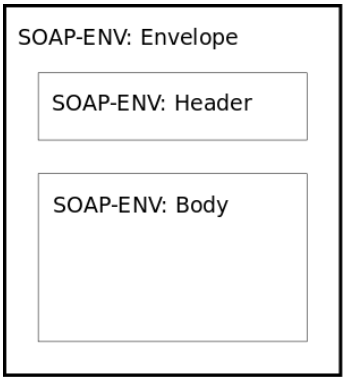

Figura 6.1: Esquema básico de un mensaje SOAP

Enlaces recomendados.

- Especificación de la W3C.
  http://www.w3.org/TR/soap/

- SOAP - Wikipedia (versión inglesa).
  http://en.wikipedia.org/wiki/SOAP

- Tutorial de la w3schools.
  http://www.w3schools.com/soap/default.asp

6.2. Esquemas extendidos - WS*

Uno de los fuertes de los servicios web, por estar basado en XML, es la modularidad. Es posible adaptar la especificación a cualquier necesidad sin más trabajo que el de crear un nuevo esquema XML para dar soporte a nuestras capacidades. Los esquemas creados hasta ahora se denominan genéricamente como WS-* y satisfacen necesidades como la descripción del servicio (WDSL), descripción de políticas de uso (WS-Policy), etc.
Veamos los esquemas más utilizados.

6.2.1. WSDL (Web Services Description Language)

- **ESPECIFICACIÓN:** http://www.w3.org/TR/wsdl
- **USO:** Describe las operaciones que ofrece un servicio. Cómo se invoca cada función y qué devuelve así como el formato de los mensajes.

6.2.2. WS-Addressing

- **ESPECIFICACIÓN:** http://www.w3.org/Submission/ws-addressing/
- **USO:** Sirve para establecer direcciones en las que se pueden encontrar recursos. Por ejemplo referencias a URLs. La forma de decir dónde están las cosas.

6.2.3. WS-Policy

- **ESPECIFICACIÓN:** http://www.w3.org/Submission/WS-Policy/
- **USO:** Describe la política/reglas del servicio. Los requisitos que hay que cumplir para invocar una operación (como estar autenticado).

6.2.4. WS-PolicyAttachment

- **ESPECIFICACIÓN:** http://www.w3.org/Submission/WS-PolicyAttachment/
- **USO:** Sirve para asociar políticas (WS-policy) a las diferentes partes (a un descriptor WDSL o UDDI).

6.2.5. WS-Security

- **ESPECIFICACIÓN:**
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- **USO:** Permite aplicar criptografía al mensaje de dos formas:

  1. Cifrando el mensaje
  2. Firmando el mensaje

  No define ningún algoritmo criptográfico nuevo, sino que posibilita usar alguno que ya exista. A grandes rasgos el formato del mensaje SOAP es:

  - En la cabecera se establece:
    - La operación (firma o cifrado)

    - El algoritmo que se usará
    - El material criptográfico necesario, como por ejemplo una clave pública.
    - En el caso de que se firme también apareceran:
        - El valor de la firma del cuerpo.
        - Algún método de canonicalización de los datos.
- El cuerpo:
    - Cifrado en el caso de que estemos cifrando.
    - Datos en claro si solo se firma.

Se asume que el material criptográfico está en la construcción "security token" y es propio del algoritmo. Por otra parte hay especificaciones satélite (traducción literal del inglés) conocidas como perfiles de testigo o "token profiles". Cada perfil describe cómo derivar ese material criptográfico a partir de la tecnología usada, traduciendo las peculiaridades del sistema en características genéricas del tipo WS-Security.

Ejemplo(18) Un servicio web obliga a que el cuerpo del mensaje esté firmado sin especificar el tipo de clave que se usa. Un usuario firma un mensaje con el material criptográfico obtenido de su certificado x.509 y otro usuario lo hace con material obtenido de un ticket Kerberos. A partir de los perfiles, el servicio podría generar ese material y verificar la firma (buscando en un almacén de certificados o formando parte del dominio Kerberos).

La potencia de este esquema radica en que permite usar criptografía simétrica, asimétrica, resúmenes (hash) y se mantiene al margen del algoritmo. Una solución genérica con la que puedo usar desde un simple rot13 (o cifrado del césar) hasta curvas elípticas.

6.2.6. WS-Trust

- **ESPECIFICACIÓN:** http://docs.oasis-open.org/ws-sx/ws-trust/
- **USO:** Es una extensión del WS-Security que añade operaciones para emitir, renovar y validar el material criptográfico o "security token""(a partir de ahora me referiré a él como token a secas). Mientras que WS-Security sólo permite el envío de tokens, éste se aprovecha y lo convierte en un TAD (tipo abstracto de datos). La forma de ofrecer estas nuevas operaciones es a través de un servicio web especial llamado STS.

STS (Security Token Service)
Es un tipo especial de servicio web que define este esquema. Su labor es la de transformar tokens de WS-Security. Se le suministra un token, y devuelve otro mediante un sistema de petición (RST - Request for Security Token) y respuesta (RSTR - Request for Security Token Response).

6.2.7. WS-MetadataExchange

- **ESPECIFICACIÓN:** http://www.w3.org/Submission/WS-MetadataExchange/
- **USO:** Es un esquema que permite al invocador de un servicio obtener sus metadatos (normalmente esquemas WDSL o políticas).

6.2.8. WS-SecurityPolicy

- **ESPECIFICACIÓN:** http://docs.oasis-open.org/ws-sx/ws-securitypolicy/

- **USO:** Es un esquema que se basa en WS-Policy para definir un marco que exprese requisitos de seguridad de una forma estándar, como requerir la presencia de un token o definir qué partes van firmadas o cifradas y con qué claves.

6.2.9. WS-Federation

- **ESPECIFICACIÓN:** http://www.ibm.com/developerworks/library/specification/ws-fed/

- **USO:** Es un esquema que se basa en WS-Trust y WS-Security para ofrecer una capa de abstracción con la que definir escenarios en los que hay federación. Dada una topología del problema en la que habrá: clientes, servicios y STSs; WS-Federation establecerá la secuencia de mensajes que debe ser generada por las partes para obtener un resultado. Según los actores podemos describir dos casos:

  - Solicitante activo
    Es capaz de usar criptografía en los mensajes y probar la posesión de las claves. Por ejemplo un servicio web.
  - Solicitante pasivo
    No es capaz de usar criptografía de la forma anterior, tiene que usar seguridad en el transporte (HTTPS), no en el mensaje. Por ejemplo un navedador web.

  En este punto podemos definir el modo de funcionamiento del metasistema como "federación centrada en el usuario". No es un caso de federación clásico ya que las relaciones entre las partes no son tan fuertes pero el concepto es similar.

6.3. Servicios Web en el metasistema

Tras la descripción técnica, veamos cómo integramos esta tecnología en el metasistema.

6.3.1. Componentes del metasistema como características de los WS

Negociación

Con WSDL, WS-Policy y WS-SecurityPolicy se establecen los requisitos básicos de uso del servicio. WS-MetadataExchange permite obtener estos requisitos y así se mantiene el acoplamiento entre las partes en niveles bajos.

Encapsulación

WS-Security ofrece unas características interesantes para encapsular la información que queramos mandar. En nuestro caso es la identidad digital (podemos concebirla como un token) y toda la fase de autenticación.

6.3.2. El baile de la identidad II - Confianza negociada

Tomaré el escenario de confianza negociada del capítulo anterior (ver figura 5.2) por ser más completo que el canónico. La figura 6.2 refleja el flujo de mensajes WS-*.

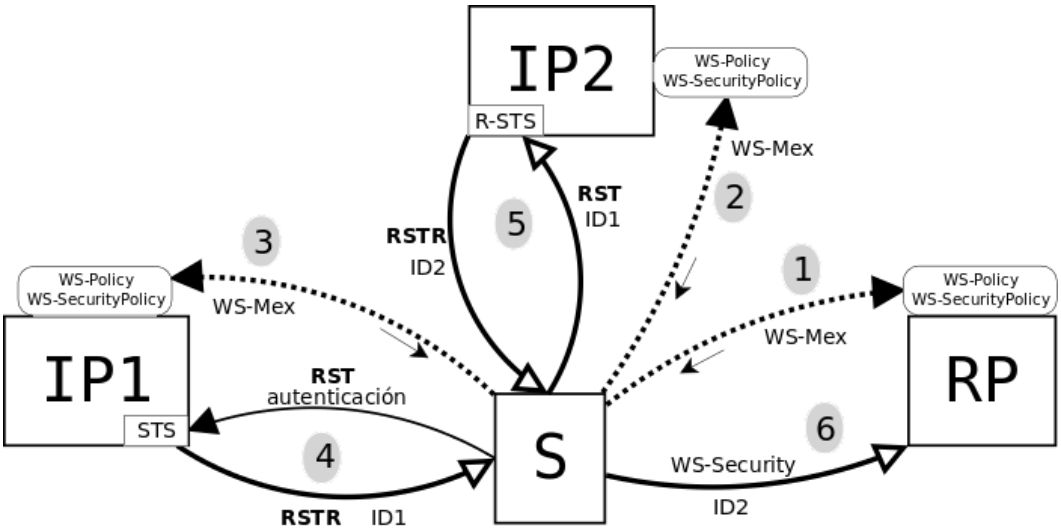

Figura 6.2: Flujo de mensajes WS-* en un escenario de confianza negociada

Veamos como queda el flujo de datos con servicios web:

1. S invoca a la RP. A través de WS-MetadataExchange (WS-Mex) adquiere un documento con la política (WS-Policy y WS-SecurityPolicy).
   Requisitos:

   - Una identidad digital emitida por IP2.
   - Los claims en el formato RP.

   S comprueba que no tiene ninguna identidad digital que se ajuste a lo pedido, con lo que se procede a inspeccionar los requisitos de IP2.

2. Negociación con IP2 para ver qué exige. Otra vez se obtienen los requisitos y la política a través de WS-MetadataExchange, la negociación es similar al caso anterior.
   Requisitos:

   - Una identidad emitida por IP1.
   - Los claims en el formato IP2 (que es igual que el que maneja IP1).

   S comprueba que tiene una identidad que satisface los requisitos de IP2.

3. Negociación con IP1 para ver cómo se invoca, obtiene los requisitos. Otra vez se obtienen los requisitos a través de WS-MetadataExchange. Ahora tenemos la política de invocación de IP1, con quien S mantiene una relación (es su proveedor de identidad).

4. S se autentica ante IP1 (petición o RST) y recibe una identidad digital ID1 (respuesta o RSTR). A continuación decide si se la presenta al IP2.

5. S presenta (RST) la identidad digital ID1 al IP2 (la asegura mediante WS-Security) y este le devuelve (RSTR) otra identidad digital ID2.
6. S presenta la identidad digital ID2 a la RP (otra vez la asegura con WS-Security) y gana acceso al servicio.

Consideraciones:

- La identidad digital va encapsulada como un token de WS-Security.
- La emisión de un token se define en WS-Trust.
- Los proveedores de identidad IP1 e IP2 ofrecen un servicio de STSs para emitir identidades.
- IP2 se considera un Resource STS o R-STS, y su única función es traducir claims. Sí, con WS-Trust y a través del STS que define, es posible hacer transformadores de claims.
- WS-Security permite que la comunicación esté protegida (recomiendo ojear otra vez los casos que se definen según los actores en WS-Federation). Si el caso fuera que la RP es una web a la que S accede con un navegador, el mensaje 6 sólo se podría asegurar en el transporte con HTTPS, no asegurarlo propiamente con WS-Security.
- WS-Trust garantiza que si hay un canal seguro entre las partes, el sistema será capaz de intercambiar información.
- La identidad que exige la RP va firmada por el IP2, una de las operaciones que permite WS-Security.

6.4. Conclusión

La idea de los servicios web es usar una forma universalmente entendible de describir datos y de definir cómo comunicarse con los programas. Esto se logra con especificaciones que se basan unas en las otras para ir añadiendo sucesivos niveles de complejidad según el modelo de datos. Como todo, esta solución tiene sus pros y sus contras.

- **Ventajas**
  - Interoperabilidad.
  - Basados en texto, funcionamiento y depuración sencilla.
  - Se transmiten por HTTP, a través del puerto 80. Puerto que casi nunca cierran los "firewalls".
  - La W3C gestiona los esquemas más usados, lo que los convierte en protocolos estándar y abiertos.
- **Inconvenientes**
  - Basados en texto (XML), poca optimización en la relación tamaño del mensaje/datos útiles y transmisión a través de un protocolo de aplicación (HTTP), lo que los hace más lentos en transmisión y procesado que otros sistemas como RMI, CORBA o DCOM. Aunque no es una característica deseable, es común a cualquier tunelización, pero con la potencia de cómputo actual, las ventajas superan a los invonvenientes.
  - Al pasar a través de los "firewall" puede ser difícil su contención al tener que recurrir a medios más específicos como un filtrado "proxy".

Capítulo 7

SAML

SAML (Security Assertion Markup Languaje) o Lenguaje por Marcas de Aserciones de Seguridad (traducción libre), desarrollado por el Comité Técnico de Servicios de Seguridad de OASIS, es un marco de trabajo basado en XML para transmitir información de usuario como autenticación, autorización o atributos. Como su nombre sugiere, SAML permite a las entidades de negocios hacer aserciones con respecto a la identidad, atributos y permisos de un sujeto (una entidad que suele ser un usuario humano) a otras entidades, como una empresa asociada u otra aplicación empresarial.

[OASIS, a]

La razón de ser de SAML es proveer de un entorno de comunicación para permitir hacer inicio de sesión único (SSO).

NOTA: *El primer método que se usó fue el de la cookie compartida, al iniciar sesión se cargaba una cookie en el navegador que era verificada por cada servicio, pero este truco no funciona cuando los servicios están en dominios distintos ya que no pueden acceder a las mismas cookies. Esto hizo que surgieran diferentes soluciones propietarias que dificultaban la interoperabilidad.*

7.1. Fundamentos

SAML está basado en las siguientes tecnologías.

7.1.1. XML

Ver capítulo 6.1.1

7.1.2. HTTP

Ver capítulo 6.1.2

7.1.3. SOAP

Ver capítulo 6.1.3

7.1.4. XML schema

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C) y alcanzó el nivel de recomendación en mayo de 2001.

[Wikipedia, n]

Recomiendo mirar los siguientes enlaces.

- W3C XML Schema.
  http://www.w3.org/XML/Schema
- XML Schema Tutorial.
  http://www.w3schools.com/Schema/default.asp
- DOs and DON'Ts.
  http://www.kohsuke.org/xmlschema/XMLSchemaDOsAndDONTs.html

7.1.5. XML Signature

También se conoce como XMLDsig, XML-Dsig o XML-Sig. Es una recomendación de la W3C para hacer firmas digitales mediante XML.

Puede ser de tres tipos.

1. Independiente: cuando lo que se firma no está en el documento XML.
2. Envuelta: si firma una parte del documento que la contiene.
3. Envolvente: si lo que se firma va incluído dentro de la propia firma.

La estructura de una firma XML es la siguiente.

```
<Signature>
  <SignedInfo>
    <SignatureMethod />
    <CanonicalizationMethod />
    <Reference>
      <Transforms>
      <DigestMethod>
      <DigestValue>
    </Reference>
    <Reference /> etc.
  </SignedInfo>
  <SignatureValue />
  <KeyInfo />
  <Object />
</Signature>
```

Una firma digital se construye de la siguiente manera. Se toma la información a firmar y se le aplica una función resumen (hash). El valor que devolvió esa función se cifra con la clave privada de quien lo quiere firmar. Es una forma rápida de garantizar la procedencia e integridad de una información. Útil para difusiones públicas: B.O.E., documentos oficiales, un comentario en un foro... cuando no importa la privacidad del contenido del mensaje o cuando el mensaje se transmite por un canal seguro.

La firma XML cumple estos principios pero tiene ciertas peculiaridades que hay que tener en cuenta por la forma en que se representa y transmite la información (al ser texto).

La información que queremos firmar, si va incluida en el documento, debe ser representable como texto plano. No se pueden incluir datos binarios, hay que codificarlos (normalmente se usa base64).

Antes de aplicar la función resumen a la información, es necesario canonicalizarla. Canonicalizar un XML es aplicarle un algoritmo para transformarlo en su forma canónica, equivalentemente lógica (con operaciones como: eliminar espacios y saltos de línea, ordenar etiquetas en el mismo nivel alfabéticamente, etc). Esto es necesario porque es posible que las diferentes plataformas por las que pase el documento XML no traten el texto de la misma manera (saltos de línea, codificación, etc), incluso que las aplicaciones que procesen el XML ordenen las etiquetas como más les convenga para su representación interna o utilicen diferentes prefijos para los esquemas (namespaces). Hay que recordar que el cambio de un solo bit en la información, propiciará el deseado efecto cascada en la función hash dando un valor completamente diferente e invalidando la firma. El algoritmo más utilizado para canonicalizar documentos XML es la canonicalización exclusiva C14N.

El principal problema de la canonicalización XML es su complejidad (el autor lo comprobó personalmente durante la realización de este proyecto) y la latencia asociada al pobre rendimiento del algoritmo [Wikipedia, m, Sec Criticisms].

Sobre la equivalencia lógica

Cualquier documento XML es parte de un conjunto de documentos XML que son equivalentemente lógicos en el contexto de aplicación pero que pueden variar en su representación física basada en los cambios sintácticos permitidos por XML1.0 y namespaces en XML.

[W3C, a]

Enlaces recomendados.

- W3C - XML Signature Syntax and Processing (Second Edition).
  http://www.w3.org/TR/xmldsig-core/
- RFC 4648 - The Base16, Base32, and Base64 Data Encodings.
  http://tools.ietf.org/html/rfc4648
- W3C - Canonical XML.
  http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- W3C - Exclusive XML Canonicalization (XML-C14N).
  http://www.w3.org/TR/xml-exc-c14n/
- Peter Gutmann - Why XML Security is Broken.
  http://www.cs.auckland.ac.nz/~pgut001/pubs/xmlsec.txt

7.1.6. XML Encryption

También se conoce como XML-Enc, es una recomendación de la W3C que define cómo encriptar el contenido de un elemento XML.

Esta especificación otorga cierta granularidad para elegir qué queremos cifrar, si un elemento, nodos hijo, contenido textual (datos arbitrarios), el documento entero, o hacer superencriptación.

Comparte características con la recomendación XML-Sig como la canonicalización de los datos a cifrar y la representación en base64 de los datos binarios.

Enlaces recomendados.

- W3C - XML Encryption Syntax and Processing.
  http://www.w3.org/TR/xmlenc-core/

- Wikipedia - Xml encryption (en castellano).
  http://es.wikipedia.org/wiki/Xml_encryption

- Ritter's Crypto Glossary and Dictionary of Technical Cryptography - Superencryption http://www.ciphersbyritter.com/GLOSSARY.HTM#Superencryption

7.2. Arquitectura

Podemos distinguir tres partes:

1. **Usuario**: es el que inicia todo el flujo de datos al querer acceder a un recurso.

2. **Parte Asertiva, Autoridad SAML o AP (Asserting Party)**: se encarga de emitir aserciones SAML sobre el usuario. Esta parte suele tomar el rol de proveedor de identidad (IP o IdP), así que según contexto, se nombrará de una forma u otra.

3. **Parte Confidente, Consumidor de Aserciones o RP (Relying Party)**: recibe aserciones SAML y las evalúa para autenticar y autorizar al usuario. Es necesario que tenga una relación de confianza con la AP para que acepte sus aserciones. Suele ser parte de un proveedor de servicios o SP por lo que es posible nombrarlo así.

7.3. Componentes

Veamos los componentes básicos de este marco de trabajo.

7.3.1. Aserciones

SAML permite (a una AP) emitir declaraciones que contienen información de seguridad. Estas declaraciones se denominan aserciones SAML. Una aserción contiene información referida a un sujeto (no siempre representado de forma explícita), las condiciones usadas para validar la aserción y declaraciones propias.
La figura 7.1 muestra un ejemplo de aserción SAML.

```
1: <saml:Assertion xmlns:saml=?urn:oasis:names:tc:SAML:2.0:assertion?
2:   Version="2.0"
3:   IssueInstant="2005-01-31T12:00:00Z">
4:   <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
5:     http://idp.example.org
6:   </saml:Issuer>
7:   <saml:Subject>
8:     <saml:NameID
9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10:         samuel.mh@gmail.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2009-01-31T12:00:00Z"
15:     NotOnOrAfter="2010-01-31T12:10:00Z">
16:   </saml:Conditions>
17:   <saml:AuthnStatement
18:     AuthnInstant="2009-01-31T12:00:00Z" SessionIndex="67775277772">
19:     <saml:AuthnContext>
20:       <saml:AuthnContextClassRef>
21:         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
22:       </saml:AuthnContextClassRef>
23:     </saml:AuthnContext>
24:   </saml:AuthnStatement>
25: </saml:Assertion>
```

Figura 7.1: Aserción con sujeto (mi correo), condiciones y una declaración de autenticación.

Declaraciones

Una aserción está compuesta por uno o más de los siguientes tipos de declaraciones:

- **Autenticación**: especifica el método y la fecha en la que se autenticó el usuario.
- **Atributos**: son atributos identificativos del usuario (nombre, correo, DNI...).
- **Decisiones de autorización**: definen lo que se le permite hacer al usuario, a qué recursos tiene aceso, etc.

7.3.2. Protocolos

SAML define los siguientes protocolos (u operaciones básicas) de petición/respuesta.

- **Petición de Autenticación**: se pide una aserción de autenticación. Se usa para establecer el inicio de sesión único SSO.
- **Fin de Sesión Único (Single Logout)**: es un mecanismo para cerrar todas las sesiones asociadas a la principal de una vez (lo contrario al SSO).
- **Consulta y petición de aserción**: define un conjunto de consultas de las que se obtendrán aserciones SAML. Una RP puede preguntar a una AP por una aserción referida a un usuario.
- **Resolución de Artefacto**: define un mecanismo por el que los mensajes SAML se pasarán por referencia usando un valor pequeño de longitud fija llamado artefacto. El receptor del artefacto usará este protocolo para preguntarle al emisor del mensaje por el mensaje en sí.
- **Gestión del nombre de identificador**: define un mecanismo para cambiar el valor o el formato del identificador de nombre para referirse a un usuario. Pueden usarlo tanto el SP como el IP. También permite terminar la asociación de un identificador entre las dos partes.
- **Correspondencia ("mapeo") del nombre de identificador**: permite asociar un identificador de nombre SAML a otro.

7.3.3. Envolturas o tunelizaciones ("Bindings")

Las envolturas detallan como se enviarán los mensajes SAML usando protocolos de transporte.

- **Redirección HTTP**: transporte sobre los mensajes de redirección HTTP (código de respuesta 302).
- **HTTP POST**: transporte sobre formularios HTTP codificados en base64.
- **Artefacto**: transporte de un artefacto a través de HTTP. O en la cabecera como parte de la URL o como parte del cuerpo en un formulario.
- **SAML SOAP**: transporte de mensajes usando SOAP.
- **SOAP inverso (PAOS)**: permite a un cliente HTTP ser un respondedor.
- **SAML URI**: permite obtener una aserción SAML resolviendo una URI.

7.3.4. Perfiles

Los perfiles SAML definen como se combinan las aserciones, protocolos y envolturas para adaptarlos a diferentes escearios.

- **SSO con navegador**: usa el protocolo de petición de autenticación, mensajes de respuesta SAML y aserciones para lograr un inicio de sesión único con navegadores web estándar. Define cómo se usan los mensajes en combinación con las envolturas de redirección HTTP, de HTTP POST y de artefacto.

- **Cliente y Proxy Mejorados o ECP (Enhanced Client and Proxy)**: define un perfil especializado de SSO en el que el cliente o el proxy pueden hacer envolturas de SOAP inverso.

- **Descubrimiento de Proveedores de Identidad**: mecanismo para que los proveedores de servicios aprendan sobre los IPs que ha visitado el usuario previamente.

- **Fin de Sesión Único**: cómo usar el protocolo de fin de sesión único con envolturas de SOAP, redirección HTTP, HTTP POST y artefacto.

- **Consulta y petición de aserción**: cómo usar el protocolo de consulta y petición de aserción.

- **Resolución de artefacto**: cómo usar el protocolo de artefacto con una envoltura síncrona, como SOAP, para obtener el mensaje.

- **Gestión del nombre de identificador**: cómo usar el protocolo de gestión del nombre de identificador con las envolturas.

- **Correspondencia ("mapeo") del nombre de identificador**: cómo usa el protocolo de correspondencia del nombre de identificador una envoltura síncrona como SOAP.

La figura 7.2 muestra cómo se anidan los componentes de SAML.

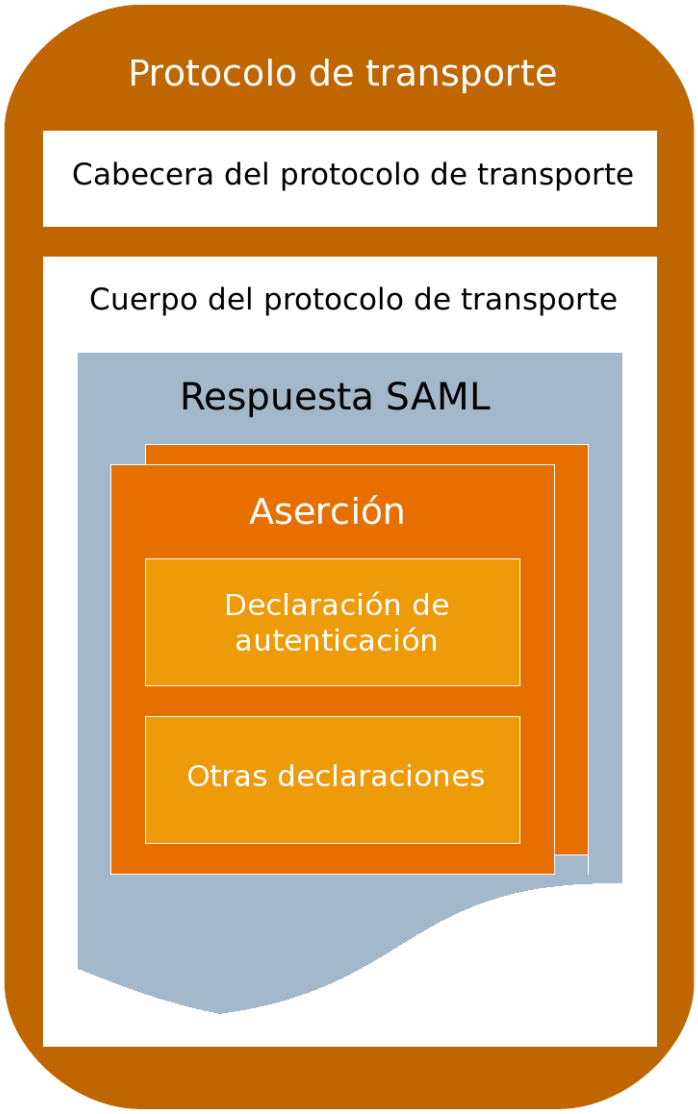

Figura 7.2: Estructura de un mensaje completo, anidamiento de componentes.

NOTA: *Se pueden encontrar ejemplos de mensajes SAML en la siguiente dirección.*
http://rnd.feide.no/samlexamples

7.4. Caso de uso

A continuación expongo el caso de uso en el que un usuario hará inicio de sesión único con su navegador web. El usuario accederá a la web (https://sp.ic/recurso) de un proveedor de servicios o SP gestionada por una RP que acepta SAML. El SP necesita saber saber la identidad del usuario antes de permitirle acceder al servicio. La figura 7.3 muestra el flujo de datos resultante.

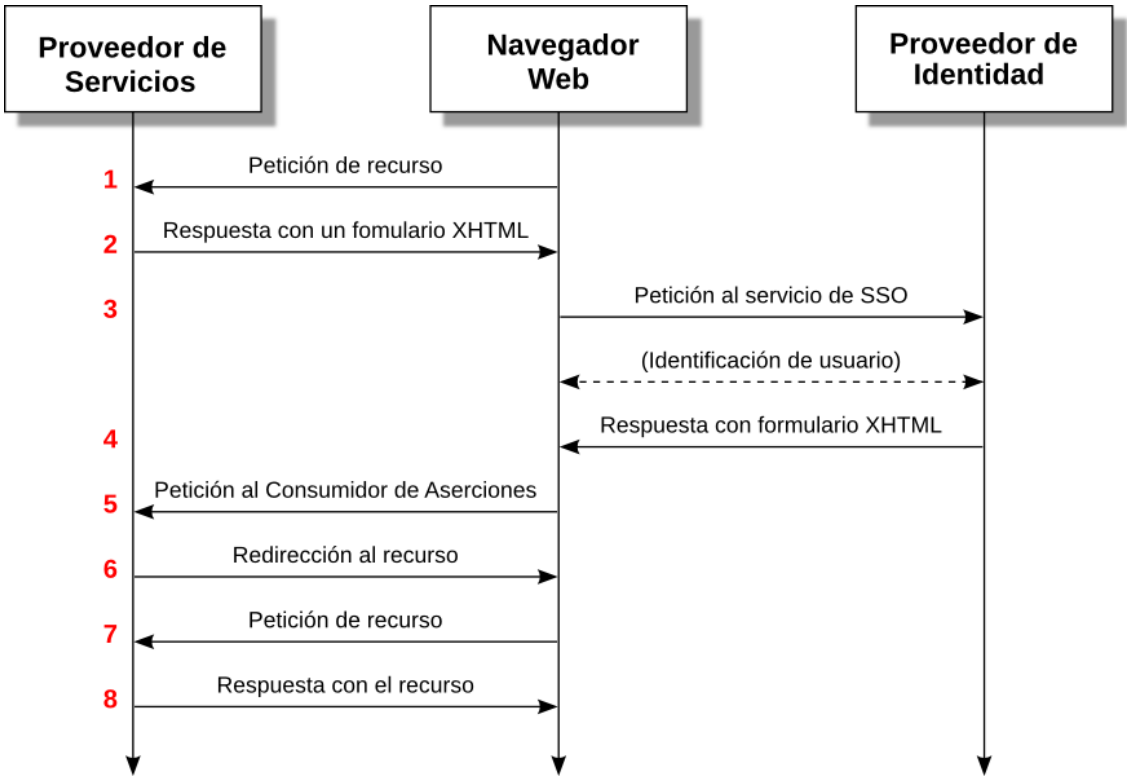

Figura 7.3: Flujo de mensajes al hacer SSO con SAML.

Mensajes.

1. **Petición de acceso a un recurso del SP.**
   El usuario pide a través de su navegador, acceder a un recurso protegido del proveedor de servicios (`https://sp.ic/recurso`). El SP reliza una comprobación de seguridad para ver si existe un contexto de seguridad (tener sesión), si así es pasa al punto número 8.

2. **Respuesta con un formulario XHTML.**
   El proveedor de servicios responde con un formulario que tendrá la siguiente forma.

```
<form method="post" action="https://idp.ic/SAML2/SSO/POST" ...>
  <input type="hidden" name="SAMLRequest" value="(aquí la petición SAML)" />
  ...
  <input type="submit" value="Submit" />
</form>
```

Es un formulario para el proveedor de identidad. El valor del campo SAMLRequest es la codificación en base64 de un elemento `<samlp:AuthnRequest>`.

3. **Petición de inicio de sesión única SSO al IP.**
   El navegador manda la petición SAML mediante el método POST del formulario anterior al proveedor de identidad. El servicio de SSO procesa el elemento AuthnRequest y hace las comprobaciones de seguridad pertinentes. Si el usuario no tiene un contexto de seguridad válido, el IP deberá autenticar al usuario con algún método.

4. **Respuesta con un formulario XHTML.**
   Una vez que el servicio de SSO ha validado la petición, responde con un formulario XHTML parecido al siguiente.

```
1  <form method="post" action="https://sp.ic/SAML2/SSO/POST" ...>
2     <input type="hidden" name="SAMLResponse" value="(aquí la respuesta SAML)" />
3     ...
4     <input type="submit" value="Submit" />
5  </form>
```

   El formulario es para el proveedor de servicios. El valor del campo SAMLResponse es la codificación en base64 de un elemento `<samlp:Response>`.

5. **Petición al consumidor de aserciones en el SP**
   El navegador manda la respuesta SAML mediante el método POST del formulario anterior al servicio del SP que consume aserciones.

6. **Redirección al recurso.**
   El consumidor de aserciones procesa la respuesta, crea un contexto de seguridad en el proveedor de servicios (algo como iniciar sesión) y redirige al usuario al recurso que pidió en un principio.

7. **Petición de acceso a un recurso del SP (por segunda vez).**
   El navegador es redirigido a un recurso protegido del proveedor de servicios (`https://sp.ic/recurso`). El SP realiza una comprobación de seguridad para ver si existe un contexto de seguridad y esta vez sí que existe.

8. **Respuesta con el recurso.**
   Como existe un contexto de seguridad, el proveedor de servicios permite al usuario acceder al recurso.

7.5. Web Services

7.5.1. Envoltura SOAP

En el ejemplo anterior hemos visto que los mensajes SAML se pasan por valor, se manda el mensaje entero, a través de formularios. La comunicación entre el IP y la RP es indirecta.

Existe otra forma de mandar esos mensajes, por referencia o mediante un artefacto (como vimos en la subsección de protocolos). El proveedor de identidad le daría al usuario un artefacto en vez de una aserción con su identidad, y este se lo presentaría a la RP. El consumidor de aserciones identificaría el artefacto y establecería una **conexión directa** con el proveedor de identidad para obtener la aserción con la identidad digital. Esta conexión sería a través de un servicio web usando la envoltura de SAML sobre SOAP (SAML sobre SOAP sobre HTTP).

SAML por si mismo no hace uso de la cabecera SOAP o de la envoltura SOAP, pero no impide a las aplicaciones basadas en SAML hacerlo si lo necesitan.

[OASIS, c, líneas 698-699]

7.5.2. WS-Security

Es posible transmitir aserciones SAML sin recurrir al protocolo SAML de Petición/respuesta o a los perfiles definidos en el la especificación. Por ejemplo, a través de los servicios web. Usaremos el esquema de WS-Security ya que provee de las funciones de autenticación, integridad de datos y confidencialidad.

WS-Security define un elemento `<Security>` que define cómo va protegido el mensaje. Irá incluído en la cabecera del mensaje SOAP. WS-Security hace uso de los mecanismos de firma y encriptación XML vistos al comienzo del capítulo para proteger tanto la información de la cabecera como la del cuerpo del mensaje SOAP. El elemento `<Security>` especifica qué operaciones se usaron y en que orden junto con sus claves, y también qué atributos e información está asociada. También se puede incluir la fecha.

En WS-Security hablamos del "security token" (ver 6.2.5), la forma en que se especifica el material criptográfico. Estos tokens pueden ser binarios (información en bruto) o una estructura XML (como una aserción SAML). Los tokenes XML se insertan como subelementos del elemento `<Security>`.

También vimos que existían especificaciones satélite o "token profiles"para manejar según el caso un tipo u otro de token. Existe uno de estos perfiles para manejar aserciones SAML, se puede encontrar en: http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

7.5.3. Diferencias

Hay que hacer una clara distinción entre el hecho de usar la envoltura SOAP para mandar mensajes SAML y usar servicios web para mandar aserciones SAML.

He aquí las diferencias más significativas.

Características del protocolo Petición/Respuesta de SAML sobre SOAP.

- Se usa para obtener aserciones SAML ajenas al intercambio de mensajes SOAP. SOAP no es más que una envoltura, poca cohesión. Las aserciones no protegen el mensaje.

- Las aserciones SAML van en una respuesta SAML que va en el cuerpo de un mensaje SOAP.

- Las aserciones SAML son emitidas por una autoridad de confianza, y pueden o no ir dirigidas a la parte que las pidió.

Características del envío de aserciones SAML con servicios web WS.

- Las aserciones se tranportan en un elemento <Security> en la cabecera de un mensaje SOAP.

- Normalmente las aserciones protegen el mensaje que las transporta, suelen contener una clave para firmar el cuerpo del mensaje.

- Aunque la aserción se refiere al emisor del mensaje SOAP, normalmente no la genera él, sino que las obtiene de alguna parte.

La figura 7.4 muestra dos esquemas, el de la izquierda es un mensaje SAML tunelizado sobre SOAP, y el segundo es una aserción SAML encapsulada en un mensaje de servicios web con WS-Security.

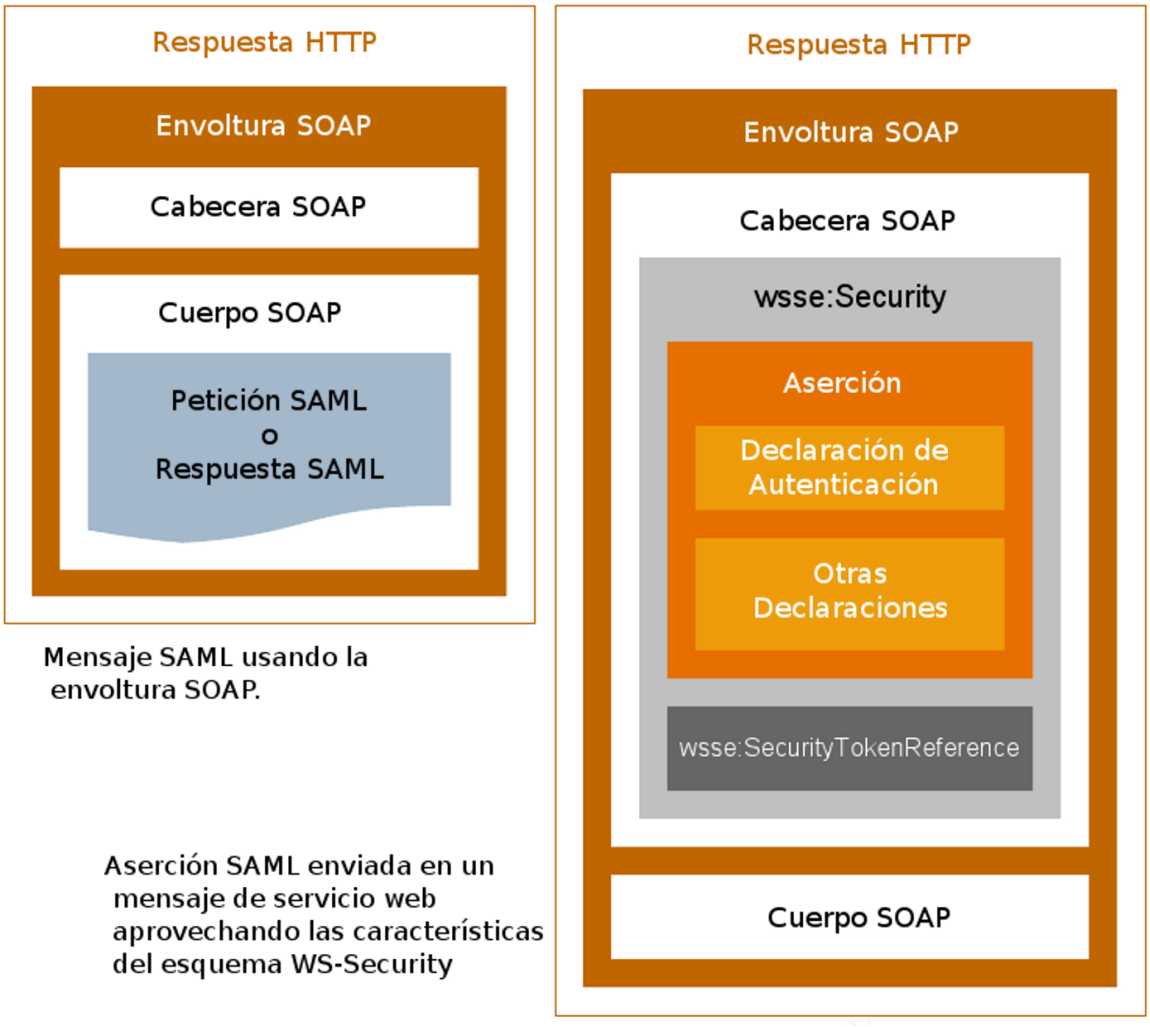

Figura 7.4: Diferencias entre envoltura SOAP y aserción protegida por WS-Security.

7.6. Herramienta: SimpleSAMLphp

SimpleSAMLphp es un a aplicación simple, escrita en PHP que solventa el problema de la autenticación. SimpleSAMLphp soporta varios escenarios de federación, mecanismos de autenticación y puede ser usada tanto para autenticación local, como proveedor de servicios SP o como un proveedor de identidad IP.

[Feide]

La figura 7.5 muestra el logotipo simpleSAMLphp.

Figura 7.5: Logotipo de simpleSAMLphp.

Esta herramienta permite crear un escenario de inicio de sesión único SSO, es de código abierto, gratuita y tiene una creciente comunidad de desarrolladores trabajando en ella (con lo que las dudas técnicas, peticiones y soluciones fluyen rápidamente en su lista de correo). Por todas estas cualidades fue seleccionada como herramienta para este proyecto. Ya tenemos gestor de aserciones SAML.

Recomiendo visitar la siguiente dirección si se va a introducir más a fondo en esta herramienta:

- Documentación de simpleSAMLphp
  http://rnd.feide.no/view/simplesamlphpdocs

7.7. Otras opciones

No sería justo obviar otras alternativas actuales de pleno vigor para el manejo de la identidad digital.

7.7.1. Shibboleth

El sistema Shibboleth®es un paquete software basado en estándares y de código abierto que permite establecer inicio de sesión único. Permite a los sitios tomar decisiones informadas sobre la autenticación de los accesos de usuario a recursos protegidos de una forma que preserva la privacidad.

[Internet2]

La figura 7.6 muestra el logotipo Shibboleth.

Figura 7.6: Logotipo de Shibboleth.

Shibboleth implementa estándares de identidad federada ampliamente usados como SAML para ofrecer un servicio de inicio de sesión único y un marco para el intercabio de atributos.

Recomiendo ojear los siguientes enlaces para profundizar en esta herramienta.

- Página principal de Shibboleth.
  http://en.wikipedia.org/wiki/Shibboleth_(Internet2)
- Página de la Wikipedia.
  http://shibboleth.internet2.edu/

7.7.2. OpenID

Es un sistema para autenticarse en servicios web haciendo uso de una cuenta maestra.

La figura 7.7 muestra el logotipo Shibboleth.

Figura 7.7: Logotipo de OpenID.

OpenID es un sistema de inicio de sesión único para Internet que permite al usuario controlar la situación. OpenID es una tecnología centrada en el usuario que permite a una persona tener el control de cómo se gestiona y usa su identidad digital. Al ser descentralizado no hay un servidor central en el que tengan que registrarse todo servicio y usuario. La gente puede elegir el proveedor de OpenID (es un proveedor de identidad) que desee.

[Powell and Recordon]

Su arquitectura consta de tres partes: el sujeto, un proveedor de identidad y una parte confiante.

Recurro a la explicación mediante caso práctico [Bar-or and Thomas].

Ejemplo(19)

- Un usuario quiere acceder a un recurso de la parte confiante, si no tiene una sesión, se le muestra un formulario para que introduzca su identificador de OpenID.
- Un indentificador OpenID es una URL o XRI que identifican al usuario dentro del IP (EJ: `http://samuelmh.fakeip.com`). En teoría deberían ser identificadores unidireccionales como vimos en la ley de Identidad dirigida (ver 4.4), pero se quedan a medio camino por la propia naturaleza del uso que se les tiene que dar.
- La RP analiza el identificador y redirige al usuario a la página de autenticación del dominio que gestiona su identificador.
- El usuario se autentica en la página del domino que gestiona su identidad. El método de autenticación puede ser cualquiera, el clásico es el par usuario-contraseña.
- El IP pide una confirmación al usuario de la información que va a enviar y a quién y si se acepta, envía a la RP la identidad a través del navegador usuario.
- La RP valida la identidad y da acceso al recurso que pidió el usuario.

La figura 7.8 muestra el proceso de autenticación con OpenID usando un identificador de Google [Google].

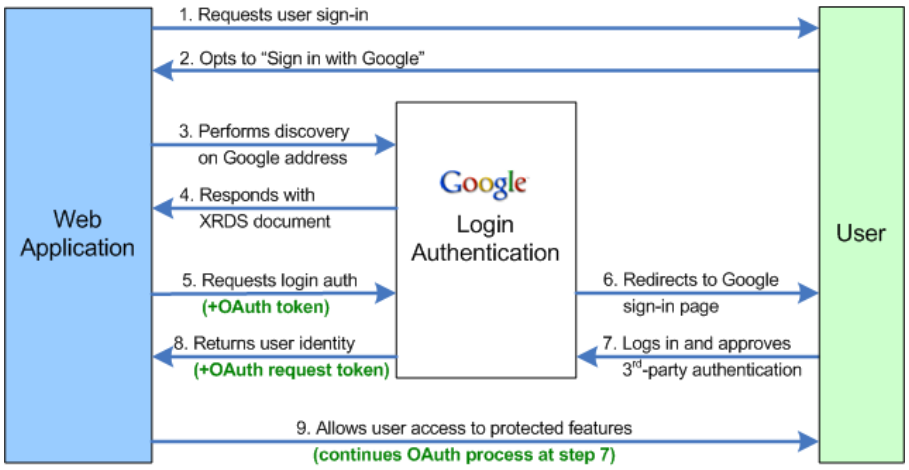

Figura 7.8: Proceso de autenticación con OpenID usando a Google como proveedor de identidad.

OpenID se sitúa como uno de los candidatos favoritos para establecerse como el nuevo sistema de autenticación en internet. Solventa el problema de la fatiga de contraseñas y el de inicio de sesión único de una forma elegante y sin suponer un trauma para el usuario, aunque sí un cambio de paradigma y suscite una cierta desconfianza.

Como principales contras, destaco el problema de la múltiple identidad que acarrea varias cuentas de OpenID para diferentes perfiles de navegación, el uso de identificadores no unidireccionales que se convierten en un dato identificativo y problemas de phishing [Wikipedia, h].

Sin embargo, aunque para muchos webmasters el sistema nos puede parecer muy sencillo, para muchos usuarios es bastante difícil de comprender que un sitio web pueda identificarles con su cuenta de Google sin sospechar que su password (y, por lo tanto, su seguridad) están en riesgo. Hay que recordar que si te haces con una cuenta de Google puedes conocer mucha información privada de una persona (sus correos, sus búsquedas, sus ingresos en AdSense, sus documentos de Google Docs, su número de tarjeta bancaria, o incluso los contenidos de su Disco Duro), y muchas personas serán reacias a identificarse con este nuevo sistema si previamente Google no hace una labor pedagógica para convencerles de la seguridad de este servicio.

[google.dirson.com]

Recomiendo mirar los siguientes enlaces.

- Presentación con caso de uso.
  http://middleware.internet2.edu/idtrust/2009/slides/01-fletcher-identity.pdf
- Página de la Wikipedia.
  http://en.wikipedia.org/wiki/Openid

7.8. Conclusión

Lo que hemos visto hasta ahora es una pequeña introducción al mundo de la federación, para completar, quiero destacar conceptos fundamentales que no han sido formulados explícitamente.

El protocolo SAML resuelve el problema del SSO, pero solo cuando existe una fuerte relación de confianza entre las partes. El problema es que la información fluye de forma directa o casi directa entre el proveedor de identidad y la parte confiante. En este sentido, el usuario no tiene el control absoluto de lo que está pasando ni de los datos que se van a transmitir. Esta cualidad hace que SAML sea una buena tecnología cuando estamos en un escenario con relaciones sólidas, flujos establecidos y políticas de datos: relaciones empresariales, académicas, FEDERACIÓN... Pero no una opción a tener en cuenta para la autenticación en internet (demasiada pluralidad y poca confianza).

El que haya múltiples proveedores de identidad con sus diferentes páginas de autenticación no soluciona el problema de la experiencia de usuario inconsistente. Añadiéndo a esto que las comunicaciones se realizan con el navegador usando canales seguros (HTTPS), en algún momento aparecen los certificados y volvemos al problema de la usabilidad. En pocas palabras, no hemos solucionado el problema del "phishing".

Hemos visto la potencia de SAML como token para poder expresar identidades digitales. Era la pieza que nos faltaba hasta ahora en el metasistema para materializar nuestra definición de identidad

digital (referencia).

También hemos visto que SAML se integra perfectamente con los servicios web, ahora la identidad puede fluir con total seguridad y confianza (la que nos ofrece WS-Security) entre las diferentes partes del metasistema.

Existen varias herramientas para lidiar con el problema del SSO y he hecho una elección razonada.

Como colofón, propongo los siguientes enlaces para profundizar en el tema.

- OASIS - Security Assertion Markup Language (SAML) V2.0 Technical.
  http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf
- OASIS - SSTC Wiki.
  http://wiki.oasis-open.org/security
- Wikipedia - Security Assertion Markup Language.
  http://en.wikipedia.org/wiki/Saml
- Feide - Anatomy of SAML Messages.
  http://rnd.feide.no/content/anatomy-saml-messages

Capítulo 8

Infocards

Hasta ahora hemos visto las bondades del metasistema con un enfoque estructural y técnico. Nos queda la otra parte. ¿Qué pasa con el usuario? ¿Cómo se resume toda esta complejidad en algo que se pueda utilizar?

La herramienta indiscutible a tener en todo ordenador de uso personal que esté conectado a Internet es el navegador web. A través de una página web el usuario accederá a la RP para autenticarse y hacer uso de los servicios. El principal problema de los navegadores web es que en el contexto del metasistema se consideran agentes pasivos, es decir, aunque pueden hacer peticiones HTTP (característica fundamental) éstas solo se realizan a petición del usuario. Dicho de otro modo, no implementan características para hacer uso de servicios web.

Volviendo a las leyes de la identidad (capítulo 4), quedan cuestiones parcialmente resueltas o no tratadas todavía. ¿Qué hay de la experiencia de usuario consistente? ¿Y de la integración humana? ¿Cómo va a consentir y controlar el usuario? ¿Identidad dirigida? Mayoritariamente son problemas que caen dentro de la usabilidad, eso que el ordenador hace "mágicamente" y nos abstrae de la complejidad inherente al sistema. A medida que aumentamos la seguridad, la obligación de que el usuario entienda detalles técnicos se hace más fuerte y es un problema. Como vimos en la sección de usabilidad (subsección 1.3.9), si ya es un problema para el usuario entender la capa de transporte TLS y los certificados SSL, el que tenga una idea de cómo funcionan los servicios web (por ejemplo WS-Trust o WS-Federation) para usarlos activamente roza la utopía.

Por último queda el tema de la gestión de la identidad digital, ¿cómo puedo crear una identidad digital y hacer uso de ella? Hemos visto cómo se puede representar usando una aserción SAML; pero para el usuario es otro problema de usabilidad, quizá convendría expresarlo de una forma más icónica.

La conclusión a la que se llegó es que hace falta un nuevo tipo de herramienta, programa o agente que solucione los dilemas anteriores. Un programa que transforme protocolos, aserciones y comunicaciones en algo que está ahí pero de lo que no tengo que preocuparme, que siempre funcione igual o con mínimas variaciones y que me permita gestionar mi identidad digital.

8.1. El selector de identidad

La herramienta que necesitamos se conoce como selector de identidad. Es un programa que abstrae al usuario de los problemas comentados anteriormente (manejo de los servicios web) y le ofrece una interfaz amigable para manejar su identidad digital.

Otra cualidad que tiene es que está contruido teniendo en cuenta las 7 leyes de la identidad digital, con lo que cumple con todos los requisitos deseables. Veamos a continuación las características que todo selector de identidad debe cumplir.

8.1.1. Características

Integración con el navegador

Este es el punto de partida. Al intentar acceder a un servicio, nos encontraremos con que la RP nos tiene que autenticar, debemos proporcionarle una identidad digital. Es en este momento cuando nuestro navegador web haría uso de una extensión (o plugin) para llamar al selector de identidad y transmitirle los requisitos de la RP. Una vez finalizado el trabajo del selector, debería mandarle de vuelta al navegador la identidad resultante para que se la mande a la RP y finalizar el proceso.

"Iconización" de la identidad digital

Es la cosificación de las identidades que tiene a disposición el usuario final. Veíamos en la última ley de la identidad, "Experiencia consistente a través de contextos" (sección 4.7), la necesidad de separar contextos (saber qué identidad queremos usar en qué caso concreto). El selector de identidad nos permite materializar en iconos las identidades a nuestra disposición. De esta forma la gestión de una identidad se reduce a cuatro operaciones.

- Importación: cargamos la identidad en el selector para futuros usos.
- Borrado: quitamos la identidad del selector, no aparecerá más.
- Uso: caso común en el que seleccionamos una identidad previamente cargada en el selector para autenticarnos con los "claims" que ofrece.
- Exportación: empaquetamos las identidades que usamos en una máquina para cargarlas en otra.

Comunicación a través de servicios web

Es una característica oculta al control del usuario (aunque siempre puede cancelar la comunicación) que permite establecer todas las comunicaciones necesarias con los demás elementos del metasistema. Comunicación con el proveedor de identidad de forma directa y con la RP de forma indirecta (a través del navegador).

Seguridad

Son características fundamentales, ya que no debemos olvidar que estamos tratando con un sistema de autenticación. La ventaja que disponemos es que al reducir el ámbito de incertidumbre al selector, podemos mostrar mensajes concisos para mejorar el control del usuario sobre la situación. En contraposición a esta situación están los navegadores web, formularios diversos para autenticarse y en algunos casos páginas no lícitas para robar información.

Manejo de certificados

Volvemos al problema de los certificados SSL para proteger las comunicaciones en la capa de trasporte. Un buen selector nos tendría que dejar gestionar los certificados (al estilo del navegador Firefox de Mozilla) o por lo menos facilitarnos la tarea. Parece imposible abstraernos de esta tecnología tan necesaria, con lo que el problema de usabilidad sigue pendiente. No obstante, diré en favor del selector de identidad que normalmente y por ser una herramienta instalada (se supone que bajo control del usuario) facilita la tarea del manejo de certificados accediendo a los almacenes del sistema operativo donde se guardan los certificados. Esto es que aunque tiene su propio almacén de certificados (nivel de aplicación), suele acceder a los que están disponible a nivel de sesión en la máquina. Parece que no soluciona nada pero nada más lejos de la realidad, si yo se manejar mis certificados a nivel de sistema con una herramienta a la que estoy acostumbrado, no tendré problemas para usar esta característica del selector.

Ejemplo(20) Suponemos que quiero pedir cita médica en el SESCAM (seguridad social de castilla la mancha). Este servicio usa un certificado de la FMNT que debería tener como Autoridad de certificación pero que no viene por defecto en los navegadores. Al abrir la web, mi navegador saltará diciendo que la conexión es insegura y que si deseo aceptar el certificado. Pongamos que requieren que me autentique con una idetidad digital, entonces en el formulario de entrada salta mi selector de identidad. ¿Tengo que volver a importar el certificado de la FMNT? La respuesta es que ¡NO! El selector busca el certificado en su propio almacén y al no encontrarlo recurre a los del sistema/aplicaciones de confianza y... lo encuentra. Al final el problema del manejo de certificados reside en la navegación, ya que si me tengo que autenticar ante una página ¡es porque ya la he visitado y me fío!

Autorizar página

La primera vez que nos autentiquemos ante una RP, el selector debería de preguntar si nos fiamos de esa página para mandarle información, ya que es la primera vez que lo hacemos y podría ser inseguro.

Desactivar página

También debería ser posible desactivar el uso del selector de identidad para las páginas que deseemos. Una web maliciosa podría hacer peticiones de identidad en el navegador de forma indiscriminada de forma que nos bloqueara el ordenador al invocar repetidamente al selector de identidad. También serviría para discriminar páginas en las que no nos queremos autenticar con esta tecnología o tener una lista negra de sitios fraudulentos.

Escritorio protegido

Cuando aparece la ventana del selector de identidad, todo el escritorio se bloquea hasta que hemos terminado con la aplicación. Esto permite focalizar la atención del usuario en la tarea requerida y evitar que deje la aplicación en segundo plano y abandone la máquina (de forma inconsciente). También protege contra ataques al navegador web en los que podría a parecer una ventanita de pop-up simulando ser el selector y pidiendo credenciales (ataque de phishing).

Inicio protegido por contraseña

Al arrancar el selector la primera vez en la sesión de usuario en la máquina, se le pedirá una contraseña para dejarle usarlo. Puede parecer una característica menor y sin importancia, pero las identidades digitales podrían estar cifradas con esa contraseña en el almacén del selector, de esta forma solo podría acceder a ellas su legítimo dueño.

Experiencia consistente

En esta parte veremos todo lo que se ha avanzado en la integración del usuario. Hace tiempo se vio que sistemas especialmente complejos no tienen por qué ser más seguros que otros más simples pero con usuarios que saben manejarlos bien. Al final el punto más débil suele estar entre la máquina y la silla.

Secuencia predecible

Al depender la autenticación de un programa, podemos uniformar todos los procesos de autenticación para que sean iguales. He aquí los pasos fundamentales del caso más extenso y genérico.

- RP pide identidad
- Salta el selector de identidad.
- El usuario selecciona una identidad.
- El proveedor de identidad pide los credenciales.
- El usuario introduce los credenciales.
- El selector informa de la información que enviará a la RP.
- El usuario consiente o cancela el envío.
- Si consiente, el selector manda los datos al navegador, este a la RP y esta le autentica.
- El usuario puede acceder al servicio protegido.

La ventaja de esta uniformidad es que al ser predecible, si algo no sale como se espera, es porque algo va mal. Los formularios que se le presentan al usuario (selección de identidad, credenciales) siempre serán iguales, botones en el mismo sitio, ventanas con las mismas características, etc. La sensación de seguridad provocada por la poca incertidumbre del proceso hace que el usuario acepte antes el uso del selector.

El usuario está informado, integración humana

Esta característica es un compendio de cosas ya vistas y otras que añado.

- Iconificación de la identidad. El usuario sabe con certeza qué identidad está usando.
- Autorizar/desactivar páginas. Si puede mandarle una identidad a una página es porque ha autorizado su uso. Recordemos que tendemos a relajarnos con el tiempo o con la rutina. De esta forma solo tenemos que estar alerta la primera vez.
- Ante cualquier error inesperado, por ejemplo que no tenga un certificado, se le puede presentar una pantalla de ayuda personalizada para que lo solucione.
- Puede tener un registro de las páginas autorizadas, los certificados que usa así como sus políticas de privacidad y las identidades que ha presentado.
- Cuando se presenta la pantalla de selección de identidad, solo se muestran aquellas identidades que coinciden con los requisitos de la RP.
- Antes de enviar una identidad digital se muestra la información que contiene.
- En cualquier momento el usuario puede consultar la información que contiene una identidad digital.

El usuario consiente

Característica indispensable. Al final el usuario es el último responsable del envío de sus datos. Si el usuario no acepta el envío de información a la RP no hay nada que hacer. Esta característica se deriva de un usuario informado. Precisamente lo que se intenta es acabar con el Siguiente-¿Siguiente-¿Siguiente-¿...-¿Siguiente-¿Aceptar ¡vaya, quería cancelar! de muchos sistemas y reducir el proceso a los pasos indispensables.

Efectos secundarios

De usuarios conscientes de su privacidad, informados y con herramientas adecuadas no puede salir sino un sistema más seguro. Gracias a todas las características anteriores, se derivan otras más específicas.

¿Robo de identidad o Phishing? Problema que he mencionado anteriormente (1.2.1). Con el selector de identidad no está solucionado del todo (es imposible garantizar la seguridad absoluta de un sistema complejo) pero se mitiga el efecto en mayor medida que con el navegador web. Veamos por qué.

- **Manejo de certificados SSL.**

  En la RP, si no tenemos el certificado de una RP nos aparecerá como sitio inseguro.

  En el IP, si intentan que importemos una identidad falsa para robarnos los credenciales de la verdadera, al intentar importarla el selector nos alarmará con que la identidad falsa procede de un emisor no fiable.

- **Autorizar a una página.**
  Ante un ataque de phishing, la página de la RP nos aparecerá como nueva y el selector nos pedirá autorización. Se supone que si es una página de uso habitual (no es la primera vez que entramos) el selector no pide autorización ya que ¡sólo lo hace la primera vez!. Esta gran inconsistencia debería de ser suficiente para poner al usuario en alerta y que verificara la dirección del estafador y el certificado para asegurarse.

- **Falsear pantalla.**
  Hemos visto la característica del escritorio protegido, pero sería posible que algún maleante colara en la máquina de la víctima un pseudoselector de identidad para recopilar credenciales. Ante esto solo nos puede proteger nuestra reacción ante... sí, ante la incosistencia. Cuando se abriera el programa falso, probablemente no tendrá nuestras identidades, aquellas que tenemos importadas en el que solemos usar, esto significa que algo va mal. ¡Alerta, inconsistencia!

Privacidad

"El usuario consiente y controla" es la primera ley de la identidad y diría que la más importante para el usuario. Al ser conscientes de con quién tratamos y qué información vamos a dar, somos más celosos con nuestros datos, más aún si sabemos qué fines potenciales pueden tener.

Se estima que entre el 89 % y el 91 % del correo que circula por Internet es SPAM [MAAWWG, 2009, pág 2], correo basura con fines publicitarios, estafas y demás morralla que no interesa.

Imaginemos que creamos una identidad falsa para navegar de forma anónima pero autenticada, es decir, a los proveedores de servicios no les interesa quiénes somos en realidad, sino que siempre seamos los mismos físicamente. Con esta identidad tendríamos asociada una cuenta de correo con el único fin

de registrarnos en sitios.

Podríamos hacer uso de esa identidad para rellenar formularios que usarán los datos con fines dudosos (qué más da, son datos falsos y una cuenta de correo que usaremos para este fin) y acceder así a los servicios.

Continuando con las identidades, presento el caso contrario. Quiero acceder a la banca online y mi banco me expide una identidad con este fin, evidentemente la seguridad se ha incrementado para mi. Sé qué identidad debo elegir para acceder al banco, el icono con el logo de mi banco. Ya estoy haciendo distinciones según el contexto y la seguridad que necesito. No es lo mismo ser el "troll" macarra y anónimo de un foro que un respetable inversor de la bolsa online.

8.1.2. Programas disponibles

Recopilación de los selectores de identidad que hay disponibles.

Windows CardSpace

Es el selector de identidad de la empresa Microsoft para su sistema operativo Windows. La primera versión estuvo disponible para Windows XP con el marco de trabajo .NET 3.0 y ya en Windows Vista y Windows 7 viene incluido por defecto.

En la figura 8.1 se puede ver una captura de este programa en pleno proceso de selección de identidad.

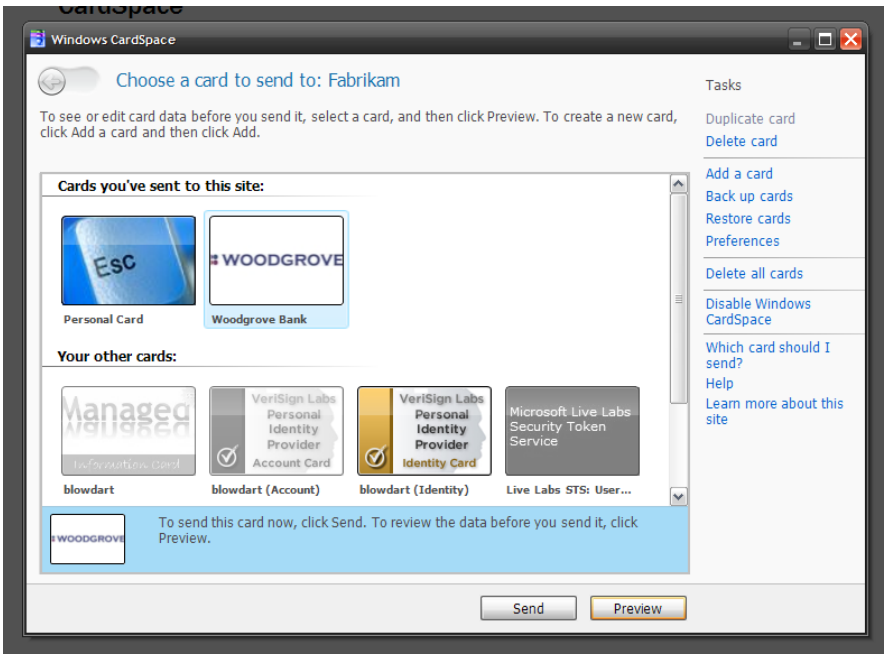

Figura 8.1: Windows CardSpace.

Recomiendo mirar los siguientes enlaces:

- Página oficial del programa.
  http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx

- Blog del equipo Geneva de Microsoft sobre identidad federada y el metasistema de identidad.
  http://blogs.msdn.com/card/

DigitalMe

Es la alternativa de código abierto a Windows CardSpace. Se puede utilizar en los sistemas operativos GNU/Linux (pieza fundamental de este proyecto) y Mac OS X. Potencialmente es posible utilizarlo en cualquier plataforma que disponga de las herramientas necesarias para su compilación.

En la figura 8.2 se puede ver una captura de este programa en pleno proceso de selección de identidad.

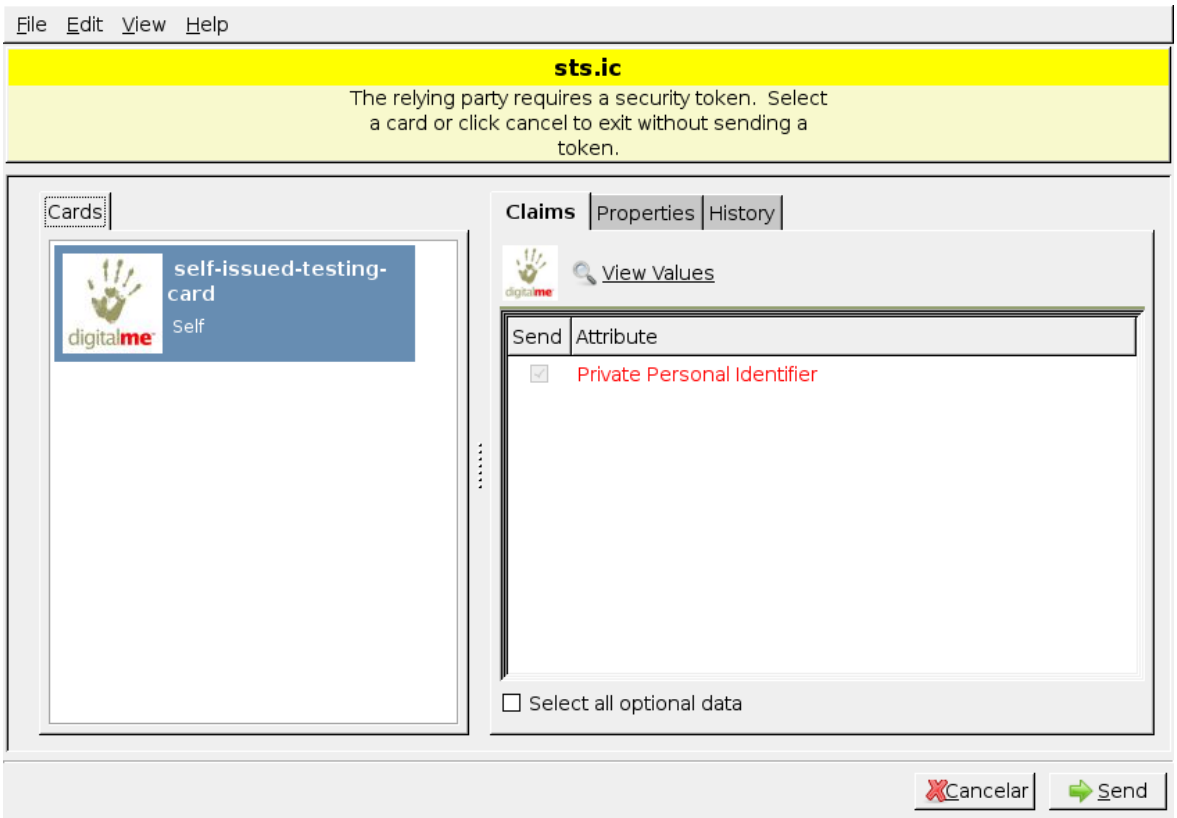

Figura 8.2: DigitalMe.

La página principal de este proyecto es:
http://code.bandit-project.org/trac/wiki/DigitalMe

openinfocard

Openinfocard es un proyecto bajo licencia BSD que agrupa un selector de identidad y código en Java para construir RPs,STSs, generar tarjetas y ejemplos diversos.

El selector de identidad de este proyecto tiene la peculiaridad de que es una extensión para el popular navegador web de Mozilla, Firefox. Esta cualidad hace que el selector de identidad sea multiplataforma (si funciona firefox en ese sistema, funcionará el selector).

La prueba que he realizado con este selector ha sido insatisfactoria. A pesar de ser multiplataforma, gran ventaja, tiene el inconveniente de que es un código bastante inmaduro. Lo he conseguido instalar en mi navegador, pero no hay forma de crear identidades digitales o de importarlas. El problema al ser un complemento es que una actualización del navegador implica cambios en el selector.

La figura 8.3 muestra una captura del proceso de selección de identidad.

Figura 8.3: Plugin openinfocard para Mozilla Firefox.

La página principal de este proyecto es:
http://code.google.com/p/openinfocard/

Azigo

Es un selector de identidad para los sistemas operativos Windows y Mac OS X. En su propia página web no dejan bien claro la finalidad del producto (más allá que la que se le supone a un selector de identidad). Indagando un poco más vemos dónde está el negocio.

El pseudónimo para las identidades que usa azigo es I-Cards, algo alejado del comunmente usado, infocards.
En la portada de su página se ven dos puntos extraños.

> *Leverage the information you've poured into various on-line sites when you are on other sites, like see when your memberships such as AAA earn you discounts, when you are on other sites like Google.*
> *Personalize Azigo with I-Cards that reflect your memberships and interests.*
>
> [Azigo]

Indagando un poco sobre las I-Cards llego a la siguiente web: http://www.azigo.com/icards.html. Se ve qué tipo de entidades emiten dichas tarjetas: empresas de autenticación (como Privo, BBB o Equifax), empresas de compras (Ebates), de servicios (AAA), museos, bibliotecas, asociaciones de comerciantes...
Es decir, el beneficio de Azigo proviene de la federación de servicios.

Ateniéndonos al primer punto de la cita, parece que este selector permite compartir información entre varias entidades que no guardan relación relación entre sí, es decir, están reconstruyendo nuestra identidad. Obviamente esto parece una violación de la ley "Identidad dirigida". Parece que la sombra de Gator sigue presente (http://en.wikipedia.org/wiki/Claria_Corporation#Gator).

¿Qué motivo tendría entonces un usuario para reemplazar su selector de identidad por Azigo? En la página http://www.azigo.com/company/2009/12/azigo-pays/ podemos leer la experiencia de un empleado de la compañía al que Ebates le pagó 161 USD por usar dicho selector.

En conclusión, Azigo es un selector de identidad comercial, con un entorno de socios en los que se ofrece ventajas por el uso de esta herramienta. Aun sin haber indagado mucho, supongo que esto es una violación a la privacidad (no tiene que ser malo siempre que el usuario consienta y se respete cierto anonimato en los estudios de mercado), pero la finalidad de esta sección no es denunciar las prácticas de Azigo, sino exponer un modelo de negocio real sobre el metasistema. Tarea del lector será evaluar la bondad de dicha opción.

La figura 8.4 muestra una captura de pantalla de este selector. Se puede apreciar que visualmente es mucho más atractivo que los anteriores, característica indispensable para ganarse al gran público, que entre por los ojos.

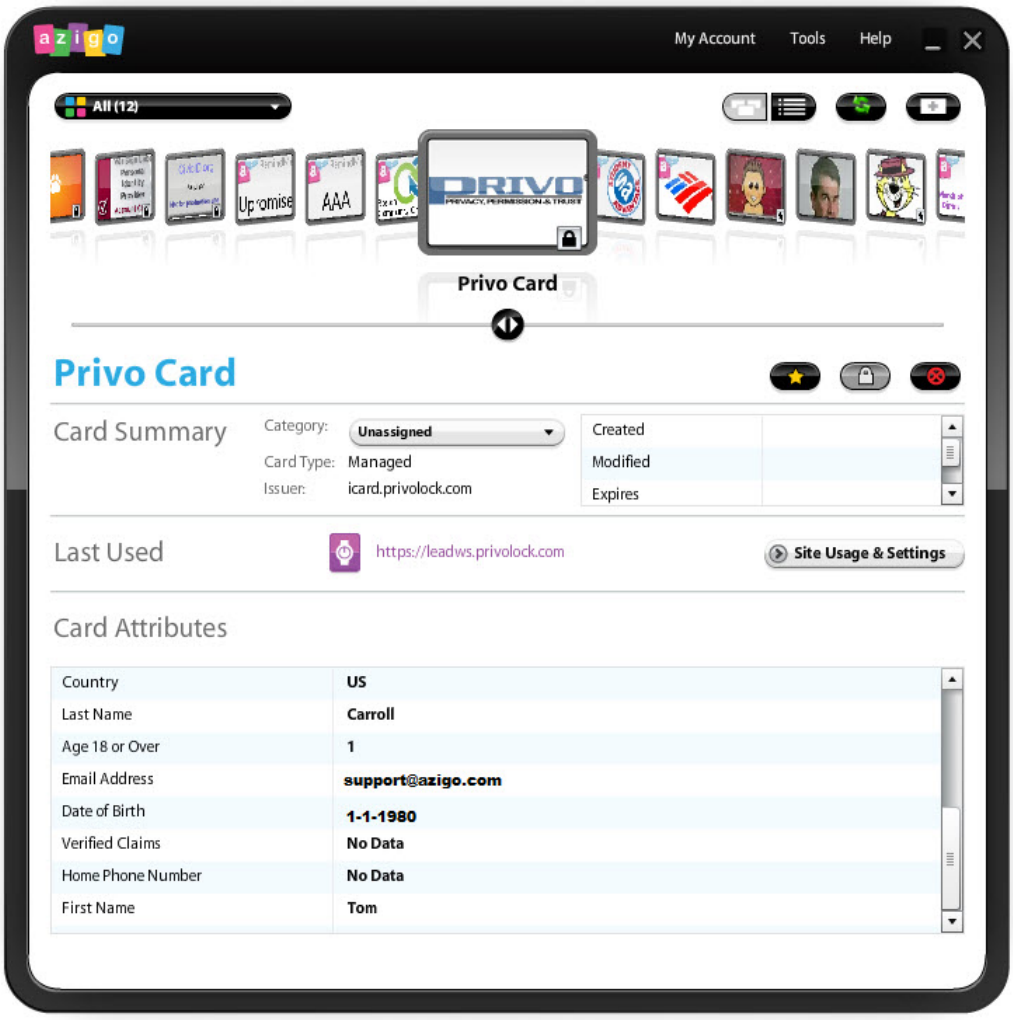

Figura 8.4: Azigo.

Para más información recomiendo consultar los siguientes enlaces:

- Página oficial de Azigo
  http://www.azigo.com/

- Vídeo en el que se muestra el funcionamiento de este selector de identidad.
  http://www.youtube.com/watch?v=D4HUZ5IhLa4

8.2. Information Cards

8.2.1. Tecnología

Llegados a este punto, concluyo la explicación teórica sobre el metasistema hablando de las "Information Cards" (tarjetas de información) también conocidas como infocards. Information Cards es el nombre con el que se conoce a la materialización de todas estas ideas sobre el metasistema. Es una serie de recomendaciones desarrolladas por Microsoft y basadas en estándares abiertos (pluralidad de operadores) que está avalada por organizaciones con tanto prestigio como OASIS. De hecho está a punto de convertirse en un estándar de facto [Wikipedia, d] ya que resuelve de manera elegante el problema de implementar una capa de identidad en Internet.

La figura 8.5 muestra el logotipo de Information Cards. Se usará tanto como icono para representar un fichero que contiene una de estas tarjetas como en formularios web para advertir de que se puede usar esta tecnología.

Figura 8.5: Logotipo de una Information Card.

8.2.2. Identidad

Hemos visto que en el selector de identidad, una identidad digital es un icono que podemos seleccionar. Microsoft hizo la analogía de identidad digital igual a tarjeta de presentación, es decir, yo me autentico en los servicios con tarjetas virtuales (una forma de materialización icónica para no expertos) y mi selector de identidad es un gestor de tarjetas virtuales. Ahora bien, ¿qué es una infocard? ¡una tarjeta virtual que dice quién soy!

Para el usuario, una infocard no tiene el sentido anterior de tecnología, sino que sería sinónimo de una identidad digital que puede usar. Este símil con la realidad de "alguien me pide identificación y le enseño una tarjeta" hace mucho en favor de la usabilidad y entendimiento del sistema ya que es una situación que todos hemos vivido en algún momento.

8.2.3. Tipos

Según el emisor podemos distinguir dos tipos de infocards.

Gestionadas

Son identidades generadas una tercera parte que tiene el rol de proveedor de identidad. Se usan en

escenarios en los que la RP se fía de un IP. Ejemplos de estas identidades son los pasaportes, D.N.I. y cualquier tipo de carnet emitido por una entidad (biblioteca, club, gimnasio, etc).

La figura 8.6 muestra el selector de Microsoft importando una tarjeta de este tipo.

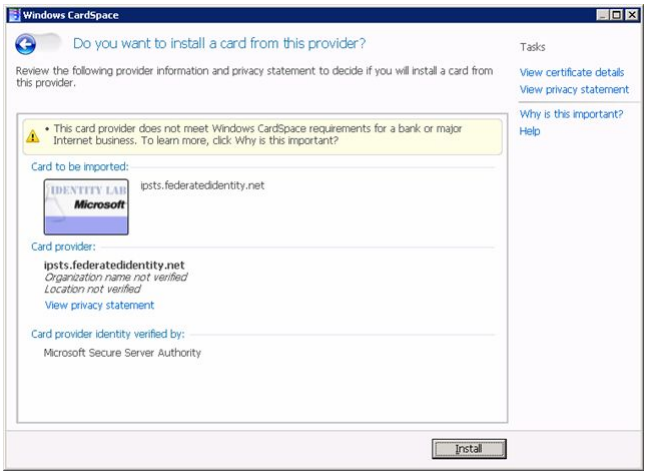

Figura 8.6: CardSpace importando una tarjeta gestionada.

Autoemitidas

Son identidades generadas por uno mismo. Útiles para rellenar formularios (envíos de correo, preferencias y demás datos que no hacen falta que sean verídicos desde el punto de vista del proveedor del servicio) y para autenticarse en servicios que pueden requerir de anonimato pero asegurarse que el usuario es siempre el mismo.

La figura 8.7 muestra el selector de Microsoft generando una identidad autoemitida.

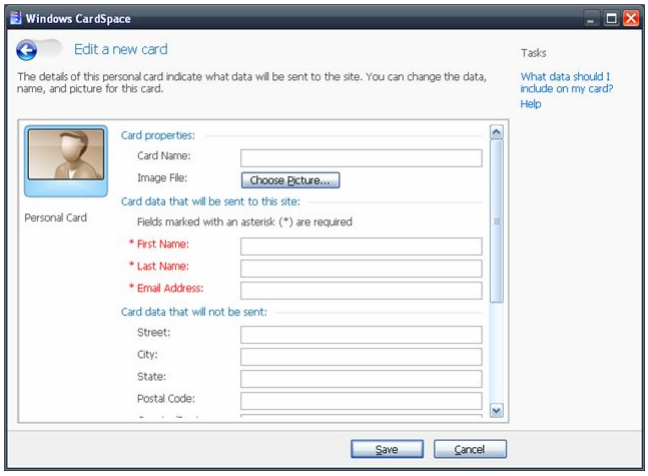

Figura 8.7: CardSpace generando una tarjeta autoemitida.

8.2.4. Estructura del fichero

Una Information Card se representa como un documento XML firmado que es emitido por un proveedor de identidad.

[3.1.1 OAS, 2009, Pág 17]

El selector de identidad tiene la opción de crear infocards autoemitidas. ¿Pero qué pasa con las gestionadas? Un proveedor podría emitirnos una infocard, un fichero que descargamos en nuestro ordenador y abrimos con el selector de identidad. Si abriésemos ese fichero con un editor de texto, veríamos que es un XML con una estructura parecida al de la figura 8.8.

```
<Object>
   <InformationCard>
      <InformationCardReference>...</InformationCardReference>
      <CardName>...</CardName>
      <CardImage>...</CardImage>
      <Issuer>...</Issuer>
      <TimeIssued>...</TimeIssued>
      <TimeExpires>...</TimeExpires
      <TokenServiceList>...</TokenServiceList>
      <SupportedTokenTypeList>
         <TokenType>...</TokenType>
      </SupportedTokenTypeList>
      <SupportedClaimTypeList>
         <SupportedClaimType Uri="...">
            <DisplayTag>...</DisplayTag>
            <Description>...</Description>
         </SupportedClaimType>
      </SupportedClaimTypeList>
      <PrivacyNotice>...</PrivacyNotice>
   </InformationCard>
</Object>
```

Figura 8.8: Estructura genérica de una infocard.

Campos

Veamos cuál es el significado de cada una de las etiquetas XML.

- **InformationCardReference:** información para identificar la tarjeta.
- **CardName:** es el nombre de tarjeta que mostará el selector de identidad. Es el único valor que puede modificar posteriormente el usuario.
- **CardImage:** es la imagen (codificada en base64) que mostrará el selector de identidad asociada a la tarjeta.
- **Issuer:** es una URI que representa al emisor de la tarjeta (normalmente el proveedor de identidad). Cuando la RP pide una identidad emitida por un cierto IP, es este valor el que usa el selector para mostrar la tarjeta.

- **TimeIssued:** fecha de cuando se emitió la tarjeta.
- **TimeExpires:** representa la fecha de expiración de la tarjeta.
- **TokenServiceList:** información sobre el IP (podrían ser más, por si el IP primario se cae) y su servicio de STS encargado de emitir identidades digitales en forma de token.
- **TokenType:** tipos de token en con los que se expresará la identidad digital, lo normal es una aserción SAML versión 1.
- **SupportedClaimTypeList:** lista de claims que puede contener nuestra identidad digital.
- **PrivacyNotice:** URL con la política de privacidad del IP.

NOTA: *Los tiempos de emisión y expiración son puramente ornamentales. Depende del selector de identidad el que se realice alguna acción o no. Por ejemplo, Windows CardSpace los muestra pero no hace nada si están fuera del rango razonable de uso.*

Desglosemos algunos campos interesantes.

InformationCardReference

En la figura 8.9 se muestan los elementos de este campo.

```
<InformationCardReference>
   <CardID>...</CardID>
   <CardVersion>...</CardVersion>
</InformationCardReference>
```

Figura 8.9: Estructura del campo InformationCardReference.

Veamos cuál es el significado de cada una de las etiquetas XML.

- **CardId:** es el identificador único de la tarjeta, no puede haber dos o más tarjetas en el selector con el mismo valor de CardId (ni siquiera deberían existir dos tarjetas en el mundo con el mismo CardId). Si al importar una tarjeta existiese otra con este mismo valor, el selector preguntaría al usuario si desea reemplazarla al considerarlas la misma tarjeta.
- **CardVersion:** se usa para identificar la versión de la tarjeta, por ejemplo cuando se importa una tarjeta con el mismo CardId, así se disguingue cuál es más nueva.

TokenServiceList

En la figura 8.10 se muestan los elementos de este campo.

Veamos cuál es el significado de cada una de las etiquetas XML.

- **TokenService:** STS a utilizar, en caso de haber más de uno, el selector elegirá siempre el primero y continuará en orden.
- **EndpointReference:** direcciones del servicio
- **Address:** URL del STS del IP

```
<TokenServiceList>
   <TokenService>

      <EndpointReference>
         <Address>...</Address>
         <Metadata>
            <Metadata>
               <MetadataSection>
                  <MetadataReference>
                     <Address>...</Address>
                  </MetadataReference>
               </MetadataSection>
            </Metadata>
         </Metadata>
      </EndpointReference>

      <UserCredential>
         <DisplayCredentialHint>...</DisplayCredentialHint>
         ...Método de autenticación...
      </UserCredential>

   </TokenService>
</TokenServiceList>
```

Figura 8.10: Estructura del campo TokenServiceList.

- **Address (dentro de Metadata):** URL de la página que contiene la información del servicio de STS en un XML con WS-MetadataExchange.
- **UserCredential:** define el método de autenticación en el IP. Se verá más adelante.
- **DisplayCredentialHint:** frase para ayudar al usuario a entender qué hacer con ese método de autenticación. Si tuvieramos un método de autenticación basado en usuario/contraseña pondríamos: "por favor, introduzca su usuario y contraseña".

8.2.5. Identificadores unidireccionales

Un campo de la tarjeta es el CardId, si la RP usara este atributo para identificar al usuario de manera unívoca, tendríamos un problema. Este identificador sería el mismo para todas las RPs y por consiguiente va en contra de de ley de "Identidad dirigida" (subsección 4.4). El identificador que usa el usuario en la RP debe ser unidireccional (uno diferente para cada RP).

Contamos con dos variables, el CardId y la RP (su identificador omnidireccional), para establecer un identificador que relacione esa infocard con el servicio. Esta tarea la lleva a cabo el selector de identidad generando un claim llamado PPID (private personal identifier) o identificador privado y personal. La figura 8.11 muestra el concepto que estamos tratando.
El funcionamiento es el siguiente: el CardId es un valor fijo de la tarjeta y cuando una RP nos pide el identificador de esa tarjeta, el selector de identidad toma el valor del CardId, el identificador omnidireccional de la RP y genera el PPID como claim de la identidad que mandaremos a la RP.

Figura 8.11: Cálculo del PPID.

El claim PPID de un usuario representa un identificador único para ese usuario en una RP que es diferente de cualquier identificador que pueda usar en otra RP.

[Nanda and Jones, 2008, pág 53]

NOTA: *El identificador ominidireccional de una RP suele ser su certificado SSL, en el caso de que no disponga de tal, se utiliza la URL del sitio.*

Para aprender a calcular el PPID recomiendo mirar el siguiente documento, el punto 8.6 "The PPID Claim".

- Identity Selector Interoperability Profile V1.5
  http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5.pdf

8.2.6. Métodos de autenticación

Cuando un usuario elige una infocard gestionada, recurre a un proveedor de identidad para que le emita una identidad. ¿Cómo reconoce el IP que el usuario es el legítimo propietario? ¿Sirve acaso como credencial el poseer el CardId o lo que es lo mismo, poseer la tarjeta? La respuesta es no, necesitamos un método de autenticación. Recordemos que la tecnología de Information Cards está orientada a la gestión de la identidad y a solucionar problemas de usabilidad, no a proponer nuevos métodos de autenticación.

En los escritos oficiales [ver punto 5 Nanda and Jones, 2008, págs 31-33] se establecen cuatro métodos de autenticación, no obstante ya hemos visto que sería fácil añadir cualquier otro método que se nos ocurriera debido a la modularidad de XML. El mayor problema estaría en modificar el selector de identidad, de ahí la necesidad del código abierto.
Comento los métodos de autenticación estándar.

Usuario/contraseña

El clásico de toda la vida, el selector de identidad le pedirá un nombre de usuario y una contraseña para poder autenticarse en el IP.

Certificados X.509

Para escenarios que requieran medidas de seguridad más robustas. El selector de identidad pedirá un certificado x.509. Estos certificados pueden estar en dispositivos físicos, como el DNI electrónico, o en el propio ordenador, como los que expide la Fábrica Nacional de Moneda y Timbre para realizar trámites en el catastro, padrón y demás organismos oficiales.

Kerberos

Para autenticación con Kerberos. No entro en detalle.

Tarjeta autoemitida

Este caso es el pilar fundamental de este proyecto, recomiendo al lector que preste especial atención.

Consiste en autenticarse ante el IP con una infocard autoemitida. Esto implica que para conseguir la infocard gestionada del IP, previamente debe conocer la infocard autoemitida que usaremos como credencial.
Un problema añadido es que si cambiamos de máquina, deberemos llevarnos las dos tarjetas o generar una nueva tarjeta gestionada con una autoemitida generada en la nueva máquina.
La gran ventaja es que una vez configurado todo no, el proceso de autenticación es como si no existiera; se selecciona la identidad a usar (tarjeta gestionada) y el selector de identidad ya tiene la tarjeta autoemitida y no pide ningún credencial, algo mucho más rápido que los métodos anteriores. Gana la usabilidad.

8.2.7. Seguridad

Es momento de hacerse algunas preguntas relacionadas con la seguridad de todo lo que llevamos visto.

¿Qué pasa si me roban una tarjeta? ¿Me han robado mi identidad digital?
El caso más probable sería que nos robaran una tarjeta gestionada en la fase de transporte, cuando el IP nos la manda después de haberla generado. El atacante, tendría la tarjeta, pero le faltaría lo más importante, los credenciales de autenticación para obtener una identidad del IP.

Otro caso sería que nos robaran un pendrive con nuestras tarjetas, gestionadas y autoemitidas, que llevamos encima por si queremos hacer uso de ellas en otra máquina. En este caso también sería inviable ya que el selector de identidad, al exportar tarjetas, las cifra con una clave simétrica que sólo conoce el dueño.

Una infocard no es más que un artefacto que automatiza la obtención de una identidad digital, uniformando el proceso de autenticación y abstrayéndonos de las comunicaciones. Así que no, el robo de una infocard no es el robo de una identidad digital, aunque facilite las cosas al atacante.

¿Qué seguridad tiene una infocard autoemitida? ¿Es solo el PPID/CardId la autenticación?
Si las tarjetas autoemitidas solamente se apoyaran en el PPID para la identificación del usuario, no sería difícil, conociendo el PPID, emitir identidades para suplantar al usuario en esa RP; o conociendo el CardId, en todas. Cuando el selector de identidad genera el CardID, también crea una clave maestra. Análogamente al caso del PPID, el selector genera con esa clave maestra y el identificador omnidireccional de la RP un par de claves asimétricas (pública y privada) con el que podrá firmar las identidades digitales (técnicamente tokens).

Una infocard autoemitida no es el CardID, sino este valor y su clave maestra. La seguridad reside en que la clave maestra no se transmite como sucede con el PPID, sino que se envía lo que se conoce como prueba de posesión.

¿Qué pasa si alguien espía los mensajes entre cualquiera de las partes?
Esta situación corresponde a ataques en la fase de transporte. Lo normal es que las comunicaciones se realicen aprovechando la seguridad que ofrece la capa TLS (ligadura de transporte o "transport binding"), por lo que actualmente queda descartado cualquier ataque en este sentido.

¿Cómo puede estar segura una RP de que una identidad proviene del IP en el que confía?
Cuando un IP emite una identidad digital la firma (ver XMLSEC) y le añade una marca temporal, un número de secuencia y en algunos casos información sobre la RP que la pidió. Esto significa que la RP puede verificar la procedencia de la identidad así como el momento en que se expidió y ,en algunos, casos si el IP sabe que se lo manda a ella (se puede considerar una violación de la privacidad el que el IP sepa a qué RPs accedemos).

¿Cómo de seguro es importar una Information Card?
Esta pregunta surge de un posible ataque en el que nos hacen creer que estamos importanto en el selector una infocard de una entidad fiable para que a la hora de autenticarnos (contra un falso IP), el criminal nos robe los credenciales. Todas las infocards gestionadas están firmadas (ver XMLSEC) por el emisor. Cuando el selector de identidad importa una infocard gestionada, comprueba la firma con los certificados del almacen que tenemos en nuestra máquina. Es de suponer que en ese almacén guardamos certificados de entidades fiables, así que el selector rehusará cargar esa tarjeta.

8.3. Caso de uso

Veamos un caso de uso completo. En la figura 8.12 mostramos los distintos mensajes que fluyen por el metasistema.

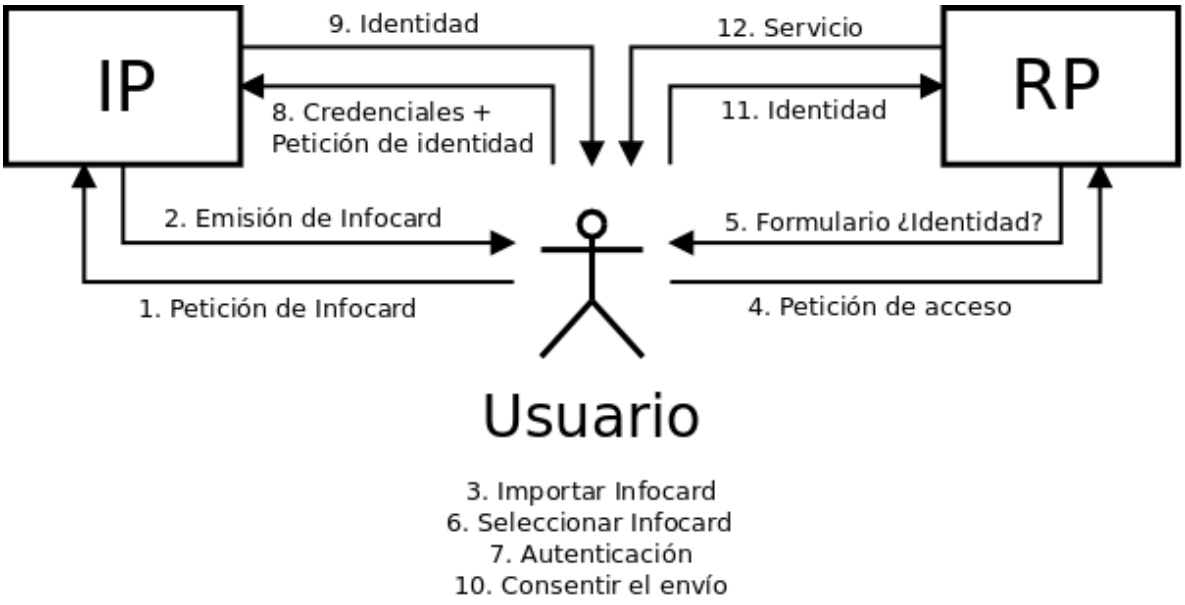

Figura 8.12: Caso de uso genérico con infocards.

1. El usuario tiene una cuenta en el proveedor de identidad IP y hace una petición para que éste le emita una Information Card. Es de suponer que tiene una sesión iniciada y por lo tanto está autenticado.

2. El IP genera y emite la infocard.

3. El usuario recibe el fichero que se le mostrará en su equipo como un icono del estilo del de la figura 8.13 e importa la nueva tarjeta con su selector de identidad. Si ya existiera la tarjeta, el selector le prguntará si desea reemplazarla, como se ve en la figura 8.14.

Figura 8.13: Icono del fichero de una infocard.

Figura 8.14: Importando una tarjeta que ya existía previamente.

4. El usuario hace una petición a una página web.

5. La RP sabe que el usuario no tiene una sesión iniciada y le presenta un formulario como el de la figura 8.15 de autenticación/petición de identidad mediante infocard.

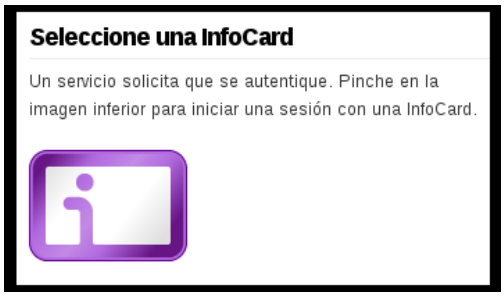

Figura 8.15: Formulario web de acceso con infocard.

6. Al pinchar en la imagen de la tarjeta del formulario, el navegador invoca al selector de identidad y el usuario debe seleccionar una infocard.

7. Posteriormente se le presenta una ventana como la de la figura 8.16 para que introduzca los credenciales de utilización (los del IP) de esa tarjeta. En este caso, la autenticación es mediante el método de usuario/contraseña.

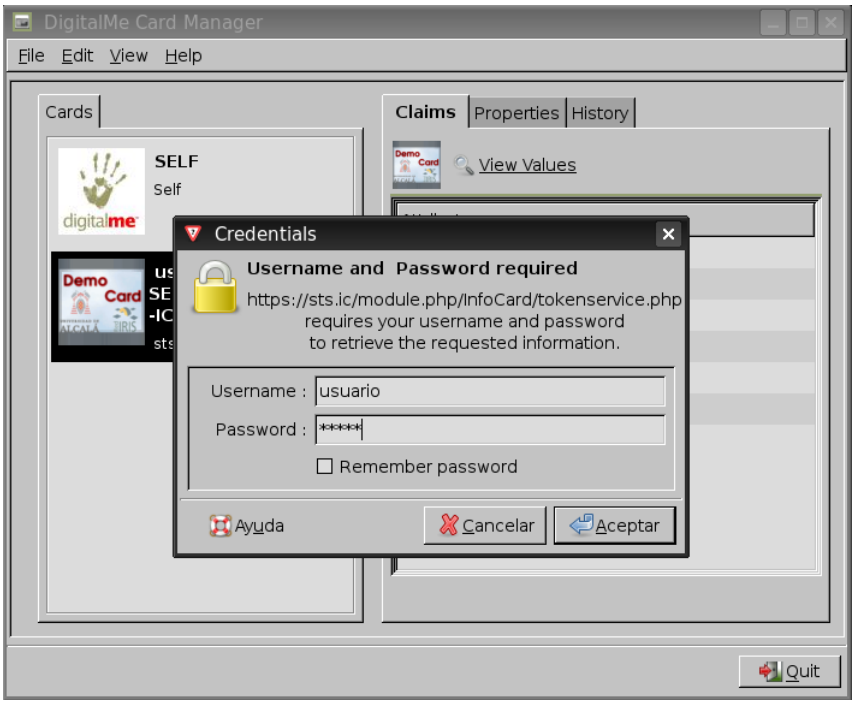

Figura 8.16: Método de autenticación por usuario/contraseña.

8. El selector de identidad envía una petición de identidad al IP, en ese mensaje también van incluidos los credenciales de autenticación.

9. El selector de identidad valida la petición y devuelve al selector una identidad firmada.

10. El selector muestra al usuario la información que va a mandar a la RP y si lo consiente. Ver figura 8.17.

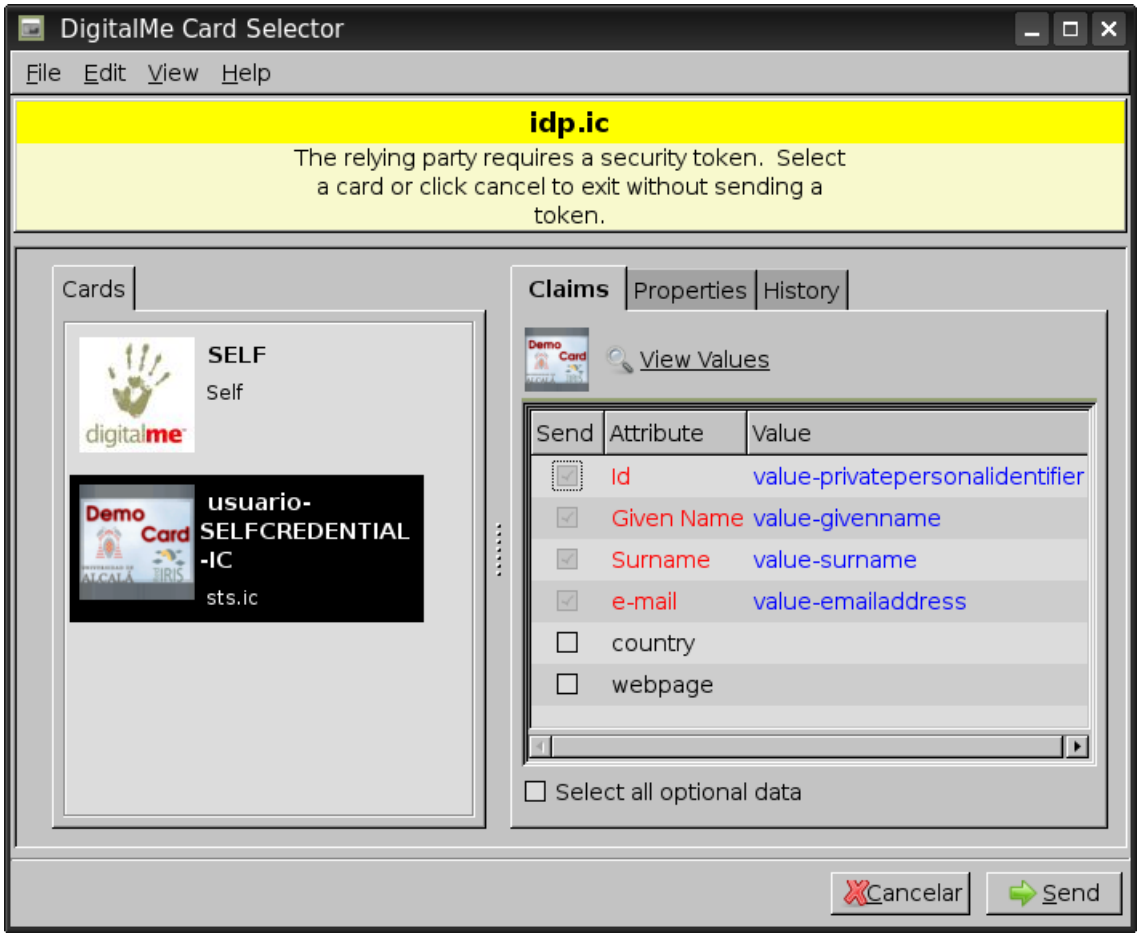

Figura 8.17: El selector presenta los datos y el usuario puede consentir el envío.

11. Si el usuario acepta, el selector manda la identidad a la RP.

12. La RP valida la identidad y autentica al usuario, ya tiene sesión y puede hacer uso del servicio. En este caso la RP presenta la información sobre la identidad del usuario que le llegó. Ver figura 8.18

Figura 8.18: La RP y el servicio muestra los datos de la identidad recibida.

8.4. Conclusiones

Veamos las ideas clave que tendría que interiorizar el lector.

Una infocard es:

- un método que unifica y automatiza la gestión de una identidad digital.
- el artefacto XML que lo posibilita.
- la iconización de una identidad digital en un selector de identidad.

El robo de una infocard gestionada no implica el robo de la identidad asociada.

El selector de identidad es una herramienta para gestionar las infocards.

Ventajas de las Information Cards

- Usabilidad.
- Seguridad y seguridad por usabilidad.

- Control de la privacidad.

Desventajas de las Information Cards

- Hace falta un selector de identidad, más software para el usuario.
- Hay que aprender el paradigma (puede no ser intuitivo para todo el mundo).
- No resuelve todos los problemas de seguridad (algo imposible) aunque soluciona bastantes.
- No soluciona el problema de usabilidad de los certificados, aunque tampoco lo empeora.

Enlaces

Además de los enlaces ya presentados, recomiendo ojear los siguientes para profundizar en el tema.

- InfoCard Explained Final - Vídeo en el que se explica la tecnología de Information Cards de manera bastante informal.
  http://mschnlnine.vo.llnwd.net/d1/ch9/0/8/0/1/8/1/InfoCard_Explained_Final.wmv
- Design Rationale behind the Identity Metasystem Architecture - Diseño racional detrás del metasistema. Un resumen de las decisiones tomadas para construir el metasistema a partir de las necesidades.
  http://research.microsoft.com/~mbj/papers/Identity_Metasystem_Design_Rationale.pdf
- Cardspace Managed Card and STS Test Harness - Página con un esquema de caso de uso y diversas funciones para probar las infocards.
  CardspaceManagedCardandSTSTestHarness
- Escritos varios sobre las Information Cards, detalles técnicos.
  - A Guide to Using the Identity Selector Interoperability Profile V1.5 within Web Applications and Browsers - Guía para adaptar una RP a las infocards.
    http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5_Web_Guide.pdf
  - An Implementer's Guide to the Identity Selector Interoperability Profile V1.5 - Manejo de las comunicaciones del selector de identidad con los diferentes roles (IP y RP).
    http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5_Guide.pdf
  - Identity Selector Interoperability Profile V1.5 - Sobre el manejo y emisión de las tarjetas.
    http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5.pdf
  - Identity Metasystem Interoperability Version 1.0 - Recopilación de OASIS sobre el metasistema.
    http://docs.oasis-open.org/imi/identity/v1.0/identity.html

Capítulo 9

eduroam

La última herramienta que queda por explicar (ya hemos visto el metasistema y métodos de inicio de sesión único) es eduroam.

9.1. ¿Qué es eduroam?

Traduciendo de la página oficial [eduroam, a].
eduroam (**edu**cation **roam**ing) es el servicio seguro de itinerancia a nivel mundial desarrollado por la comunidad internacional de investigación y educación.
Eduroam permite a estudiantes, investigadores y personal de las instituciones miembros disponer de conectividad a internet entre los campus y cuando visitan otras instituciones miembros simplemente abriendo su ordenador portátil.

Traduciendo de la Wikipedia [Wikipedia, p].
eduroam es un servicio seguro de itinerancia para usuarios de la educación superior. La confederación europea (una confederación de servicios autónomos de itinerancia) está basada en un conjunto de requisitos técnicos y organizacionales que cada miembro de la confederación debe aceptar firmando la política GN2-07-328.

Según RedIRIS.
... eduroam ES es una iniciativa englobada en el proyecto RedIRIS y que se encarga de coordinar a nivel nacional las iniciativas de diversas organizaciones con el fin de conseguir un espacio único de movilidad a nivel nacional. Este espacio único de movilidad consiste en un amplio grupo de organizaciones que en base a una política de uso y una serie de requerimientos tecnológicos y funcionales, permiten que sus usuarios puedan desplazarse entre ellas, disponiendo en todo momento de servicios móviles que pudiera necesitar. El objetivo último sería que estos usuarios al llegar a otra organización dispusieran, de la manera más transparente posible, de un entorno de trabajo virtual con conexión a Internet, acceso a servicios y recursos de su organización origen, así como acceso a servicios y recursos de la organización que en ese momento les acoge.

Objetivos del proyecto eduroam ES:

- *Coordinar a la puesta en marcha de infraestructuras de movilidad en nuestra comunidad, sirviendo de punto de encuentro de problemas y soluciones.*
- *Coordinar el desarrollo de una política de uso con el fin de crear un espacio único de movilidad entre nuestras organizaciones y compatible con el desarrollado a nivel europeo.*
- *Homologar las soluciones tecnológicas a implantar en las diferentes organizaciones con las acordadas a nivel europeo e internacional en este sentido.*
- *Trabajar en soluciones que ayuden a difundir información sobre tipos de instalaciones e información a nivel de organización sobre: modos de acceso, cobertura, etc.*
- *Informar de todos los temas relativos a la movilidad: guías de apoyo, estándares, soluciones (tanto propietarias como de libre distribución), etc.*
- *Promocionar nuevas soluciones e iniciativas originadas en organizaciones de nuestra comunidad tanto dentro de nuestra red, como a nivel internacional.*

[RedIRIS, b]

En resumen.

Para el usuario final (relacionado con el mundo universitario), eduroam es un servicio que le permite tener acceso a internet a través de la red inalámbrica de la universidad en la que se encuentre (siempre y cuando dicha entidad forme parte de eduroam) símplemente introduciendo sus credenciales (usuario y contraseña) de su universidad de origen.

En el contexto de este proyecto, eduroam será la gran puerta que tendrá que cruzar el usuario para acceder a un universo lleno de servicios sin que se le vuelva a preguntar por sus credenciales, en todo caso por su identidad digital (como enseñar la pulsera de "todo incluido"). Es por esto que necesitemos comprender más a fondo el funcionamiento de la infraestructura.

En la figura 9.1 podemos ver el logo de eduroam. Eduroam es una marca registrada de TERENA.

Figura 9.1: Logotipo de la marca eduroam

9.2. Organización

eduroam se organiza en tres confederanciones.

- eduroam Europa, ver imagen 9.2.
- eduroam Asia-Pacífico, ver imagen 9.3.
- eduroam América (formada únicamente por Canadá)

Figura 9.2: Confederación europea

9.2.1. eduroam Europa

Según la página oficial de eduroam Europa (http://www.eduroam.org/?p=europe).

El servicio de eduroam en Europa es un servicio confederado que cuenta con la colaboración de 36 federaciones de nivel nacional. Esto son cientos de intituciones, la mayoría de las cuales posee y controla su propia infraestructura. La coordinación nacional e internacional la controlan los operadores

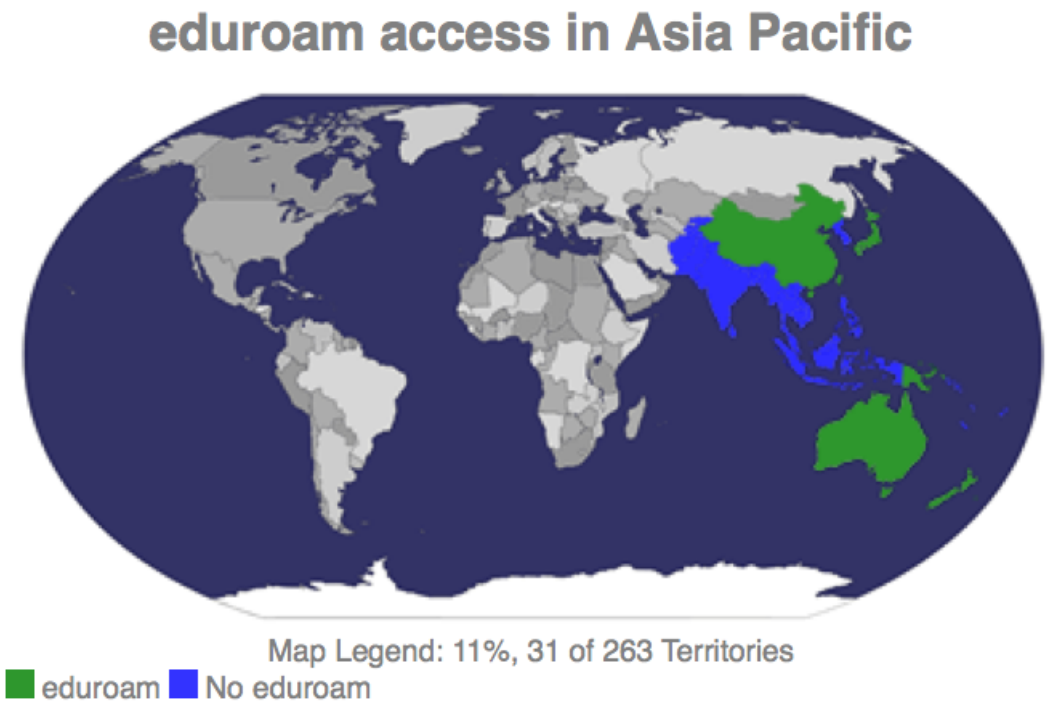

Figura 9.3: Confederación Asia-Pacífico

nacionales y un equipo central operacional de eduroam fundado por el proyecto GÉANT (GN3). Este equipo operacional lleva a cabo tareas rutinarias con participantes como Srce, SURFnet, UNI-C y TERENA.

El Servicio de Actividad de eduroam del proyecto GÉANT ha creado herramientas para monitorizar el servicio y dar soporte a los usuarios finales.

eduroam continúa su desarrollo y evolución gracias al apoyo del GÉANT-funded Joint Research Activity (JRA3).

Hasta el momento se ha conseguido:

- Tentativas para estandarizar RadSec dentro del IETF.
- Integración de RadSec en la infraestructura eduroam.
- RadSecProxy.
- Infraestructura inicial para soportar RadSec en FreeRADIUS.

En el futuro se continuará con lo anterior y se incluirá:

- Nuevos protocolos de autenticación extensibles.
- Soporte para la internacionalización de nombre de usuario en eduroam.
- Privacidad preservando la identificación para análisis estadísticos y de riesgos.

9.3. Arquitectura 802.1X

Ya hemos visto sin entrar en implementaciones qué es eduroam, ahora procedo a explicar la técnica que hay detrás y hace posible que funcione esta infraestructura.

La clave está en el estándar del IEEE 802.1X, que ofrece mecanismos para autenticar a clientes (usuarios de la red) y controlar el acceso a la red de una forma basada en puerto.

Reduciéndolo al máximo, 802.1X permite autenticarse contra un punto de acceso (máquina que hace de pasarela a una red) a la que me se conecta directamente (como mucho estamos en la misma LAN) para conseguir acceso a una red. La complejidad reside en el método de autenticación, la seguridad y los nodos que recorre la información (mis credenciales normalmente) antes de darme como válido en el servicio.

Este enfoque es perfecto para el funcionamiento de eduroam, ya que ofrece dos características fundamentales, autenticación WiFi e itinerancia.

Traduciendo de "Inventory of 802.1X-based solutions for inter-NRENs roaming" [Dobbelsteijn, sección Introduction].

Cuando una red soporta el estandar IEEE 802.1X, inicialmente el usuario debe realizar un par de acciones para ser capaz de iniciar sesión. El sistema operativo debe soportar 802.1X o hay que contar con un programa que aporte esta capacidad, también hay que configurar la cuenta de usuario.
Una vez configurado lo anterior, el usuario puede usar una red 802.1X (ya sea cableada LAN, o inalámbrica WiFi).
El estándar IEEE 802.1X para autenticación basada en puerto es una solución en la capa 2 entre el cliente y el dispositivo de control de acceso (un punto de acceso inalámbrico o un conmutador "switch"). En el marco de trabajo 802.1X, la información de autenticación va sobre EAP (Extensible Authentication Protocol) que permite el uso de varios métodos de autenticación. Los dispositivos de control de acceso se comunican con un terminal RADIUS para verificar al usuario, este sistema es seguro y escalable.
Después de la autenticación, la comunicación entre el cliente y el punto de acceso va encriptada usando claves dinámicas.

9.3.1. ¿Qué es RADIUS?

Traduciendo de [Linux, punto 1.5].

RADIUS (Remote Authentication Dial-In User Service o Servicio de Usuario para Autenticación Remota por Marcado) fue un servicio originalmente usado por los proveedores de acceso a internet que requerían a sus usuarios que se autenticaran mediante usuario/contraseña antes de autorizarles el acceso a la red.
802.1X no especifica un tipo de servidor de autenticación, así que RADIUS se ha convertido en el servidor de autenticación "de facto".
No hay muchos protocolos AAA disponibles, pero RADIUS y DIAMETER cumplen con los requisitos.

Resumiendo:

RADIUS es un protocolo AAA (Authentication, Authorization and Acounting), esto significa que ofrece capacidades para autorización, autenticación y contabilidad. Un servidor RADIUS permite controlar el acceso de los usuarios a una red, así como establecer limitaciones y llevar un registro del

uso que hacen de la misma. Esto es precisamente lo que se hace en eduroam.

9.3.2. Arquitectura

El marco 802.1X añade funcionalidad a los componentes de una red, por lo que no hace falta adquirir ninguno más.
En una red muy simple, se podrían localizar tres componentes básicos:

Terminal o suplicante: Es el equipo desde el que se conectará el usuario, un PC con una tarjeta de red y un S.O. con soporte para 802.1X.

Autenticador: Es un conmutador o un punto de acceso con soporte para 802.1X. Provee al terminal de un puerto para la conexión que puede abrir o cerrar.

Servidor de autenticación: Es el servidor RADIUS al que consultará el autenticador para ver si se le permite al terminal usar un puerto de conexión y determinar por qué red virtual (VLAN) irá el tráfico.

En la figura 9.4 tenemos un ejemplo de esta miniarquitectura.

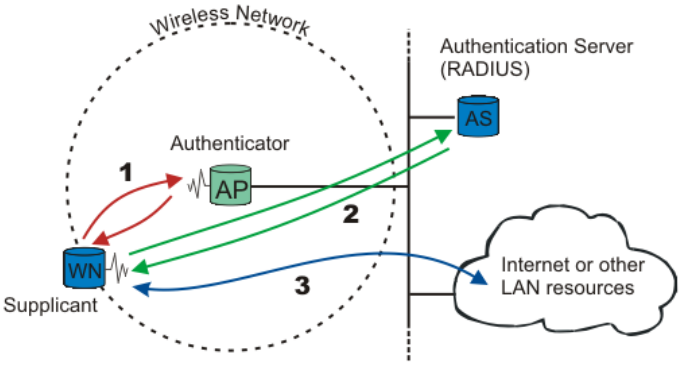

Figura 9.4: Arquitectura eduroam simplificada

9.3.3. Caso de uso

Veamos un caso de uso genérico.

- John, un holandés que pertenece a la institución **institution_b** está en la UAH y ha descubierto una red eduroam. Pretende conectarse para navegar por internet, para ello introduce sus credenciales, su identificador de usuario `john@institution_b.nl` y su contraseña.
- El punto de acceso recoge esa petición y pregunta al servidor RADIUS al que está conectado si debe darle acceso al servicio, o dicho de otra forma, si los credenciales son correctos.
- El servidor RADIUS descompone el identificador de usuario y descubre que no es el responsable del dominio `@institution_b.nl`, la uah maneja el sufijo `@uah.es`. Reenvía la petición al servidor RADIUS de nivel nacional NREN (en este caso sería RedIRIS).

- El servidor nacional descubre que él no maneja el sufijo `.nl` sino el `.es`, así que reenvía la petición al servidor RADIUS de nivel europeo (DANTE).
- El servidor RADIUS de nivel europeo descubre que el prefijo `.nl` es de Holanda y manda al servidor RADIUS de nivel nacional de Holanda (SURFnet) la petición.
- El servidor RADIUS de nivel nacional de Holanda analiza el sufijo `institution_b.nl` y envía la petición al servidor RADIUS de la institución institution_b.
- La institución analiza los credenciales de John y si son válidos, emitirá un mensaje de aceptación que recorrerá el camino inverso hasta el servidor RADIUS de la UAH.
- El servidor RADIUS de la UAH, le da el visto bueno al AP y este le permite hacer uso de la red.

En la figura 9.5 se puede ver una representación genérica de lo explicado anteriormente.

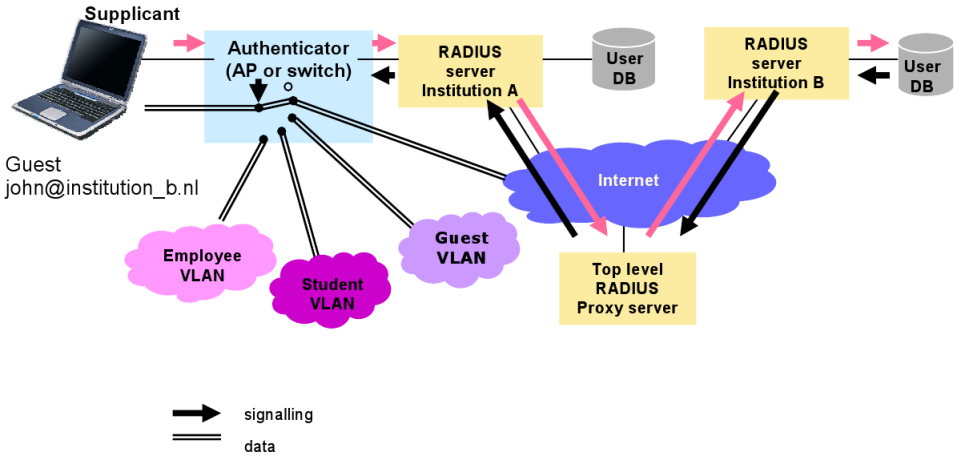

Figura 9.5: Arquitectura genérica de eduroam (©SURFnet)

9.3.4. Escalabilidad

Como se vio anteriormente, el servidor RADIUS local sólo debe saber a qué servidor RADIUS de retransmisión enviar las peticiones de los usuarios desconocidos. Cuando una nueva red entra en este consenso de itinerancia, sólo hay que actualizar el servidor de retransmisión nacional (a no ser que se añada un país o un grupo de instituciones, que sería el servidor de más alto nivel y habría que añadir un nuevo domino, como .nl si se adjuntara Holanda).

Para extender esta infraestructura de itinerancia a una escala europea, hace falta únicamente un servidor RADIUS de retransmisión de nivel internacional (ver figura 9.6).

Es posible establecer relaciones bilaterales entre servidores que intercambian mucho tráfico o tráfico que solo es relevante a nivel local.

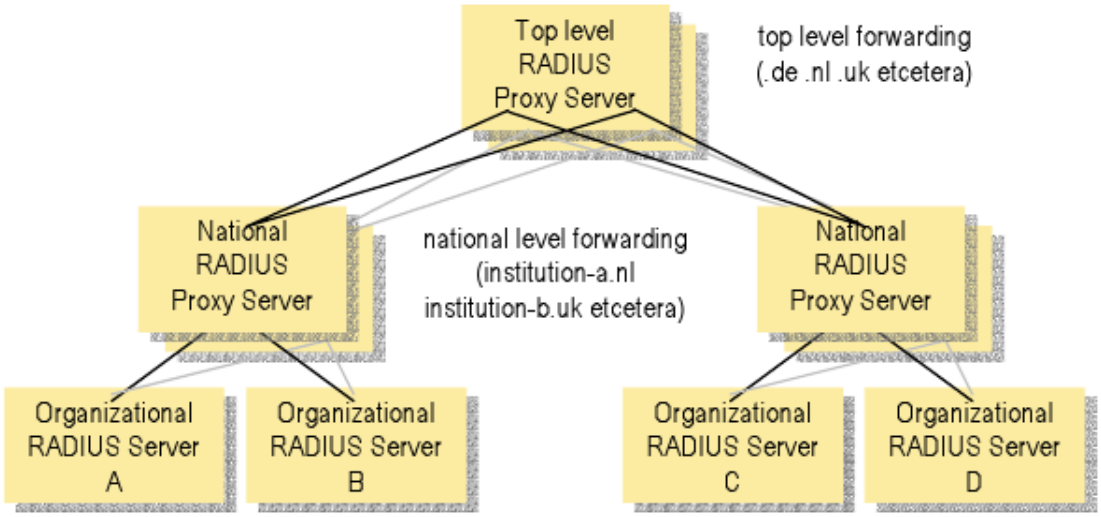

Figura 9.6: Jerarquía de servidores RADIUS

El uso de RADIUS rambién hace fácil conectar la infraestructura de itinerancia existente con una red de telecomunicaciones (WIFI, GPRS o UMTS).

Según esta configuración es posible que se produzcan ciclos en el camino de los mensajes, lo que puede llevar a fallos en los servidores. Para prevenirlo, cada servidor RADIUS puede ser oblidado a no retransmitir los mensajes con el sufijo que gestiona. También, los servidores de retransmisión pueden filtrar estos eventos y controlar el número de saltos que realiza cada paquete.

Es posible establecer robustez en todos los niveles de la arquitectura. Los dispositivos de control de acceso se pueden instalar por pares, aunque no es lo normal por los altos costes. También se pueden configurar los disposivos de control de acceso para comunicarse con varios servidores RADIUS, así cuando uno falla, el otro responde. Se puede usar el mismo mecanismo entre servidores de retransmisión.

Puesto que el programa que ofrece el servicio RADIUS no consume muchos recursos de la máquina, un PC de gama media puede gestionar cientos de peticiones por segundo. La autenticación solo se necesita al principio de una sesión de usuario y cuando el usuario cambia de punto de acceso (movilidad geográfica). Visto así, un servidor RADIUS local o uno de retransmisión, puede, potencialmente, gestionar miles de sesiones simultáneamente.

La escalabilidad en términos de rendimiento está implícitamente relacionada con el hecho de que los puntos de acceso manejan la encriptación a nivel de capa de enlace a velocidad hardware.

9.3.5. EAP

Como los credenciales viajan por servidores intermedios que no están bajo el control de la institución a la que pertenece el usuario, es necesario que vayan protegidos. Este requisito limita los tipos de autenticación que pueden usarse. Básicamente hay dos categorías de de métodos de autenticación:

Credenciales basados en mecanismos de clave pública: EAP-TLS, EAP-SIM.
Es difícil de implementar porque requiere de una infraestructura de clave pública-privada.

Autenticación tunelizada: EAP-TTLS, PEAP.
Es el más usado porque solo necesita certificados en los servidores. (Ver figura 9.7)

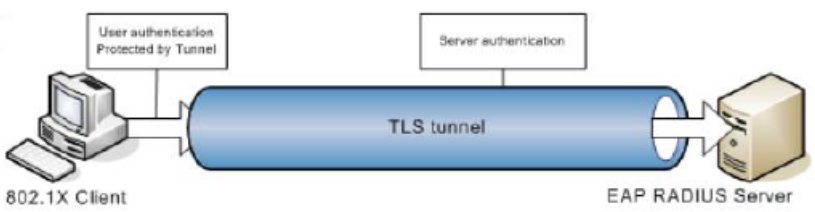

Figura 9.7: Autenticación tunelizada (©Alfa&Ariss)

En la figura 9.8 se muestra la pila de protocolos de marco 802.1X. La información de autenticación se transporta sobre EAP [RFC, 2004], un protocolo que permite el uso de cualquier credencial, como usuario/contraseña, certificados, OTPs (contraseñas de un solo uso)... Estos mecanismos se implementan sobre los tipos EAP: MD5, TLS, TTLS, MS-CHAPv2, PEAP, Mob@c y EAP-SIM.
Tanto el suplicante como el servidor RADIUS deben utilizar el mismo tipo EAP, aunque el dispositivo de control de acceso y los servidores RADIUS intermedios no tienen porqué.

Actualmente, TLS (seguridad de capa de trasporte), TTLS (Seguridad de capa de transporte tunelizada) y PEAP (EAP protegido) son las apuestas más firmes. También se están realizando pruebas adicionales con la autenticación basada en contraseñas de un solo uso enviadas mediante SMS. TLS, TTLS y PEAP establecen una conexión TLS basada en el certificado del servidor RADIUS entre el cliente y el dispositivo de control de acceso. Este mecanismo de autenticación mutua, puede prevenir ataques de "Man in the Middle" o intermediario [Wikipedia, a]. TLS usa un certificado de cliente para autenticar al usuario, mientras que TTLS normalmente se usa para transportar el credencial (usuario/contraseña). Puesto que TTLS y PEAP son protocolos de tunelización, cualuier otro protocolo puede usarse sobre estos.

9.3.6. Seguridad

Anteriormente se ha visto que la fortaleza de la autenticación reside en una capa TLS que da un nivel de seguridad/extensibilidad/manejo adecuado para este menester, además de prevenir ataques del tipo intermediario.
Para asuntos como la integridad de la información o privacidad, se han propuesto unas cuantas extensiones de seguridad (WPA, TKIP, 802.11i, etc.) que también casan con el marco de trabajo 802.1X. Sin embargo el mecanismo actual de claves WEP que se renuevan, proporciona un alto nivel de enciptación cuando las claves se renuevan regularmente (normalmente cada 20 minutos o menos con claves de 64 bits). Se recomienda esta longitud de clave por retrocompatibilidad con adaptadores

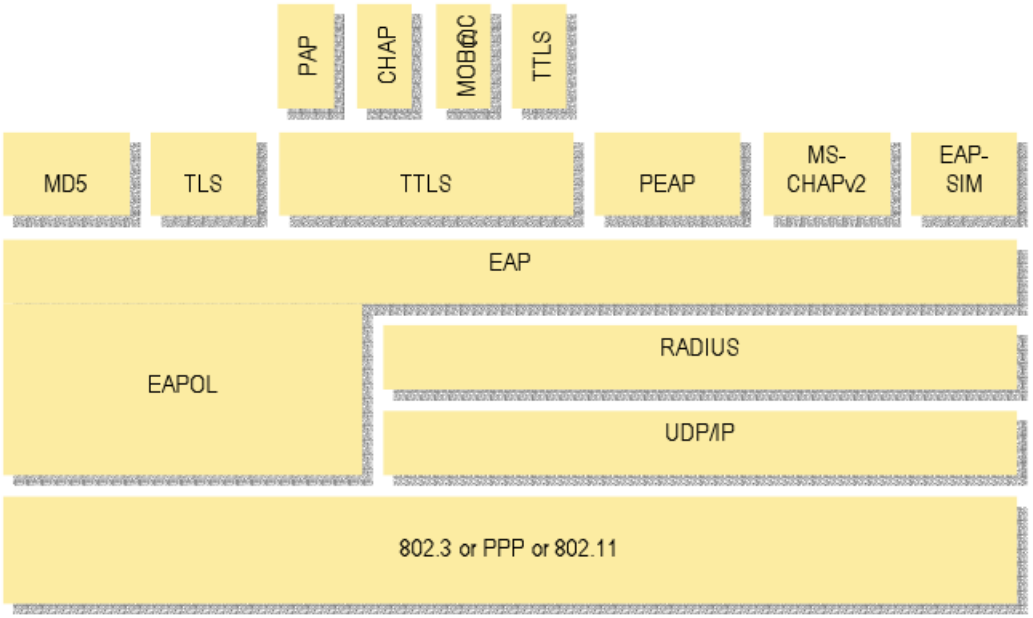

Figura 9.8: Torre de protocolos, EAP puede soportar varios mecanismos de autenticación.

de red antiguos.

La seguridad en la infraestructura RADIUS viene dada por el uso de claves compartidas entre servidores y por la instalación de estos servidores en partes seguras de la red diseñadas cuidadosamente. Cada máquina tiene su clave propia clave compratida. Sin embargo, algunos mensajes podrían ser alterados por el camino, así que como extra, las rutas entre servidores radius pueden ser protegidas con túneles IPSEC.

A continuación muestro una revisión de las posibles amenazas relacionadas con la seguridad y los mecanismos que proporciona 802.1X para hacerles frente.

Robo de identidad
Es difícil detectar en cualquier proceso de autenticación cuándo un usuario emplea los credenciales de otra persona. RADIUS establece un registro detallado de las sesiones de usuario, así que se pueden relacionar sesiones repudiadas por la víctima con la sesión actual del ladrón de identidad.

Pérdida de credenciales
Tan pronto como se le notifique a los administradores la pérdida de credenciales, esa cuenta se puede desactivar.

Abuso del ancho de banda
Detectar y prevenir el abuso de ancho de banda en el nivel 2 es un problema común a cualquier red. Se pueden tomar acciones para gestionar el tráfico basándose en parámetros de la red, pero esto no evita que el usuario inunde el medio con cantidades ingentes de información en el caso de una red de difusión (una red WIFI). Usando 802.1X, el emisor podría ser localizado.

En la capa 1, el administrador de red está indefenso. Emitiendo microondas en la banda de los 2,4GHz inutilizaría la red.

Abuso de contenidos

Cada abuso puede ser rastreado hasta una cuenta de usuario desde que el momento en que el registro del usuario está vinculado a una dirección IP. Evidentemente requiere una relación entre el identificador de usuario y la IP asignada para la sesión.
Cualquier cuenta sospechosa o incluso ciertos dominios pueden ser bloqueados en cualquier nivel de la arquitectura para prevenir que el usuario o la institución hagan uso de nuestros recursos (autoprotección).

9.3.7. Dispositivos de red

Revisaré las opciones que hay para cada uno de los tres tipos de dispositivos que puede haber.

Clientes

Para autenticación por usuario/contraseña, EAP-TTLS se ha probado ampliamente. Es fácil de configurar porque no requiere de una infraestructura de clave pública-privada (PKI) con certificados para los usuarios. Sólo el servidor RADIUS debe tener un certificado. Si ya existe una PKI, EAP-TLS puede ser una forma aún más segura de autenticación.

Para usar la autenticación basada en 802.1X, el cliente debe de usar un sistema operativo o un programa sobre dicho sistema que lo soporte, así como EAP y el método de autenticación sobre el que se base.

A día de hoy existen clientes (o suplicantes) para los sistemas operativos mayoritarios (Windows, MacOS y basados en UNIX), además existen proyectos de código abierto como open1x [Open1X] o wpa_supplicant [Malinen] que deberían permitir portar dichas capacidades a otros sistemas.

Dispositivos de control de acceso

La mayoría de los puntos de acceso que se venden tienen soporte para autenticación por 802.1X. En la imagen 9.9 se ve un linksys WRT54G, un modelo asequible de una filial de CISCO.

Servidores RADIUS

El servidor Radius de la institución de origen, el que evaluará el mensaje de petición de acceso, debe soportar el tipo EAP establecido en la autenticación. Lo servidores intermedios sólo deben reenviar el mensaje.

Ahora mismo, la solución más extendida es FreeRADIUS, aunque se están portando muchos sistemas a RADIATOR.

La contabilidad (accounting) es una característica de estos servidores que permite llevar un registro de las peticiones de autenticación. Combinando estos registros con la asignación de direcciones IP es posible rastrear intentos de acceso maliciosos o abuso de la red. Esta característica se convierte en especialmente relevante cuando se conectan proveedores comerciales a la plataforma de itinerancia. Los mensajes de contabilidad se pueden retransmitir sobre la infraestructura de servidores.

Figura 9.9: Linksys WRT54G, con capacidad para 802.1X

9.3.8. Usabilidad

Uno de los criterios de diseño fue la facilidad de uso para el usuario final. Las pruebas muestran que, una vez configurado el cliente de 802.1X, la conexión a los puntos de acceso se maneja de forma sencilla y transparente.

La única desventaja es que este marco de trabajo es bastante nuevo y normalmente hace falta instalar y configurar el cliente. Cuando venga por defecto como parte del sistema operativo, ya no habrá excusa.

La decisión e usar TTLS a corto plazo fue por la disponibilidad de clientes que lo soportaran en detrimento de suplicantes para PEAP. Cuando PEAP gane cierta inercia, la migración será simple.

9.4. Proyectos

En esta sección veremos dos proyectos basados en eduroam.

9.4.1. eduGAIN

Traduciendo de la página oficial [eduGAIN].
El propósito de eduGAIN (logotipo en la imagen 9.10) es proveer de los medios para alcanzar la interoperabilidad entre diferentes infraestructuras de autenticación y autorización (AAI).

Figura 9.10: Logo de eduGAIN

Hay varios desarrollos de AAIs en uso sobre servidores de nivel nacional (NREN). Shibboleth es la tecnología de federación usada en en las redes de investigación de EE.UU. (Internet2), Suiza, Finlandia, Alemania, Gran Bretaña, Hungría y Grecia. En España usamos PAPI, A-Select en Holanda, simpleSAMLphp en Dinamarca y Noruega. También hay AAis basadas en RADIUS y SAML en Croacia. Todas estas soluciones tienen un mismo objetivo: permitir la creación de un entorno de confianza en el que los usuarios puedan ser identificados electróniamente mediante un sistema de gestión de la identidad.

Para permitir el acceso a recursos protegidos y servicios de otras federaciones, los usuarios necesitan en primer lugar estar autenticados en su AAI de origen y autorizados por el proveedor de servicios anfitrión. eduGAIN provee de la tecnología necesaria para seguir estos pasos e interconectar diferentes infraestructuras. Aquí entra en juego el rol de la confederación, una federación de federaciones.

La tecnología eduGAIN implica una traducción de protocolos entre los propios de las AAIs y SAML así como el mapeado de atributos dependiendo de las definiciones locales. Se asegura un transporte seguro de la información mediante canales de comunicación cifrados entre entidades. La información necesaria para las entidades locales de diferentes federaciones se centraliza en un servicio de metadatos, donde puede ser consultada y actualizada dinámicamente.

Según RedIRIS [RedIRIS, a].
eduGAIN es el nombre que recibe la arquitectura de autenticación y autorización de la red GÉANT2. GÉANT2, formada por 30 redes académicas u organizaciones europeas, es una pieza clave para la colaboración académica e investigación. Con 30 millones de usuarios es la mayor red de comunicaciones de la historia creada para la comunidad académica e investigadora europea con un amplio alcance geográfico. Además está conecta con otras regiones del mundo como Latinoamérica (ALICE-RedCLARA), el área Asia-Pacífico (TEIN2/ORIENT), el Norte de África y Oriente Medio (EUMEDCONNECT), y mantiene conexiones con Norteamérica, permitiendo así una colaboración global en investigación.

eduGAIN persigue proporcionar un servicio de autenticación y autorización completo actuando como conector de arquitecturas de AA ya establecidas, ya sean a nivel nacional o internacional. Así eduGAIN constituirá una superestructura para federar las arquitecturas ya existentes, las cuales utilizarán las interfaces de eduGAIN a través de los elementos apropiados. Por lo tanto eduGAIN

establecerá los lazos de confianza que permite a los componentes de la infraestructura interactuar directamente entre ellos y proporcionar las interfaces necesarias para encaminar y trasladar dichas interacciones.

eduGAIN es una arquitectura basada en concepto de confederación y en el establecimiento de dinámico de confianza. En el entorno eduGAIN una confederación es conjunto de parejas de federaciones, a nivel nacional o de una comunidad de usuarios, que cooperan para proporcionar servicios a los miembros de la organización y a usuarios más allá de los límites de gestión de la organización.

Como los miembros de una federación participante no tienen por qué conocer de antemano a miembros de otras federaciones, se necesita un procedimiento para establecer confianza entre ellos. Gracias a lazos de confianza proporcionadas por el esquema de confederación la confianza se establecerá dinámicamente.

eduGAIN permite el establecimiento dinámico de la confianza a través del Servicio de metadatos, de su PKI y de un conjunto de normas de nomenclatura para sus componentes.

El objetivo de esta propuesta es diseñar un sistema de validación de los componentes de eduGAIN que se base en las raíces de confianza descritas y que permita la gestión dinámica de los lazos de confianza.

9.4.2. DAMe

Estoy obligado a citar este trabajo de la Universidad de Murcia porque ha sido la inspiración del presente proyecto, por lo menos a alto nivel.

Traduciendo de la página oficial [**Universidad de Murcia, sección Objectives**].
El objetivo principal de este proyecto es definir un sistema de autenticación y autorización unificado para servicios federados alojados en la red eduroam. Esos servicios federados pueden ser desde acceso a la red hasta el control de servicios distribuidos, como la computación grid. Muchas de las propuestas se apoyan en un mecanismo global de SSO (Inicio de Sesión Único) basado en mecanismos y arquitecturas ya desarrollados.

La arquitectura general de DAMe se muestra en la figura 9.11, ilustrando que eduroam y eduGAIN son las dos iniciativas principales de este proyecto. La primera proveerá de una infraestructura de itinerancia y eduGAIN será usada como la principal AAI (Infraestructura de Autenticación y Autorización) para intercambiar credenciales durante el proceso de inicio de sesión único y otros procesos de autorización. Como se ve en la figura 9.11, cualquier usuario de DAMe primero será autenticado usando eduroam, obteniendo un token de inicio de sesión único que será usado posteriormente para acceder a los recursos federados y protegidos. De esta forma, el proceso de SSO se inicia con el acceso a la red evitando futuras reautenticaciones a este nivel, entonces se realiza un proceso de autorización para determinar qué tipo de acceo a la red se debería ofrecer según los atributos del usuario. Finalmente, cuando se pide un recurso protegido en una institución que pertenece a la federación, se usará eduGAIN para validar el token SSO y obtener atributos adicionales si es necesario. Para lograr este objetivo, el proyecto está dividido en cuatro grandes actividades.

El primer paso en DAMe es la extensión para la infraestructura eduroam usando la arquitectura NAS-SAML, así la movilidad del usuario puede ser controlada por aserciones de seguridad y políticas expresadas en lenguajes estándar y extensibles, como SAML y XACML. De esta forma proveemos de un mecanismo para potenciar la interoperabilidad entre las diferentes organizaciones estableciendo un lenguaje común para los credenciales y las condiciones de control de acceso. Esta aproximación permite

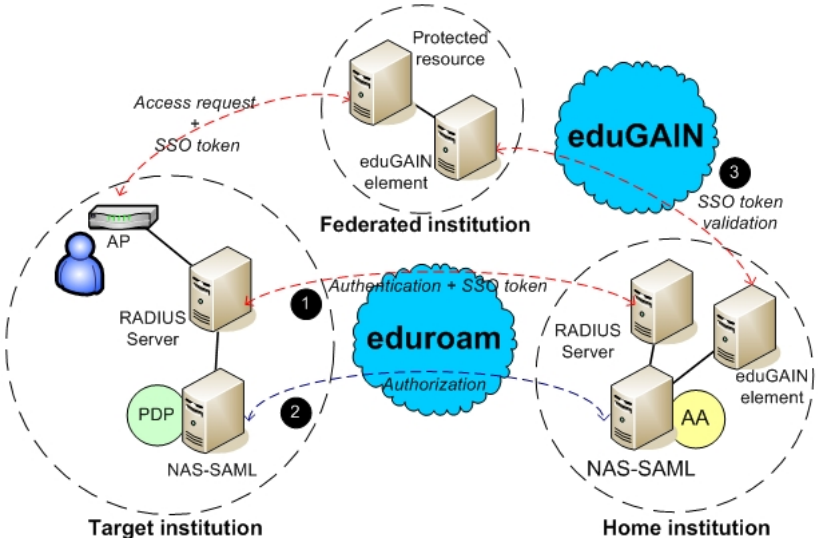

Figura 9.11: Arquitectura de DAMe

establecer mecanismos para establecer un control de acceso más ajustado, más controlable. Según los atributos asignados a los usuarios, basados en esquemas como SCHAC, obtendrán una conexión a la red personalizada, con capacidades diferentes como QoS (calidad de servicio) y opciones de seguridad.

La segunda fase se aprovecha de los mecanismos de federación como eduGAIN. Estamos al tanto de que algunas organizaciones ya han desplegado sus propias soluciones de autenticación y autorización basadas en esquemas de atributos de usuario, o incluso basadas en aserciones SAML y no-SAML, y por consiguiente debe incluirse el desarrollo de soluciones de integración para permitir interoperar a diferentes dominios de autorización. De hecho, la arquitectura NAS-SAML se ha combinado exitosamente con otras propuestas basadas en certificados de atributo X.509, específicamente el proyecto PERMIS. Siguiendo una aproximación parecida, planeamos usar los sistemas eduGAIN como autorizador de nuestro sistema de control de acceso a la red. Las aserciones SAML sobre atributos que establecen los credenciales de usuario serán gestionadas por los elementos que ya existen en la red eduGAIN, así podremos incluir un nuevo perfil NAS-SAML como parte de los servidores finales de eduGAIN.

El tercer paso es proveer de un inicio de sesión único real desde un punto de vista gobal. Los usuarios serán autenticados una vez, durante la fase de control de acceso a la red. Después de haberse autenticado en la red usando 802.1X, esa autenticación buscará los tokens necesarios firmados por eduGAIN de forma que no habrá necesidad de repetir el proceso de inicio de sesión en la capa de aplicación. De esta forma la autenticación en eduGAIN se iniciará junto con la de NAS-SAML. Esto implica la generación de tokens SAML firmados para entregar a los usuarios, usando por ejemplo el método PEAP (Protocolo Protegido de Autenticación Extensible) para mandar los credenciales de autenticación. Los usuarios contactarán con los proveedores del servicio eduGAIN y no habrá necesidad de reautenticar al usuario.

Finalmente, tenemos previsto usar una red con capacidad AAA y la información relacionada sobre la autorización para ofrecer mecanismos de autorización a servicios de nivel de aplicación. La mayoría de los servicios distribuidos, como por ejemplo los servicios Grid, tienen un componente de

seguridad responsable de determinar si un usuario en particular puede realizar una petición concreta. Proponemos el desarrollo de un mecanismo capaz de enlazar con el proceso de autorización de la arquitectura definida previamente con la finalidad de obtener los credenciales de usuario de la federación. Esta integración se realizará, cuando sea posible, usando puntos de extensión estándar, por ejemplo la interfaz de autorización OGSA-Auth para plataformas de computación Grid OGSA, u otros puntos de extensión.

Recomiendo mirar el siguiente enlace:

- Presentación del proyecto DAMe en el TNC2007
  Deploying Authorisation Mechanisms for Federated Services in eduroam (DAME).
  http://tnc2007.terena.org/programme/presentations/show9c65.html

9.5. Conclusiones

Hemos visto lo que es eduroam, una red/infraestructura de propósito académico. También sabemos cómo se organiza y cómo funciona. Qué es un servidor RADIUS y el marco de trabajo 802.1X que hace funcionar todo.
También se han mostrado dos proyectos, y resalto DAMe.

Aunque en un principio pueda parecer que lo que estoy haciendo es una reimplementación de DAMe, debo decir a mi favor que no es exactamente así. Ambos tienen la idea del inicio de sesión único unificado pero lo que aporta éste es el metasistema, lo que redunda en la gestión de la identidad o identidad centrada en el usuario. No propongo hacer sólo una herramienta "middleware", sino expandirla con una capa de identidad que, dependiendo del contexto, ofrecerá unas capacidades determinadas.

No hay mucha complicación (sin entrar en implementaciones particulares) en readaptar un método EAP para ofrecer un servicio de USSO; el problema subyace en los atributos de usuario que se manejan, la traducción de atributos entre diferentes dominios de autenticación, el control del usuario de su identidad (el flujo de sus datos), la seguridad del token que se maneja y por último la usabilidad para un público profano (indispensable para un uso real). Estos problemas los soluciona en gran medida el metasistema, con lo que es de suponer que lo que propongo, aunque no deje de ser una prueba de concepto, podría ser una solución perfectamente viable y estar en funcionamiento en un breve plazo de tiempo.

Parte III

La solución

Capítulo 10

Visión global

Este capítulo es la introducción a la parte final del proyecto. En esta tercera y última parte, procederé a comentar la solución que propongo así como a explicar los entresijos tecnológicos y la modificación de programas que he realizado.

Ante todo quiero resaltar que el producto final es una prueba de concepto, con lo que en muchos casos no se cumple el compromiso de ser universalmente genérico, es más, muchas de las decisiones tomadas y la escasez de recursos tanto humanos, temporales y de conocimiento han llevado en algunos casos a proponer soluciones "ad-hoc". Sea como fuere, funciona.

Sitúo el contexto. En la primera parte hemos visto la problemática y qué tipo de solución queremos. En la segunda, he presentado las herramientas disponibles, tanto conceptuales como programas. Es momento de juntarlo todo y fabricar la solución.

10.1. Caso de uso

La mejor forma de comprender el funcionamiento es empezar por algo general para luego ir concretando cada una de las partes. Recomiendo al lector que mantenga en mente este ejemplo durante toda esta tercera parte.

El caso de uso que propongo es extremadamente básico y no difiere mucho del visto en el capítulo 3. En la figura 10.1 se puede ver la arquitectura que propongo.

Veamos los pasos fundamentales.

- El usuario abre su ordenador en una zona con cobertura eduroam e introduce sus credenciales para conectarse. Se los manda al punto de acceso.
- El punto de acceso los reenvía al servidor RADIUS, que los validará con los que tiene en su almacén de datos.
- Si las credenciales son correctas, el servidor RADIUS pide una identidad digital al proveedor de identidad para ese usuario y manda un mensaje de aceptación al punto de acceso junto con la identidad digital.

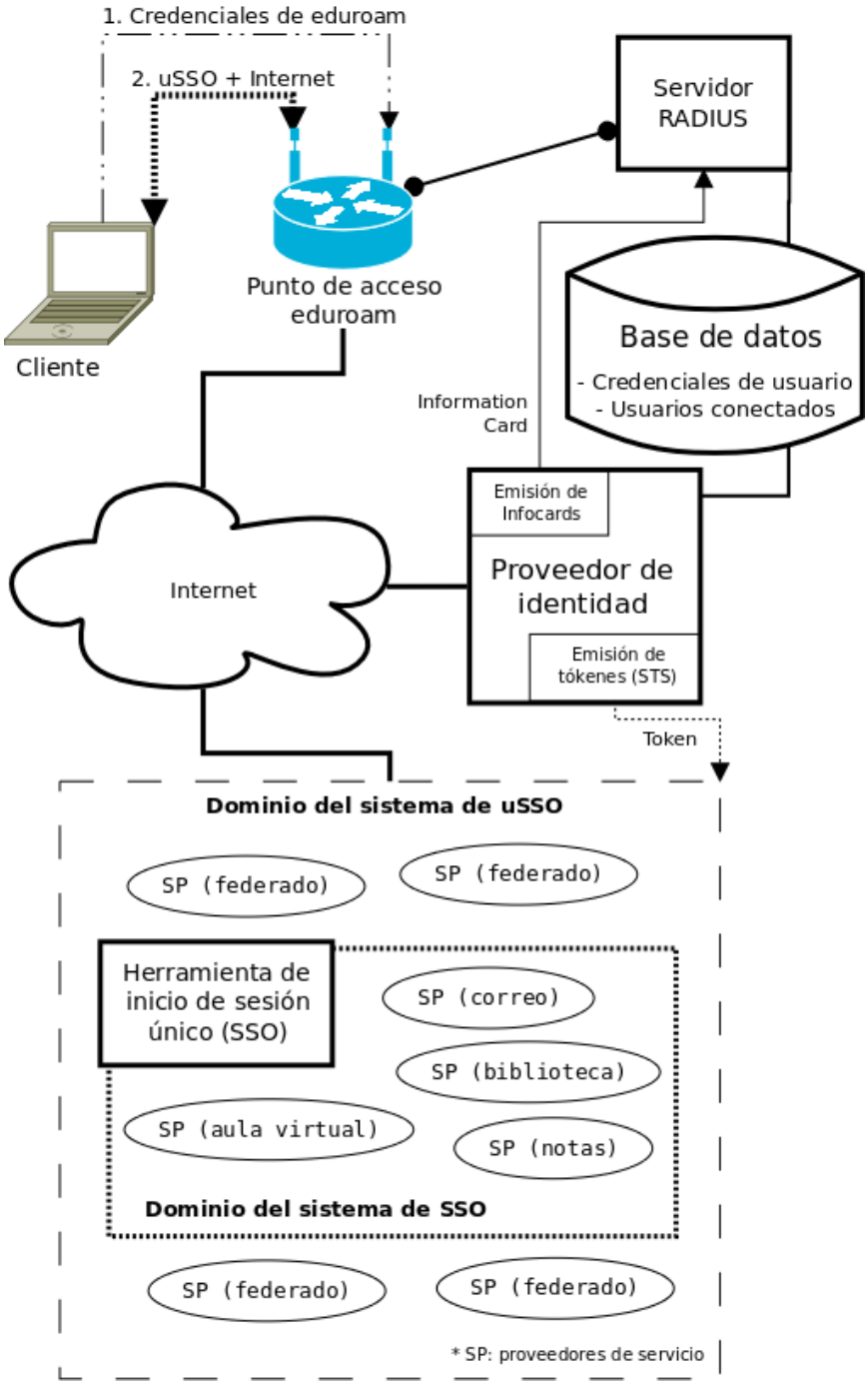

Figura 10.1: Arquitectura de la solución.

- El punto de acceso conecta al cliente a la red y le proporciona la identidad digital, una "Information Card". En este momento, el usuario ya tiene un inicio de sesión único unificado y en consonancia, no se le pedirán sus credenciales relacionadas con esa identidad digital.

- Si el usuario accede a un servicio dentro del dominio del sistema uSSO (dominio del metasistema), en la imagen "SP (federado)", dicho servicio le pedirá una identidad digital, le saltará en pantalla el selector de identidad y deberá seleccionar la que proceda (en este caso la nueva identidad que ha obtenido con eduroam).

- Si el usuario accede a un servicio dentro del dominio del sistema de SSO y no ha iniciado sesión, se le redirigirá a la herramienta (simpleSAMLphp) que gestiona el dominio. Esta herramienta le pedirá una identidad digital, proceso análogo al de acceder a un SP federado, e iniciará sesión. A partir de entonces tendrá una sesión iniciada en todos los servicios miembros de este dominio.

10.2. Flujos de información

Al ser una solución tan heterogénea tecnológicamente, me veo en la obligación de explicar las comunicaciones que se llevan a cabo entre las distintas partes en cada fase.

10.2.1. Fase de acceso

El cliente debe autenticarse para acceder a eduroam, la única parte a la que conecta física/directamente es al punto de acceso o AP. Lo normal es que sea de forma inalámbrica o WiFi.

Cliente-AP

El cliente envía sus credenciales de eduroam al punto de acceso. También envía información relacionada con sus credenciales de identidad digital (se comentará en los próximos capítulos ya que ahora no aporta nada).

AP-Servidor RADIUS

El punto de acceso está conectado con un servidor RADIUS al que envía la información del cliente. Como la conexión puede no ser directa (por ejemplo a través de internet), ambas partes establecen un secreto compartido para tener un canal cifrado y poderse comunicar mediante el protocolo RADIUS.

Jerarquía RADIUS

Aunque en la figura 10.1 no aparece, la información del cliente viajaría (en caso de ser necesario) por los servidores RADIUS de eduroam hasta llegar al que gestionase el dominio al que pertenece el usuario.

Servidor RADIUS-Base de datos

El servidor RADIUS verifica contra un almacén de datos (en la figura es una base de datos, pero puede ser un servidor ldap, un fichero ...) que las credenciales son correctas. Acto seguido, si las credenciales son correctas, registra en el mismo almacén que el usuario está conectado a la red eduroam junto con los datos de su identidad digital (la infocard del usuario queda activada).

Servidor RADIUS-Proveedor de identidad

Si las credenciales son correctas, el servidor RADIUS se comunica con el proveedor de identidad o IP y le pide una identidad digital o infocard para el usuario. Es evidente que entre ambos servidores existe una relación de confianza así como una conexión segura.

Servidor RADIUS-AP

Si las credenciales son correctas, el servidor RADIUS le mandará al AP un mensaje para autorizar al cliente así como la infocard del mismo. Si las credenciales no son correctas, el mensaje para el AP será de rechazar al cliente.

AP-Cliente

Si las credenciales han sido correctas, el AP conecta al cliente a eduroam y le manda un mensaje con su identidad digital. Si las credenciales no han sido correctas, simplemente lo rechaza.

NOTA: *Hay diversas formas de transmitir la identidad digital, note el lector que no he hecho referencia a ninguna, pues en este caso se transmite mediante referencia. No es la identidad en sí sino una dirección de dónde conseguirla lo que se le manda al usuario.*

10.2.2. Metasistema

Aquí el usuario ya está conectado a eduroam y tiene la infocard cargada en el selector, sin embargo no tiene una sesión iniciada.

Cliente-AP-Cliente

Es una conexión transparente, el punto de acceso encamina las peticiones del cliente hacia la red eduroam y desde ahí a internet si fuera necesario.

Cliente-SP (federado) 1

El SP le pide una identidad digital para iniciar sesión. En este caso, el cliente debería elegir la infocard que acaba de conseguir.

Cliente-IP (STS)
Acto seguido, el selector de identidad del cliente se comunica con el proveedor de identidad a traves del STS y presenta como credencial la información que envió al AP en la fase de acceso.

IP-Base de datos
El IP contrasta esa información con la que almacenó el servidor RADIUS de los usuarios conectados y valida el credencial.

IP STS-Cliente
Si las credenciales del cliente son correctas, le envía un token con su identidad digital.

Cliente-SP (federado) 2
El cliente manda el token al SP (federado) e inicia sesión en el mismo.

NOTA: *En la figura 10.1 aparece una flecha de puntos entre el STS del IP y el dominio del sistema de uSSO, esto es debido a que el token que emite es específico para ese conjunto de servicios, es por tanto una comunicación indirecta (a través del cliente).*

Cliente - SP (dentro del dominio del sistema de SSO)

El SP verifica si el cliente tiene iniciada una sesión de SSO, si es así le deja usar el servicio, pero si no, le redirige a la herramienta de inicio de sesión único.

Cliente - Herramienta de inicio de sesión único (SSO)
El cliente ha sido redirigido hasta aquí por un SP perteneciente al domino de esta herramienta y debe iniciar sesión (hacer Single Sign On). Esta herramienta autentica al cliente de forma análoga a un SP (federado), por lo que con que el cliente presente la identidad digital que consiguió al acceder a eduroam, iniciará sesión.

10.3. Consideraciones

Creo necesario aclarar unos cuantos conceptos y métodos que ahora mismo están poco definidos para mejorar la comprensión del lector.

10.3.1. Dominio del SSO

El dominio en el que se puede hacer inicio de sesión única está pensado para incluir a todas las aplicaciones web de la universidad. No se le exigirán credenciales de ningún tipo una vez se haya iniciado sesión. Ejemplos de estas aplicaciones web serían: el correo de la universidad (mediante webmail), los servicios de la biblioteca (reservas y renovaciones), el aula virtual (educación a distancia o clases no presenciales) y consulta de expediente académico y datos de secretaría.

Puesto que estas aplicaciones web necesitan una modificación para adaptarse a este método y son propias de cada entidad, el ámbito del sistema de SSO estará limitado a uno por universidad. No hay razón para pensar que yo tenga una cuenta de correo en una universidad a la que no pertenezco, además existen otras cuestiones como la carga del sistema y el número de sesiones abiertas (una por servicio del dominio) que hay que gestionar.

Imaginemos que soy alumno de la UAH (España) y me voy de erasmus a la RU (Dinamarca). La arquitectura que defino tendrá un dominio de SSO en la UAH y otro en la RU, en ambas podré hacer SSO, que serán independientes, esto es que tendré que suministrar mi identidad digital al sistema de la UAH y una segunda vez al de la RU.

10.3.2. Dominio del uSSO

Un efecto secundario del metasistema es que diversos tipos de aplicaciones que interpreten la tecnología de servicios web pueden consumir identidades digitales para autenticar usuarios.

El ejemplo más inmediato es el de una página web que contenga publicaciones científicas y tenga un acuerdo con la universidad para ofrecer un servicio de visualización gratuito para alumnos de la misma, con aceptar identidades digitales de alumnos de la universidad, tendría el problema de la autenticación resuelto sin tener que filtrar por rangos de direcciones IP.

Ejemplo(21) Un investigador español, colaborador de una universidad del Reino Unido, necesita tiempo de cómputo de su superordenador para realizar unas simulaciones. En vez de hacer el viaje hasta la universidad para llegar al ordenador, puede reservar la máquina unas horas y acceder remotamente a ella. El servidor de acceso (VNC o SSH) le pedirá una identidad digital antes de permitirle usarla. Al ser miembro de la comunidad eduroam, el científico no tendrá que hacer nada "raro" para acceder, solamente elegir su identidad en eduroam en el selector de identidad.

El ejemplo anterior demuestra la flexibilidad del sistema ante un caso que puede ser bastante complejo porque el proyecto puede incluir investigadores a escala mundial (pero siempre dentro de la confederación de eduroam).

Surgen dos grandes preguntas.

¿Cuántos proveedores de identidad puede haber?

Lo ideal sería uno por universidad/entidad asociada a eduroam (por temas de gestión de credenciales) cuyo certificado x509 estuviera respaldado por una autoridad de certificación (una CA para toda la confederación de eduroam) a través de una cadena de confianza que podría tener una jerarquía similar a la de los servidores RADIUS.

¿Puede funcionar la autenticación de cualquier programa con la tecnología de Information Cards?

Actualmente no, potencialmente sí. Habría que adaptar el programa para que se comunicara con el selector de identidad en el caso de que fuera un cliente o para que pudiera consumir tókenes en el caso de ser un servidor. Uno de los temas que se trataron en el tnc2009 fue precisamente esta problemática. Las opciones que más atraen mi atención de momento son la creación de librerías con estas funcionalidades para que las usen los clientes y la creación de una capa en el sistema operativo que ofrezca lo mismo. Ambas propuestas tienen sus pros y sus contras.

10.3.3. Automatizaciones

Procedo a comentar los automatismos que más se pueden echar en falta en la explicación del caso de uso.

Credencial de la tarjeta

Quedó dicho en el capítulo 8.2.6 que para hacer uso de una information card y obtener un token con una identidad digital hay que autenticarse ante el proveedor de identidad, suministrarle unas credenciales. ¿No supone esto que para acceder a cualquier SP (federado) tenemos que volver a poner nuestras credenciales de eduroam? ¡Esto arruina el uSSO! La respuesta es sencilla, credenciales sí hay que suministrar, o al menos una prueba de su posesión, pero no lo haremos nosotros directamente.

Existe un caso de autenticación con infocard mediante tarjetas autoemitidas. El credencial de uso de una tarjeta gestionada (la de eduroam) es otra tarjeta que nosotros nos hemos generado anteriormente (la analogía es hacernos nuestra propia contraseña). Es la información sobre esta tarjeta autoemitida la que se manda junto con las credenciales al punto de acceso en la fase de acceso, para que el servidor RADIUS lo almacene como credencial de uso en el registro de usuarios conectados y el servidor de identidad genere una infocard en consonancia.

Cuando un SP (federado) pide una identidad digital, al usuario le salta el selector de identidad, selecciona la infocard de eduroam y el proceso está completado. Internamente existen unas comunicaciones en las que el proveedor de identidad verifica que el usuario posee la tarjeta autoemitida. ¡La autenticación es automática!

Conector

El conector es otro automatismo fundamental, es una pieza lógica expresamente creada para esta arquitectura y dependiente del software que se utiliza. El concepto se materializa en un programa que corre sobre la máquina del cliente.

Cuando el cliente quiere iniciar su sesión, utiliza directamente el conector (en contraposición a cualquier gestor de red). Éste le pide sus credenciales de eduroam y qué tarjeta autoemitida (a través del selector de identidad) quiere usar como credencial de la infocard que se le generará. Acto seguido se conecta al punto de acceso, evalúa la respuesta y carga la infocard de eduroam en el selector de identidad.

La función principal es la de incrementar la usabilidad del sistema, no olvidemos que eduroam incluye tanto a miembros de carreras técnicas como a otro personal no íntimamente familiarizado con las nuevas tecnologías. Potenciar este objetivo al máximo es clave para la aceptación por parte del usuario y su consiguiente expansión.

Internamente el conector se encarga de las comunicaciones con el cliente de 802.1X, a partir de ahora conocido como suplicante, y con el selector de identidad, es decir, hace de puente para transmitir datos entre una aplicación que se comunica en la capa de red (el suplicante, 802.1X) y otra que lo hace en el de aplicación (selector de identidad, Web Services/HTTP). Dicha comunicación será analizando los registros o logs que proporcionan ambas aplicaciones y reconfigurando según la ocasión sus ficheros de comunicación. También se encarga de la configuración inicial del selector de identidad y de aportarle los certificados necesarios para su correcto funcionamiento. El cliente se despreocupa de cualquier aspecto técnico.

10.4. Conclusiones

Por último quiero hacer una autocrítica de todo lo expuesto. Son cuestiones que un lector bien informado de la materia podría echarme en cara ahora mismo.

¿Estoy haciendo realmente uSSO?
Introduzco las credenciales de eduroam y una tarjeta autogestionada, me conecto y se supone que ya tengo sesión iniciada. Pero... ¿por qué me piden una identidad digital los servicios de la universidad (correo, notas...) si ya saben quién soy? ¿Estoy haciendo realmente inicio de sesión único unificado?

Respuesta
La respuesta más corta es que no te piden credenciales, sino una identidad digital, luego la posesión de la infocard de eduroam es por así decirlo el testigo de que eres el propietario de la sesión. Sesión porque cada vez que te conectas a la wifi obtienes una tarjeta distinta.

La cuestión del automatismo al que se hace referencia, el de los servicios de la universidad, tiene que ver con el dominio del SSO, y aquí existe un problema enorme. Como se vio en el capítulo 2.3.2 el SSO está pensado para aplicaciones web, esto es que se ven a través de un navegador. Supongamos que el suplicante ha sido capaz de recibir la información necesaria para hacer SSO. ¿Qué navegador de los que tiene el usuario debe elegir? ¿Cómo le comunica al navegador esos datos?

Otro problema es el tamaño de esos datos, el protocolo RADIUS es AAA (Authentication, Authorization, Accounting) Autenticación, Autorización y Contabilidad. En ningún momento dice transmisión de mensajes ni información extra. Suficiente es que se puede meter una referencia a una infocard.

Transmisiones
Ya, pero si metes una infocard, también podrías poner la referencia a los datos de SSO.

Respuesta
La información extra que se puede transmitir entre el cliente y el servidor RADIUS está muy limitada. Potencialmente puedes mandar lo que quieras, pero me he atenido a estándares y protocolos. Hay un tema dedicado a la comunicación que he hecho y el lector verá que meter la referencia a la infocard ha sido un auténtico quebradero de cabeza por las modificaciones que hay que realizar en los programas (aun usando estándares).

¿Cuántos dominios de SSO hay?
Supongamos que puedo mandar la información necesaria para el automatismo del SSO. El servidor RADIUS debería obtenerla de la herramienta de SSO, ahora bien ¿cuántos dominios de SSO existen? Es trivial decir que el de la universidad. ¿Pero y si el usuario está de Erasmus? Esto solo nos lleva a más complicaciones.

¿Merece la pena el automatismo SSO?
En términos usabilidad/problemas, no. Como se ve, el acoplamiento crece exponencialmente con cada "mejora", sin duda el selector de identidad es la opción más limpia, el único acoplamiento viene por parte del conector.

¿Cuál es la ventaja entonces?
Si me hubiera limitado al automatismo, no existiría la figura del SP (federado) y no gozaría de las ventajas del metasistema. ¡Reléete el capítulo del metasistema!

¿Cuál es el principal problema?
El principal problema es conectar el metasistema con eduroam y saber definir qué tiene que hacer cada parte. Donde parar, discernir por qué el automatismo SSO no es buena idea y tomar decisiones estratégicas (cuyo impacto no es inmediato).

Es evidente que podemos pedir más, más comodidad, más usabilidad, ¿pero a qué coste?

Por último recomiendo mirar los siguientes enlaces:

- An Infocard-based proposal for unified single sing on - Página de la presentación de este proyecto en la conferencia de Terena del 2009 (TNC2009 Málaga)
  http://tnc2009.terena.org/schedule/presentations/show9a7d.html?pres_id=44
- Integrated in-Eduroam InfoCard identity metasystem for S.S.O. - Demostración en vídeo que hice para el TNC2009 de cómo funciona esta arquitectura.
  http://it.aut.uah.es/enrique/research/demo.html

Capítulo 11

Implementación del metasistema

11.1. El selector de identidad

Empezamos la implementación de la solución por el selector de identidad. Es el programa que se ejecutará en la máquina del cliente.

En el capítulo 8.1.1 se vieron las características que debía reunir así como qué opciones existían. Puesto que todas las opciones deberían funcionar de una forma similar, es momento de evaluar el valor añadido de la opción seleccionada.

La gran consideración a tener en cuenta es que el entorno de desarrollo sobre el que trabajo es GNU/Linux, en concreto la distribución Debian. Con esto descarto las opciones Microsoft CardSpace y Azigo por ser para el sistema Windows, y openinfocard por no ser un proyecto maduro cuando se realizó el proyecto. Por consiguiente me quedo con el proyecto DigitalMe de Bandit.

DigitalMe es un selector de identidad de código abierto y libre, lo que significa que potencialmente lo puedo compilar para la plataforma que desee, indagar en sus funcionalidades o adaptarlo a mis necesitades, y por último no tener que pagar un duro (está sujeto a la licencia "Eclipse Public License - v1.0" de Novell, Inc.).

Soy consciente de que tampoco es (o mejor dicho, era) una opción madura en comparación con Microsoft CardSpace pero el valor añadido de ver sus entresijos y los mensajes de depuración que es capaz de mostrar empequeñece de sobremanera a la opción de Windows. Tal es su importancia, que sin esta herramienta, el presente proyecto hubiera sido planteado de manera totalmente distinta o, directamente, no planteado.

11.1.1. Instalación

Explico cómo instalar el selector de identidad y configurarlo para su uso con el navegador Firefox/Iceweasel en una distro Debian.

Selector de identidad

El primer paso es buscar la versión del selector de identidad que más se ajuste a nuestro sistema. Vamos a la página de descargas del proyecto DigitalMe:
http://code.bandit-project.org/trac/downloads.

En mi caso, la opción preferente fue compilar la la última versión disponible del código. Tarea imposible de forma normal, instalando las librerías necesarias y compilando, ya que el compilador mostraba errores de programación.

La segunda opción era bajar un paquete específico para mi distribución. La más parecida que existe es Ubuntu (por estar basada en Debian), pero tampoco resultó.

Antes de desistir probé a bajarme un paquete rpm para Fedora (distribución linux patrocinada por la empresa Red Hat), aplicarle el programa alien y crear así un paquete deb para Debian. Tuve suerte con la versión 0.5.2402-2.1. En la figura 11.1 se ve la información que el comando `dpkg` muestra sobre el paquete.

```
Package: digitalme
Version: 0.5.2402-2.1
Architecture: i386
Maintainer: Samuel Muñoz Hidalgo <samuel.mh@gmail.com>
Installed-Size: 5084
Section: alien
Priority: extra
Description: DigitalMe
 Selector de identidad especialmente empaquetado para
  mi proyecto de fin de carrera.

 Página oficial del proyecto DigitalMe:
  http://code.bandit-project.org/trac/wiki/DigitalMe

 Compatible con Microsoft CardSpace.
 (Converted from a rpm package by alien version 8.73.)
```

Figura 11.1: Información sobre el paquete para Debian digitalme.

El paquete que creé se localiza en el fichero `/SW/DigitalMe/DMe/selector.deb` del CD-ROM que se adjunta con el proyecto. Nos situamos en la carpeta que lo contiene y lo instalaremos con permisos de superusuario con la orden siguiente.

consola

```
1 # dpkg -i DMe_selector.deb
```

Extensión para Firefox/Iceweasel

Como navegador he seleccionado Firefox o cualquier otro basado en él. En Debian por defecto se usa un proyecto paralelo o "fork" llamado Iceweasel. El modo de funcionamiento de ambos es similar.

Hay que instalar una extensión o "plugin" que le permita al navegador comunicarse con el selector de identidad. Para ello vamos a la página de descargas del proyecto DigitalMe, la de la sección anterior, y seleccionamos la pestaña firefox. Nos bajamos un fichero xpi. En mi caso fue el fichero `digitalme-firefox-0.5.2571.xpi`.

¡No funciona!

Al intentar instalarlo, el navegador muestra un error como el de la figura 11.2 diciendo que no es compatible con nuestra versión del navegador. Como ya se habrá dado cuenta el lector, tengo solución para casi todo (o de otra forma no estaría escribiendo esto).

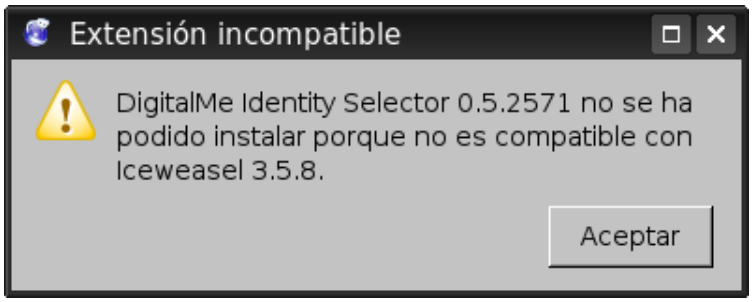

Figura 11.2: Mensaje de extensión incompatible con la versión del navegador.

Nos descargamos el fichero xpi: botón derecho "Cuardar como". Lo abrimos con un compresor zip, en mi caso uso Ark. Editamos el fichero `install.rdf` y a poco que miremos, vemos unas líneas como muestra la figura 11.3 en las que debemos editar el número de versión. En mi caso no funcionaba porque mi versión de navegador 3.5.8 es mayor que la máxima permitida, así que cambié el valor a 4.0.

```
...
<!-- Firefox -->
<em:targetApplication>
   <Description>
      <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
      <em:minVersion>1.0</em:minVersion>
      <em:maxVersion>3.5</em:maxVersion>
   </Description>
</em:targetApplication>
...
```

Figura 11.3: XML con las restricciones de la extensión.

Guarda el fichero y recomprime todo como un zip, pero mantén la extensión xpi. Ábrelo con el navegador, como muestra la figura 11.4. Aparecerá una ventana para confirmara la instalación, figura 11.5, y después habrá que reiniciar el navegador, figura 11.6.

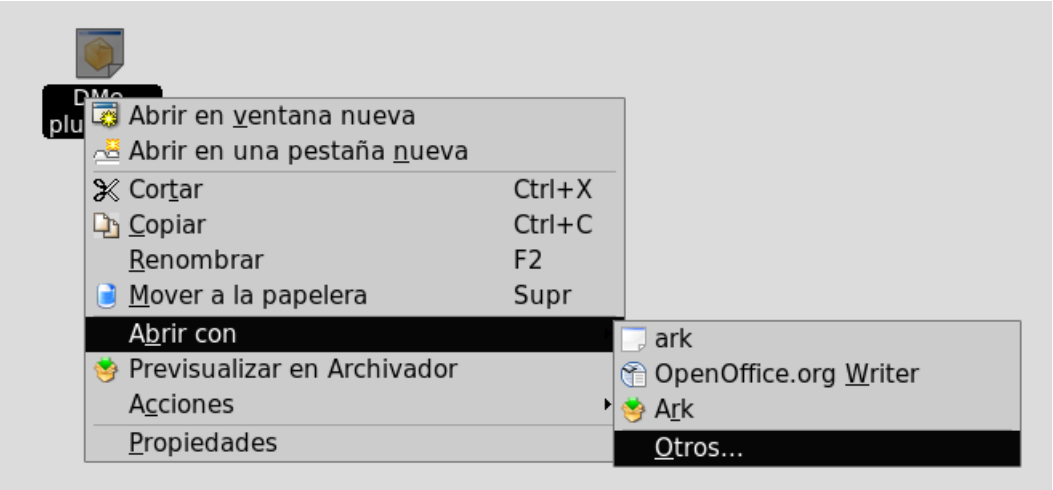

Figura 11.4: Abrimos la extensión con el navegador.

Figura 11.5: Ventana para confirmar la instalación.

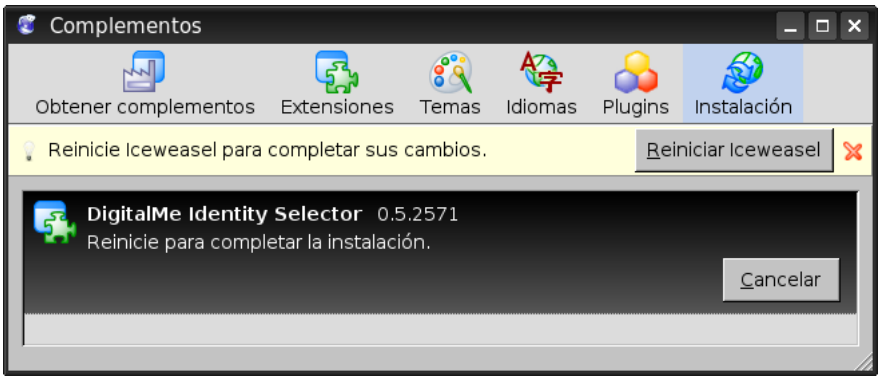

Figura 11.6: Ventana de reinicio del navegador.

11.1.2. Configuración y ficheros

Una vez instalado todo hay que tener en cuenta de qué forma se puede configurar todo para tener un rendimiento ajustado a nuestras necesidades.

Selector de identidad (arranque y certificados)

Aunque el paquete del selector de identidad no tiene dependencias, es posible que necesite de programas adicionales para funcionar.

Lanzaremos el selector de identidad con la siguiente orden.

consola

```
1 $ digitalme
```

Prestaremos atención a los posibles mensajes de error que aparezcan. Un programa necesario es el `gnome-keyring` que será el almacén de tarjetas y claves. La primera vez que iniciemos el programa, intentará crear un anillo (almacén cifrado) protegido por contraseña. Las siguientes veces que iniciemos el selector, solo la primera vez por cada sesión que arranquemos en el sistema, se nos preguntará por esa contraseña para cargar nuestras tarjetas. La figura 11.7 muestra la ventana con la que arranca el selector.

La última consideración para tener a punto el selector es conocer el fichero en el que guarda los certificados que se utilizan para firmar tarjetas y mensajes.
Dicho fichero se localiza en: `/usr/share/digitalme/certs/ca-bundle.crt` Ese fichero debería de contener los certificados codificados en base64 de las autoridades de certificación en las que confiamos, por ejemplo aquella que firma el certificado del proveedor de identidad.

Se puede añadir un certificado, en este caso "certificado.crt" al fichero (siendo superusuario) con la siguiente orden.

consola

```
1 # cat certificado.crt >> /usr/share/digitalme/certs/ca-bundle.crt
```

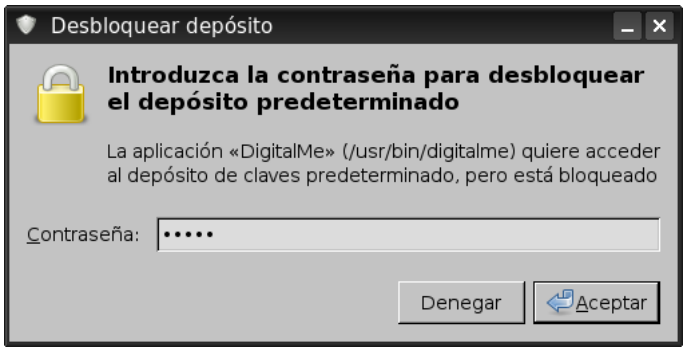

Figura 11.7: Ventana de reinicio del navegador.

Extensión del navegador

La configuración de la extensión en el navegador se realiza de la siguiente forma.

1. En el menú pricipal del navegador pulsa en "Herramientas" y luego en "Complementos".
2. Se abrirá una ventana, pulsa en Extensiones.
3. Selecciona de la lista una que se llama `DigitalMe Identity Selector` y dale a Preferencias. Ver figura 11.8.
4. Aparecerá una ventana de configuración como la que muestra la figura 11.9.

La ventana de configuración consta de tres campos:

- **General Settings.**
  Es un campo de texto donde hay que meter la ruta al ejecutable. En mi caso es `/usr/bin/digitalme`. Si no supiéramos la ruta del ejecutable, podríamos probar la siguiente orden de consola. La salida de esa orden es la ruta del ejecutable.

consola

```
1          $ which digitalme
2  /usr/bin/digitalme
```

- **Security Settings.**
  Se puede seleccionar la opción de "Secure Desktop" o escritorio seguro. Cuando se lance el selector de identidad, aparecerá como una ventana en primer plano y el escritorio se bloqueará. Esto focaliza la atención del usuario en el proceso de autenticación.
- **Debug Settings.**
  Permite especificar un fichero de depuración.

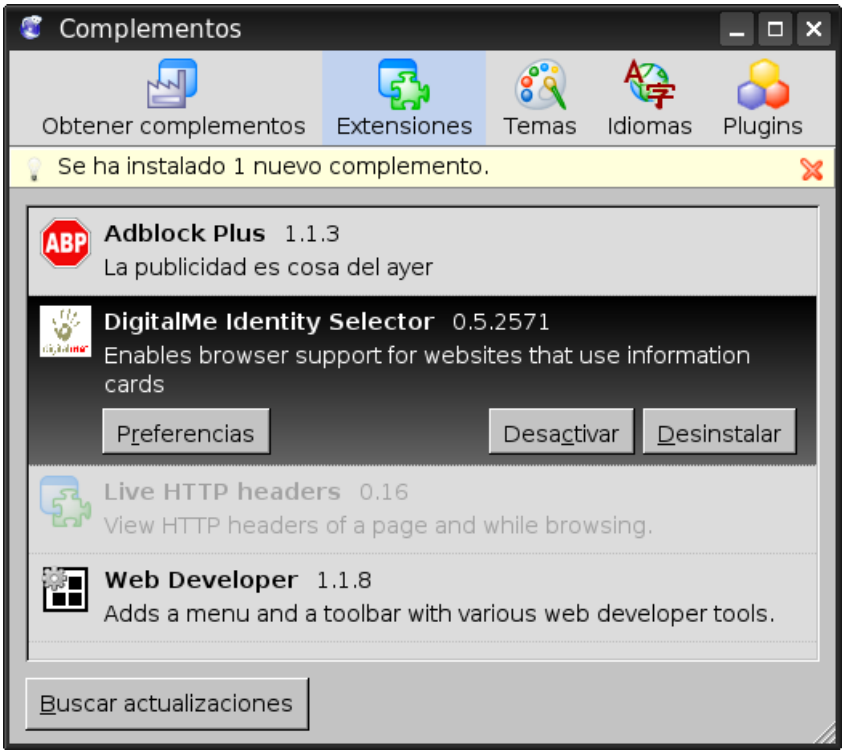

Figura 11.8: Lista de complementos instalados.

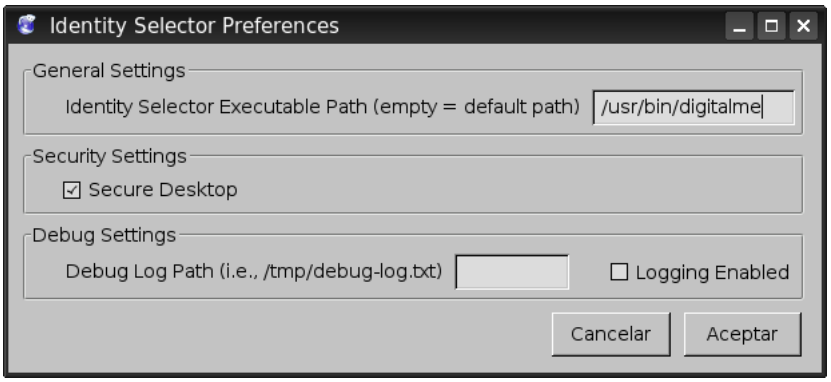

Figura 11.9: Configuración de la extensión.

11.1.3. Funcionalidades

Resumo las funcionalidades más importantes del selector de identidad

Importar tarjeta

Si nos descargamos una tarjeta se supone que es para usarla con nuestro selector de identidad. Es posible que ni siquiera queramos almacenarla explícitamente sino importarla, para ello, en la pantalla de descarga del navegador, le daremos a "Abrir con" en vez de "Guardar Archivo". En el campo de texto "Lugar:" pondremos la ruta del digitalme `/usr/bin/digitalme`.

Es posible que nos aparezca un mensaje de error como el de la figura 11.10. Se debe a que no tenemos importados los certificados necesarios. Consigue el certificado de la autoridad de certificación e impórtalo como se vio en 11.1.2.

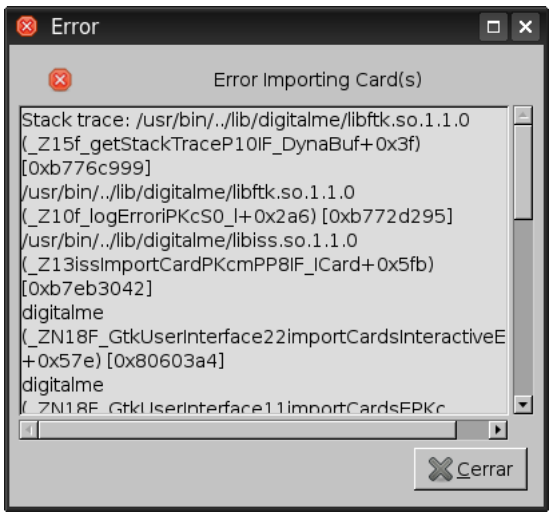

Figura 11.10: El selector muestra un error al importar la tarjeta.

Si todo va bien aparecerá una ventana como la de la figura 11.11. Aceptamos y ya tenemos nuestra tarjeta lista para usar.

Es posible invocar al selector desde consola para que cargue una tarjeta (en este caso infocard.crd) con la siguiente orden.

consola

```
1 $ digitalme infocard.crd
```

Uso normal

La funcionalidad más usada del selector será la de permitirnos elegir una identidad digital o infocard, conectar con el proveedor de identidad y mostrarnos una pantalla para introducir nuestros credenciales, conseguir un token y posteriormente gestionar el envío de ese token.

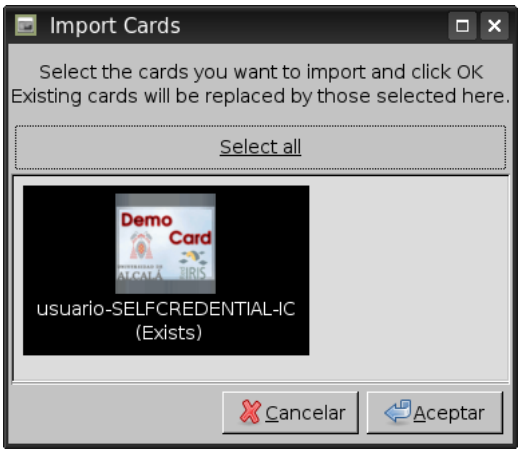

Figura 11.11: El selector importa correctamente la tarjeta.

En un principio no debería de darnos problemas más allá de los vistos con los certificados.

Depuración

Pero los problemas existen y este es un proyecto de innovación, con lo que sería de gran ayuda que el selector tuviera un modo en el que dijera en todo momento qué está sucediendo.

Con la siguiente orden puedo invocar al selector en modo de depuración. Mostrará por consola mensajes de funcionamiento, advertencia y error.

consola

```
1 $ digitalme --loglevel=debug
```

Como la salida de la orden anterior produce una gran cantidad de información, es posible que queramos guardarla en un fichero. Hay que tener en cuenta que en modo debug, la aplicación escribe en la salida estándar de error (stderr), por lo que habrá que leer del descriptor de fichero 2. La siguiente orden guarda la salida en un fichero llamado "dmelog.txt" en la carpeta personal.

consola

```
1 $ digitalme --loglevel=debug 2> ~/dmelog.txt
```

Ajustando la depuración

En la parte de configuración de la extensión he comentado una opción, debug settings, con la que es posible guardar la salida de depuración del selector cuando se invoca desde el navegador.

Cuando hice el desarrollo del proyecto, esta opción no existía (era una extensión anterior) y tuve que ingeniármelas para recrear esta funcionalidad. Tampoco era posible especificar la ruta del ejecutable ya que la extensión llamaba directamente en consola al comando `digitalme`.

Mi intención era crear una envoltura que llamara al selector en modo depuración con la salida redirigida a un fichero. Esto no es problema, como hemos visto unas líneas arriba. Lo verdaderamente extraño fue hacer que la extensión llamara a la envoltura.

Si revisamos la variable de consola $PATH, veremos que es una lista de directorios donde se buscan los comandos que ejecutamos. Mi selector estaba en `/usr/bin`, pero justo antes tengo otro directorio `/usr/local/bin`. Si llamo a mi envoltura igual que el ejecutable que invoca la extensión del navegador y la meto en un directorio de ejecutables con precedencia sobre el selector, el problema está resuelto sin tener que tocar nada de código.

Sé que esta solución está obsoleta, pero quién sabe si habría que utilizarla más adelante.

11.2. Funcionalidades básicas

La primera parte clave en el desarrollo de la solución es el disponer de una serie de funciones que permitan manejar las operaciones básicas relacionadas con las Information Cards.

El lenguaje web utilizado es PHP, así que todo el código (o pseudocódigo) que comente seguirá su sintaxis. Las funciones están distribuidas según los roles del metasistema, habrá una parte para la RP y otra para el proveedor de identidad o IP.

Como el desarrollo en general de la prueba de concepto se realizó a gran velocidad para poder indagar en partes más complejas que esta, no se ha podido producir una librería completa ni apta para su explotación. Tarea que queda pendiente para quien se interese de verdad por el tema.

11.2.1. Consideraciones

La primera cosa a tener en cuenta es que el código fue pensado para ser parte de un módulo integrado en la herramienta de SSO simpleSAMLphp. Esto genera cierto acoplamiento con las estructuras de datos que manejo, pero como dichas estructuras son de alto nivel (tipos de datos compuestos como diccionarios o vectores indexados), las comentaré en la medida de lo que crea necesario para explicar su función. Si se desea saber más de detalles técnicos, será necesario mirar el código. Otra consecuencia de este acoplamiento es la anteriormente mencionada pseudolibrería, todos los ficheros a los que hago referencia están en la carpeta `simplesaml/modules/InfoCard/lib`.

La segunda consideración tiene que ver con los métodos de autenticación que se pueden usar con las infocards. Solo he implementado dos: el de usuario-contraseña y el de tarjeta autoemitida. Como únicamente necesito el método de tarjeta autoemitida, será en el que me centraré en las explicaciones.

En la página http://dev.eclipse.org/mhonarc/lists/higgins-dev/msg01330.html se puede ver un caso de uso con el método de tarjeta autoemitida como credencial de usuario. Sin duda fue una gran ayuda para comprender cómo funciona el mecanismo, su lectura es altamente recomendable.

11.2.2. Funcionalidades del IP

Las funcionalidades del proveedor de identidad se encuentran en el fichero `STS.php`.

Generar tarjeta

La funcionalidad de crear tarjetas fue la primera que desarrollé ya que es una comunicación unidireccional del IP al selector de identidad y sirve para probar este último. También es una excelente toma de contacto con distintos aspectos de la arquitectura.

FUNCIÓN: `createCard(ICdata,ICconfig)`

PARÁMETROS DE ENTRADA:

`ICdata:` Información referida al usuario que contendrá la tarjeta: CardId, CardName, CardImage, TimeExpires, información sobre el credencial.

`ICconfig:` Configuración global del proveedor de identidad: emisor, web del STS, web del MEX, tipo de autenticación, tipos de claims soportados y política de privacidad.

ACCIONES:

- Construcción del XML con la información de la infocard. Sigue la estructura de la figura 8.8.
- Firma de la infocard.
  - Construcción del resumen, bloque SignedInfo.
    Se canonicaliza el XML de la infocard.
    Se resume con la función SHA1.
    Se codifica en base64.
    Se mete en una estructura XML similar a la de la figura 11.12.
  - Firma del resumen.
    Se canonicaliza el bloque SignedInfo.
    Se firma con la clave privada del STS.
    Se codifica en base64.
  - Empaquetando todo, bloque Signature.
    Se crea un bloque Signature
    Se agrega el bloque SignedInfo.
    Se agrega el bloque SignatureValue con la firma dentro de él.
    Se agrega un bloque KeyInfo con la cadena de certificados.
    Se agrega el bloque XML de la infocard.
- La función devuelve un bloque XML como el de la figura 11.13 que será la infocard.

AUTENTICACIÓN:

Para el método de autenticación con tarjeta autoemitida, solo es necesario introducir como credencial el PPID de la tarjeta autoemitida, queda un bloque UserCredential como el de la figura 11.14.

```
<SignedInfo  xmlns="http://www.w3.org/2000/09/xmldsig#">
   <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
   <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
   <Reference URI="#IC01">
      <Transforms>
         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>base64(SHA1(canonicalizar($infocardbuf)))</DigestValue>
   </Reference>
</SignedInfo>
```

Figura 11.12: Bloque XML SignedInfo.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
      Resumen de la infocard
   </SignedInfo>
   <SignatureValue>
      base64(firmar con clave privada del STS (bloque SignedInfo))
   </SignatureValue>
   <KeyInfo>
      <X509Data>
         <X509Certificate></X509Certificate>
      </X509Data>
   </KeyInfo>
   Información de la infocard
</Signature>
```

Figura 11.13: Bloque XML Signature.

```
<UserCredential>
   <DisplayCredentialHint>
   </DisplayCredentialHint>
   <SelfIssuedCredential>
      <PrivatePersonalIdentifier>
         PPID
      </PrivatePersonalIdentifier>
   </SelfIssuedCredential>
</UserCredential>
```

Figura 11.14: Bloque XML UserCredential.

Generar token

El servicio fundamental del proveedor de identidad es el de generar testigos o tókenes con las identidades digitales de los usuarios. Como vimos en el capítulo 6.2.6, este servicio se denomina STS o Servicio de Tókenes Seguros (Secure Token Service).

Cuando empecé con el proyecto no estaba muy versado en el tema de los servicios web, miré por recomendación de mi tutor en un programa ya existente llamado "Carillon STS". Es un STS de pruebas (versión actual 0.02) programado en PHP que tiene tiene la capacidad de funcionar con selectores de identidad como Microft CardSpace. En mi experiencia, me sirvió para ver cómo funcionaban realmente los servicios web y utilicé su código para las primeras pruebas. Otra ayuda a destacar fue el servicio MEX o de intercambio de metadatos que implementa. No sería correcto decir que el código que he escrito es una modificación de Carillón, ya que aunque en un principio lo usé, cuando mi conocimiento en la materia se hizo más profundo, reescribí toda la estructura del programa y ajusté los servicios web a las necesidades específicas.

Página web de Carillon STS: `http://www.carillon.ca/tools/demo-sts.php`.

FUNCIÓN: `createToken(claimValues,config,relatesto)`

PARÁMETROS DE ENTRADA:

`claimValues:` valores de los claims.

`config:` configuración del IP

`relatesto:` identificador de la petición de token RST hecha por el selector de identidad.

ACCIONES:

- Crea una envoltura, un mensaje de respuesta de servicios web como el de la figura 11.15
- La estructura XML consta de una cabecera o Header y un cuerpo o Body.
- El contenido del cuerpo es la respuesta RSTR (Request Security Token response) del STS a la petición de token, RST o (Request Security Token) hecha por el selector de identidad.
- Devuelve el mensaje de respuesta o token creado.

```
<?xml version="1.0"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
 xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
 xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
   <S:Header>
      <wsa:Action wsu:Id="_1">
         http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
      </wsa:Action>
         <wsa:RelatesTo wsu:Id="_2">
            Identificador de la RST
         </wsa:RelatesTo>
         <wsa:To wsu:id="_3">
            http://www.w3.org/2005/08/addressing/anonymous
         </wsa:To>
         <wsse:Security S:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="_6">
               <wsu:Created>Fecha de creación</wsu:Created>
               <wsu:Expires>Fecha de expiración</wsu:Expires>
            </wsu:Timestamp>
         </wsse:Security>
   </S:Header>
   <S:Body wsu:Id="_10">
         RSTR (WS-TRUST)
   </S:Body>
</S:Envelope>
```

Figura 11.15: Envoltura de la respuesta.

Request Security Token Response (RSTR)

Es la implementación del esquema WS-TRUST (ver 6.2.6) para dar una respuesta RSTR.

FUNCIÓN: `RequestSecurityTokenResponse (claimValues,config,assertionid,created,expires)`

PARÁMETROS DE ENTRADA:

`claimValues:` valores de los claims.

`config:` configuración del IP

`assertionid:` identificador único de la respuesta.

`created:` fecha de creación del token.

`expires:` fecha límite de uso del token.

ACCIONES:

- Crea el bloque XML RequestSecurityTokenResponse encargado de la respuesta RSTR. Ver figura 11.16.
- Rellena las restricciones temporales.
- Rellena los identificadores de respuesta.
- Rellena el bloque RequestedDisplayToken que contendrá información para el selector de identidad con el valor de los claims de la identidad digital.
- El contenido de RequestedSecurityToken es la identidad digital o token SAML.

```
<wst:RequestSecurityTokenResponse>
   <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
   <wst:LifeTime>
      <wsu:Created>Fecha de creación</wsu:Created>
      <wsu:Expires>Fecha límite de uso</wsu:Expires>
   </wst:LifeTime>
   <wst:RequestedSecurityToken>
      Identidad digital, token SAML
   </wst:RequestedSecurityToken>
   <wst:RequestedAttachedReference>
      <wsse:SecurityTokenReference>
         <wsse:KeyIdentifier
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID">

            Identificador de respuesta
         </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
   </wst:RequestedAttachedReference>
   <wst:RequestedUnattachedReference>
      <wsse:SecurityTokenReference>
         <wsse:KeyIdentifier
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID">

            Identificador de respuesta
         </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
   </wst:RequestedUnattachedReference>
   <ic:RequestedDisplayToken>
      <ic:DisplayToken xml:lang="en-us">
            <ic:DisplayClaim Uri="http://...esquema del claim">
               <ic:DisplayTag>Nombre del claim</ic:DisplayTag>
               <ic:DisplayValue>Valor del claim</ic:DisplayValue>
            </ic:DisplayClaim>
            ...
      </ic:DisplayToken>
   </ic:RequestedDisplayToken>
</wst:RequestSecurityTokenResponse>
```

Figura 11.16: Bloque XML RSTR (respuesta de WS-Trust).

Aserción SAML (identidad digital)

Esta funcionalidad crea un token firmado por el IP con la identidad digital del usuario expresada en forma de aserción SAML.

FUNCIÓN: `saml_assertion(claimValues,config,assertionid,created,expires)`

PARÁMETROS DE ENTRADA:

`claimValues`: valores de los claims.

`config`: configuración del IP

`assertionid`: identificador único de la respuesta.

`created`: fecha de creación del token.

`expires`: fecha límite de uso del token.

ACCIONES:

- Construcción de la aserción SAML.
  - Rellena las restricciones temporales.
  - Predicado SAML.
    Incluye el certificado del STS.
    Rellena los atributos (equivalente a claim en aserción SAML) de la aserción con los datos del usuario.
- Construcción de la firma.
  - Resumen
    Se canonicaliza la aserción temporal.
    Se resume con la función SHA1 y se codifica en base64.
    Este resumen se incluye en el bloque SignedInfo.
  - Firma
    Se canonicaliza el bloque SignedInfo.
    Se firma con la clave privada del STS.
    Se codifica en base 64.
    Se incluye esta información en una etiqueta SignatureValue.
- Se devuelve una estructura como la de la figura 11.17.

```
 <saml:Assertion MajorVersion="1" MinorVersion="0" AssertionID="Id
de la respuesta" Issuer="emisor" IssueInstant="fecha de creación"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
    <saml:Conditions NotBefore="fecha de creación" NotOnOrAfter="fecha de expiración" />
    <saml:AttributeStatement>
       <saml:Subject>
          <saml:SubjectConfirmation>
             <saml:ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
             </saml:ConfirmationMethod>
             <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
                <dsig:X509Data>
                   <dsig:X509Certificate>
                      Certificado del STS
                   </dsig:X509Certificate>
                </dsig:X509Data>
             </dsig:KeyInfo>
          </saml:SubjectConfirmation>
       </saml:Subject>
       <saml:Attribute AttributeName="nombre del claim" AttributeNamespace="URL del esquema">

          <saml:AttributeValue>Valor del claim</saml:AttributeValue>
       </saml:Attribute>
       ... más atributos
    </saml:AttributeStatement>
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
       <dsig:SignedInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" >
          <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />

          <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <dsig:Reference URI="#Id de la aserción">
             <dsig:Transforms>
                <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"
/>
                <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
             </dsig:Transforms>
             <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
             <dsig:DigestValue>
                base64(SHA1(canonicalización de la aserción temporal))
             </dsig:DigestValue>
          </dsig:Reference>
       </dsig:SignedInfo>
       <dsig:SignatureValue>
          base64(firmar con clave privada del STS (bloque SignedInfo))
       </dsig:SignatureValue>
       <dsig:KeyInfo>
          <dsig:X509Data>
             <dsig:X509Certificate>Certificado del STS</dsig:X509Certificate>
          </dsig:X509Data>
       </dsig:KeyInfo>
    </dsig:Signature>
 </saml:Assertion>
```

Mensaje de error

Otra de las funciones del IP es la emisión de mensajes de error, por ejemplo cuando la autenticación del usuario no es correcta. Creo una función para manejar una RST insatisfactoria.

FUNCIÓN: `errorMessage(msg,relatesto)`

PARÁMETROS DE ENTRADA:

`msg`: mensaje de error.

`relatesto`: identificador de la RST.

ACCIONES:

- Crea un mensaje XML como el de la figura 11.18 con un mensaje de error personalizado.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addres
   <s:Header>
      <a:Action s:mustUnderstand="1">
         http://www.w3.org/2005/08/addressing/soap/fault
      </a:Action>
      <a:RelatesTo>
         Id de la RST
      </a:RelatesTo>
   </s:Header>
   <s:Body>
      <s:Fault>
         <s:Code>
            <s:Value xmlns:a="http://www.w3.org/2003/05/soap-envelope">
               a:Sender
            </s:Value>
            <s:Subcode>
               <s:Value xmlns:a="http://schemas.xmlsoap.org/ws/2005/05/identity">
                  a:MissingAppliesTo'
               </s:Value>
            </s:Subcode>
         </s:Code>
         <s:Reason>
            <s:Text xml:lang="en">
               Mensaje de error
            </s:Text>
         </s:Reason>
      </s:Fault>
   </s:Body>
</s:Envelope>
```

Figura 11.18: Mensaje de error

11.2.3. Funcionalidades de la RP

La única funcionalidad que necesita la RP es la de consumir identidades.

Consumir identidad

Para desarrollar esta parte me basé (por consejo de mi tutor) en unas librerías existentes llamadas "php-infocard-rp". Están basadas en el marco de trabajo para PHP Zend y su código fuente se puede descargar de la siguiente dirección: http://code.google.com/p/php-infocard-rp/.

El directorio de instalación es `simplesaml/modules/InfoCard/lib/RP`.

Aunque esta herramienta está programada con un paradigma orientado a objetos, en términos abstractos la funcionalidad sigue siendo la misma. He aquí la forma básica de uso y las peculiaridades que he usado.

- Se construye el objeto.
- Se añade la clave privada de la RP.

  Método: `addIDPKey(clave)`
- Se añade el certificado (clave pública) del STS, proveedor de identidad.

  Método: `addSTSCertificate(certificado)`
- Se procesa el token XML.

  Método: process(token)

  Retorna: objeto con los claims extraidos del token.
- Se extraen los claims del objeto anterior.

El objeto con los claims tiene las siguientes funcionalidades.

- Saber si el token XML se procesó correctamente.

  Método: `isValid()`

  Retorna: `Verdadero o falso.`

Aclaraciones

Existe un problema con las nomenclaturas. Tenemos dos proveedores de identidad, uno en el metasistema (el único IP hasta el momento) y otro en la herramienta de SSO (al que me suelo referir como IdP).

Como estoy en el contexto de la herramienta de SSO, IdP será el proveedor de identidad de esta herramienta y el IP del metasitema se conocerá también como STS.

Así no puede haber confusiones en las funciones, por ejemplo el método `addIDPKey(...)` hace referencia a la clave privada del IdP (IP del SSO), en cambio el método `addSTSCertificate(...)` hace referencia al IP del metasistema.

El segundo problema de comprensión se podría dar en que ¡estoy hablando de la RP, la parte confiante! ¿Qué tiene que ver esto con los IPs y sus certificados? Sencillo, la peculiaridad de esta RP es que está integrada en la herramienta de SSO para poder iniciar sesión con una infocard. De

ahí que la herramienta de SSO sea proveedor de identidad (proporciona identidad del usuario a los sitios federados en el dominio de SSO) y RP (acepta infocards para iniciar SSO) a la vez.

Como resumen dejo esta correspondencia

- STS ≡ IP metasistema
- IDP ≡ IP SSO

Correcciones

Tuve unos cuantos problemillas con esta librería para la RP.

El primero fue con la validación de un esquema XML. Se saltaba una etiqueta y lo solucioné. Por ser un problema menor no registré constancia de ello.

El segundo problema, bastante más serio, fue que no validaba las firmas XML de los tokenes que recibía. Reorganicé los métodos para añadir el material criptográfico en detrimento del que ya venía `addCertificatePair(private_key_file, public_key_file, password = null)` y que semánticamente dice mucho menos. Después me sumergí en el fichero `Zend_InfoCard_Xml_Security.php` y apliqué lo que ya sabía de firmas XML (de cuando hice el generador de tarjetas) para validar la corrección del token. Básicamente es generar un resumen de lo recibido y comprararlo con la firma original, puesto que tenemos el certificado del STS almacenado localmente, el problema está resuelto.

Quisiera puntualizar la cantidad de problemas que da la canonicalización de un XML y destacar que aunque el algoritmo es el mismo para diferentes aplicaciones, no lo es así su implementación. Si se comete un error o, pero aún, una cuestión de estilo (¡es válido!) en la construcción de un XML, es muy probable que dos implementaciones distintas den resultados distintos al canonicalizarlo. Lo que se resume en falsos negativos de firmas digitales.

11.3. Módulo Infocard para simpleSAMLphp

En esta sección explicaré cómo creé el módulo de Infocard para la herramienta de SSO simpleSAMLphp.

La funcionalidad del módulo es la de podernos autenticar ante la herramienta de SSO como si fuera una RP del metasistema. Adicionalmente se le integraron las funcionalidades del IP del metasistema: generar tarjetas y emitir tókenes.

11.3.1. Características de un módulo

Recurro a la documentación de simpleSAMLphp para explicar cómo son los módulos de dicha aplicación. Está disponible en http://simplesamlphp.org/docs/1.5/simplesamlphp-modules.

Hay tres partes de simpleSAMLphp que se pueden empaquetar en forma de módulo.

1. **Fuentes de autenticación**
   Implementan diferentes métodos para autenticar a los usuarios. Por ejemplo: formularios, certificados o, en este caso, infocards.

2. **Filtros para procesar la autenticación**
   Realizan tareas varias después de que el usuario está autenticado y tiene su conjunto de atributos. Pueden añadir, borrar y modificar atributos así como hacer comprobaciones, preguntar al usuario...

3. **Temas**
   Sirven para empaquetar plantillas personalizadas.

Estructura del directorio

Cada módulo se guarda en una carpeta dentro del directorio `simplesaml/modules`. La carpeta del módulo contiene los siguientes directorios.

- `dictionaries`
  Contiene los diccionarios de internacionalización del módulo. Sirven para cambiar los mensajes de las plantillas y adaptarlos al idioma del usuario.

- `hooks`
  Contiene las funciones hook del módulo. En mi caso no lo he usado.

- `lib`
  Contiene las clases que pertenecen al módulo. Todas las clases se nombran con el siguiente patrón: `sspmod_<nombre del módulo>_<nombre de la clase>`. Cuando se invoca la clase, simpleSAMLphp buscará el fichero `<nombre de la clase>` en el directorio `lib`.
  Los guiones bajos "_" en el nombre de la clase se sustituyen por barras "/", de esta forma podemos agrupar los ficheros en directorios. Por ejemplo, la clase `sspmod_example_Auth_Source_Example` se localiza en el fichero `modules/example/lib/Auth/Source/Example.php`.

- `templates`
  Contiene las plantillas del módulo. Se considera una plantilla a la vista que se le presentará al usuario a través del navegador web. Ejemplo: una página web con algo de lógica en PHP.

- `themes`
  Contiene los temas del módulo. En mi caso no lo uso, tomo el tema por defecto.

- `www`
  Contiene los ficheros que son acesibles a través de la web siguiendo una dirección del estilo `https://.../simplesamlphp/module.php/<nombre del módulo>/<nombre del fichero>`.

Además existen unos casos particulares que explico a continuación.

- `Activación/desactivación del módulo`
  Según la existencia de los siguientes ficheros, el módulo estará.

  Activado, fichero `default-enable`.

  Desactivado, fichero `default-disable`.

- `config-templates`
  Directorio en el que se puede incluir una configuración por defecto para usarla como plantilla en el momento de la instalación.

- `docs`
  Directorio en el que se puede incluir la documentación del módulo (uso, como instalarlo, configuración, etc).

11.3.2. Fichero de configuración

Antes de empezar con la lógica del módulo, explicaré el fichero de configuración del mismo.
Este fichero es de vital importancia ya que permite permite personalizar todas las opciones disponibles: certificados, imágenes, rutas, autenticación ...

La plantilla original se localiza en
`simplesaml/modules/InfoCard/config-templates/config-login-infocard.php`.
Es necesario copiarlo en el directorio `simplesaml/config` para poder utilizarlo. Recomiendo mirar el fichero ya que contiene comentarios de cada campo así como anotaciones que serán de gran ayuda para el desarrollados.

Aprovecharé las secciones en que divido los campos para introducir nuevas funcionalidades del módulo que explicaré más adelante.

El fichero de configuración es código PHP en el que se guarda una variable "$config" cuyo valor es una matriz asociativa con los siguientes campos.

Opciones para las plantillas. Contienen la información personalizable par las vistas de usuario o plantillas. No son eran indispensables pero permiten una configuración rápida de los aspectos más básicos.

- `IClogo`
  Imagen del formulario para autenticarse con Information Cards.

- `help_desk_email_URL`
  Correo electrónico del soporte técnico.

- `contact_info_URL`
  URL de la dirección de contacto.

Certificados. Contienen las rutas absolutas a los certificados x509 o a sus claves privadas.

- `idp_key`
  Clave privada del IdP del SSO.
- `sts_crt`
  Certificado del STS.
- `sts_key`
  Clave privada del STS.
- `certificates`
  Vector que contiene la cadena de confianza de certificados, el primero es el del STS y el último el de la autoridad de certificación.

El campo `InfoCard` es una matriz asociativa con los datos referentes a la **configuración de las tarjetas**.

- `schema`
  Ruta URL común para acceder a la especificación XML de la infocard. Por ejemplo para construir la URI de los claims.
- `issuerPolicy`
  URL de donde se puede obtener el documento XML WS-SecurityPolicy del STS.
- `privacyURL`
  URL en donde se puede obtener la política de privacidad (para humanos).
- `tokenType`
  Tipo de token soportado.
- `requiredClaims`
  Matriz asociativa con los claims obligatorios.
- `optionalClaims`
  Matriz asociativa con los claims opcionales.

Estructura de los claims. La estructura de los dos campos anteriores `requiredClaims` y `optionalClaims` es la de una matriz asociativa cuyas claves son los nombres de los claims del esquema usado en `schema` y cuyo valor es una matriz asociativa con los siguientes campos.

- `displayTag`
  Etiqueta del claim. Por ejemplo el claim se llama "emailaddress" y como etiqueta le ponemos "e-mail".
- `description`
  Descripción del claim. Qué es.

Páginas Web del IP. Contienen direcciones web referentes a partes del metasistema.

- `CardGenerator`
  URL del generador de tarjetas.
- `tokenserviceurl`
  URL del servicio STS, el generador de tókenes.

- `mexurl`
  URL del servicio Metadata Exchange.

Credenciales.

- `UserCredential`
  Permite establecer qué tipo de credenciales usarán los usuarios para autenticarse en el IP del metasistema. Este parámetro afecta al STS y al generador de tarjetas (al definir el método cuando se genera la tarjeta).
  Actualmente solo hay dos métodos implementados:

  `UsernamePasswordCredential`
  Autenticación clásica por usuario/contraseña.

  `SelfIssuedCredential`
  Autenticación mediante tarjeta autoemitida.

Debug. Opciones útiles para la depuración y registro de mensajes.

- `debugDir`
  Directorio en el que se escribirirán los mensajes de depuracións. Cada par petición(RST) - respuesta(RSTR) al STS se grabará en un fichero con el patrón `urn:uuid:XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX.log` en el que X es un número hexadecimal. Si no se especifica directorio, no se registrarán las peticiones.

Interfaz IP (STS). Opciones de la interfaz del IP para comunicarse con el servidor radius y generar tarjetas.

- `configuredIP`
  Dirección IP del servidor RADIUS que hace las peticiones de URL de un solo uso para generar infocards.

- `symmetric_key`
  Clave compartida con el servidor RADIUS para intercambiar datos.

- `internal_key`
  Clave interna para cifrar la URL de un solo uso que se le manda al RADIUS, se recibe después por parte del cliente y se descifra para atender la petición de generar una infocard.

- `IC_lifetime_delivery`
  Tiempo de vida máximo de la petición de la infocard. Este tiempo va desde que el servidor RADIUS hace la petición de tarjeta al IP hasta que el usuario la consigue (hace otra petición). Si se excede el tiempo máximo, se supone que ha habido un error y no se emite infocard.

El campo `DB_params` es una matriz asociativa con los datos relacionados con la **conexión a la base de datos**.

- `DB_host`
  Dirección del servidor de base de datos.

- `DB_port`
  Puerto de conexión.

- `DB_dbname`
  Nombre de la base de datos.

- `DB_user`
  Usuario de conexión
- `DB_password`
  Contraseña de conexión.

NOTA: *El servidor PHP debe tener acceso a los ficheros que se configuran como rutas: certificados, directorio de depuración, etc.*

11.3.3. Adaptación de la librería

Es momento de adaptar las funciones que vimos en la sección de funcionalidades básicas (referencia) a la estructura del módulo para poder usarlas como una librería.

El proceso consistirá en crear los ficheros necesarios en el directorio `simplesaml/modules/InfoCard/lib` y definir una estructura de clase de acuerdo a la especificación vista.

Clase sspmod_InfoCard_STS

FICHERO: `simplesaml/modules/InfoCard/lib/STS.php`

DESCRIPCIÓN:

Contiene las funcionalidades del IP del metasistema.
Tres métodos estáticos públicos para generar tarjetas, emitir tókenes (soporte del STS) y mensajes de error.

- `createCard($ICdata,$ICconfig)`
- `createToken($claimValues,$config,$relatesto)`
- `errorMessage($msg,$relatesto)`

Dos métodos privados usados por el método de emitir token.

- `RequestSecurityTokenResponse($claimValues,$config,$assertionid,$created,$expires)`
- `saml_assertion($claimValues,$config,$assertionid,$created,$expires)`

ESTRUCTURA:

```
1   class sspmod_InfoCard_STS {
2
3       static public function createCard($ICdata,$ICconfig) {
4           ...
5       }
6
7       static public function errorMessage($msg,$relatesto){
8           ...
9       }
10
```

```
11        static public function createToken
12           ($claimValues,$config,$relatesto){
13           ...
14        }
15
16        static private function RequestSecurityTokenResponse
17           ($claimValues,$config,$assertionid,$created,$expires){
18           ...
19        }
20
21        static private function saml_assertion
22           ($claimValues,$config,$assertionid,$created,$expires){
23           ...
24        }
25
26     }
```

Clase sspmod_InfoCard_RP_InfoCard

FICHERO: `simplesaml/modules/InfoCard/lib/RP/InfoCard.php`

DESCRIPCIÓN:

Clase instanciable para procesar los tókenes en la RP.

ESTRUCTURA:

```
1     class sspmod_InfoCard_RP_InfoCard {
2        __construct() {
3        }
4
5        addSTSCertificate($sts_crt){
6           ...
7        }
8
9        addIDPKey($private_key_file, $password = NULL){
10          ...
11       }
12
13       process($xmlToken){
14          ...
15       }
16
17    }
```

Clase Zend_InfoCard_Claims

FICHERO: `simplesaml/modules/InfoCard/lib/RP/Zend_InfoCard_Claims.php`

DESCRIPCIÓN:
Clase de apoyo para la clase `sspmod_InfoCard_RP_InfoCard` cuya función `process($xmlToken)` devuelve un objeto instanciado de esta clase.

ESTRUCTURA:

```
1   class Zend_InfoCard_Claims{
2
3       public function isValid(){
4           ...
5       }
6
7       public function getClaim($claimURI){
8           ...
9       }
10
11      public function claimExists($claimURI){
12          ...
13      }
14
15  }
```

11.3.4. Plantillas

Crearé dos plantillas web o vistas de usuario.

Vista principal

FICHERO: `simplesaml/modules/InfoCard/templates/default/temp-login.php`

DESCRIPCIÓN:
Es la vista principal del módulo. Servirá para que el usuario haga inicio de sesión único a través de un formulario web con su infocard. La figura 11.19 muestra el aspecto final de la página web.

PARÁMETROS DE ENTRADA:
La plantilla posee una matriz asociativa "data" como atributo del objeto con los siguiente campos.

- `header`
  Título de la página.
- `stateparams`
  Valor del campo "AuthState" del formulario. Información actual del usuario sobre la autenticación en simpleSAMLphp.
- `IClogo`
  Ruta de la imagen del icono de Information Card para el formulario.

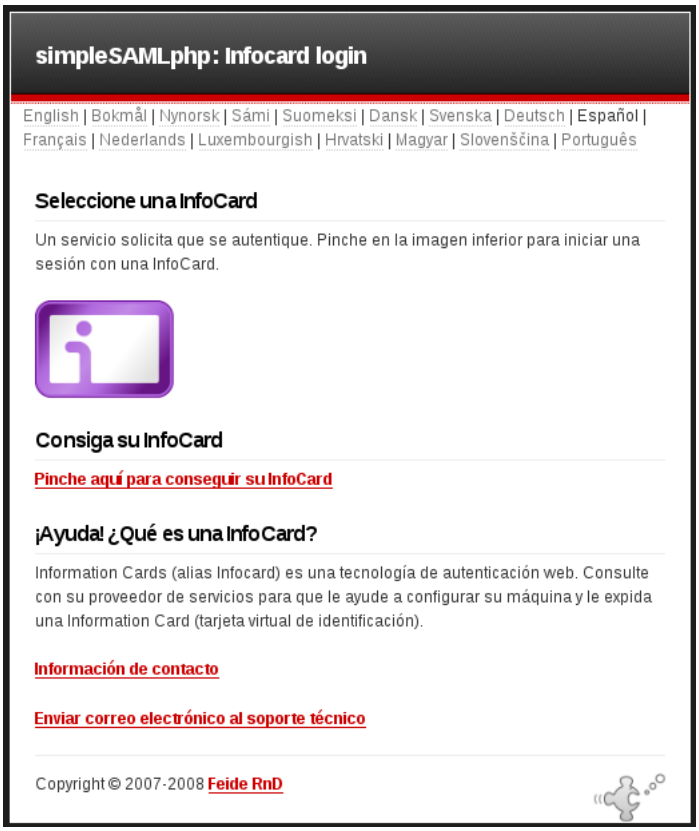

Figura 11.19: Vista principal del módulo.

- `InfoCard`
  Configuración relativa a las infocards (claims) además se mete como campo de esta matriz la clave "issuer" para especificar el IP que firmará el token.
- `CardGenerator`
  Web en donde se puede conseguir una infocard.
- `help_desk_email_URL`
  Correo electrónico del soporte técnico.
- `contact_info_URL`
  Web de información.
- `error`
  Si existe un error, contiene el mensaje para el usuario.

Además es posible acceder a los mensajes del diccionario del módulo.

LÓGICA:

1. Cargar la cabecera de la página, estilo, barra de idiomas, etc.
2. Si el campo "error" está presente, se muestra el icono de error (una bomba) junto con el mensaje de error. Ver figura 11.20.
3. Se muestra el formulario de autenticación con infocard, caracterizado por la imagen morada de la infocard. Es un botón.
4. Se muestra el enlace para conseguir una infocard.
5. Se muestran los enlaces de la sección de ayuda (correo y web de ayuda).

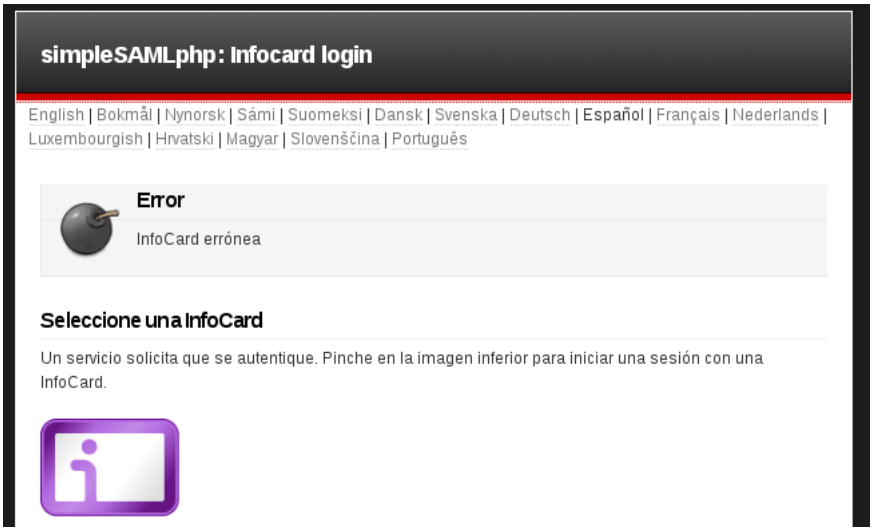

Figura 11.20: Vista con mensaje de error.

FORMULARIO WEB:

El formulario web que se presenta se compone de código HTML en el que especifica que se espera recibir un objeto de tipo infocard. Se añaden parámetros para especificar el tipo de token, el emisor del mismo y los claims requeridos; así cuando el selector de identidad interprete la información, sólo debería presentarnos tarjetas que cumplan con estos requisitos. También es posible codificarlo según un esquema XHTML, pero las pruebas no fueron satisfactorias así que opté por mostrarlo como código comentado por si en un futuro fuese necesario.
A continuación muestro un formulario web de ejemplo obtenido automáticamente con la configuración por defecto.

```
1   <form name="ctl00" id="ctl00" method="post"
2    action="?AuthState=_f9074ad8888bfc9958bbd96b5fc523dd2ce421498c">
3      <OBJECT type="application/x-informationCard" name="xmlToken">
4         <PARAM Name="issuer"
5          Value="https://sts.ic/module.php/InfoCard/tokenservice.php">
6         <PARAM Name="tokenType" Value="urn:oasis:names:tc:SAML:1.0:assertion">
7         <PARAM Name="requiredClaims"
```

```
8           Value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier
9            http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
10           http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
11           http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress ">
12         <PARAM Name="optionalClaims"
13          Value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country
14           http://schemas.xmlsoap.org/ws/2005/05/identity/claims/webpage ">
15      </OBJECT>
16      <input type='image' src="resources/infocard_114x80.png" style='cursor:pointer' />
17    </form>
```

Obtener tarjeta

FICHERO: `simplesaml/modules/InfoCard/templates/default/temp-getcardform.php`

DESCRIPCIÓN:

Es una vista opcional para que el usuario pueda obtener una infocard.

PARÁMETROS DE ENTRADA:

Análogamente al caso anterior, tiene acceso a los mensajes del diccionario y a la matriz asociativa data de la que usan los siguientes campos.

- `header`
  Título de la página.
- `stateparams`
  Valor del campo "AuthState" del formulario. Información actual del usuario sobre la autenticación en simpleSAMLphp.
- `InfoCard`
  Configuración relativa a las infocards (claims) además se mete como campo de esta matriz la clave "issuer" para especificar el IP que firmará el token.
- `CardGenerator`
  Web en donde se puede conseguir una infocard.
- `help_desk_email_URL`
  Correo electrónico del soporte técnico.
- `contact_info_URL`
  Web de información.
- `error`
  Si existe un error, contiene el mensaje para el usuario.
- `form`
  Acción a realizar.
- `username`
  Nombre de usuario.
- `password`
  Contraseña.

LÓGICA:

1. Cargar la cabecera de la página, estilo, barra de idiomas, etc.
2. Si el campo "error" está presente, se muestra el icono de error (una bomba) junto con el mensaje de error.
3. Se comprueba que exista la web del generador de tarjetas.
4. Se realiza una acción según el valor del campo `form`. Puede tomar los siguientes valores:
   - `validate`
     Se presenta un formulario para autenticación con usuario/contraseña como el de la figura 11.21.
   - `selfIssued`
     Se presenta un formulario que acepta una Information Card autoemitida como el de la figura 11.22.
   - `otros valores (end)`
     Se muestra un mensaje de proceso finalizado correctamente.
5. Se muestran los enlaces de la sección de ayuda (correo y web de ayuda).

Figura 11.21: Formulario de autenticación por usuario/contraseña para obtener una infocard.

Figura 11.22: Formulario para enviar la tarjeta autoemitida que se usará como credencial de la tarjeta gestionada.

11.3.5. Controladores (páginas y servicios accesibles)

Es momento de definir las páginas web y servicios que estarán a disposición de los usuarios a través de la herramienta de SSO simpleSAMLphp.

Como se vio antes, todo fichero o directorio que se encuentre en la carpeta `simplesaml/modules/InfoCard/www` será accesible. Aprovechando esta característica, procedo a definir lo que en un patrón de diseño M.V.C. se corresponde con el controlador, es decir, el código con más acoplamiento que se encarga de procesar la entrada, tratarla con las liberías (el modelo) y presentar un resultado (en algunos casos una vista para el usuario y en otros ficheros).

Controladores de vistas

Se encargan de recibir una entrada, procesarla y cargar una plantilla para presentársela al usuario.

Inicio de sesión

FICHERO: `simplesaml/modules/InfoCard/www/login-infocard.php`

URL: `http://.../module.php/InfoCard/login-infocard.php`

DESCRIPCIÓN:

Se encarga de presentar la vista principal del módulo y procesar la respuesta del usuario para que éste pueda hacer inicio de sesión único.

LÓGICA:

1. Carga la configuración del módulo.
2. Comprueba la sesión del usuario.

   Si recibió el token, lo procesa con el método `sspmod_InfoCard_Auth_Source_ICAuth::handleLogin(...)`

   Si no hay token o la función anterior devolvió un mensaje de error, carga la plantilla del fichero `simplesaml/modules/InfoCard/templates/default/temp-getcardform.php` con el formulario de infocard y la presenta al usuario.

Obtener tarjeta

FICHERO: `simplesaml/modules/InfoCard/www/getcardform.php`

URL: `https://.../module.php/InfoCard/getcardform.php`

DESCRIPCIÓN:

Presenta la vista de obtener tarjeta para que el usuario se autentique por el método de usuario/contraseña y luego dependiendo de la autenticación configurada, presenta otro formulario o emite la infocard.

Lógica:

1. Carga la configuración del módulo.
2. Comprueba la sesión del usuario.
3. Establece el estado de la plantilla en "validate".
4. Comprueba si recibió el formulario con el nombre de usuario y contraseña y verifica que los credenciales son correctos con el método `sspmod_InfoCard_UserFunctions::validateUser(...)`
   a) Si los credenciales son correctos, comprueba el método de autenticación que usará la infocard.
      - Usuario/contraseña
        1) Carga la información de usuario con la función `sspmod_InfoCard_UserFunctions::fillICdata()`
        2) Usa como credenciales los mismos que se pusieron en el formulario anterior.
        3) Genera la infocard con la función `sspmod_InfoCard_STS::createCard(...)`.
        4) Le presenta la infocard al usuario como un fichero descargable.
      - Tarjeta autoemitida

        Comprueba si ha recibido un token de tarjeta autoemitida. Necesita el PPID de dicha tarjeta para usarlo como credencial.

1) Si ha recibido el token.
   a' Si el token es válido.
   Obtiene la información del usuario con la función `sspmod_InfoCard_UserFunctions::fillICdata(...)`.
   Genera la infocard con la función `sspmod_InfoCard_STS::createCard(...)`.
   Le presenta la infocard al usuario como un fichero descargable.
   Se establece el estado de la plantilla a "end".
   b' Si el token no es válido.
   Establece la variable de error en la plantilla.
   Cambia el estado de la plantilla a "SelfIssued".
   Se establece el estado de la plantilla a "end".
2) Si no recibió el token.
   a' No se ha producido error, puede ser la primera vez que se carga la página.
   b' Cambia el estado de la plantilla a "SelfIssued".

b) Si los credenciales no son correctos o falta algún campo, se establece el respectivo mensaje de error.

5. Se procede con la plantilla.
6. Se carga la plantilla del fichero `simplesaml/modules/InfoCard/templates/default/temp-login.php`.
7. Se configura la plantilla.
8. Se le muestra la plantilla al usuario.

Controladores de servicios

Metadata Exchange

FICHERO: `simplesaml/modules/InfoCard/www/mex.php`

URL: `https://.../module.php/InfoCard/mex.php`

DESCRIPCIÓN:
Recurso que describe el servicio de STS que ofrece el IP mediante un XML con el esquema WSDL (Web Services Description Languaje).

LÓGICA:

1. Carga la configuración del módulo.
2. La parte más variable es el bloque que define la autenticación ante el STS con la infocard según se haya configurado el módulo.
3. Presenta un documento como el que se muestra a continuación con la descripción del servicio STS del IP. Se ha configurado para autenticación con tarjeta autoemitida, se observa en el bloque `sp:EndorsingSupportingTokens`.

```
1  <?xml version="1.0"?>
2  <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
3   xmlns:wsa="http://www.w3.org/2005/08/addressing">
4
5     <S:Header>
```

```
6          <wsa:Action S:mustUnderstand="1">
7             http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
8          </wsa:Action>
9          <wsa:RelatesTo>
10            Identificador de petición
11         </wsa:RelatesTo>
12      </S:Header>
13
14      <S:Body>
15         <Metadata xmlns="http://schemas.xmlsoap.org/ws/2004/09/mex">
16
17            <MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/"
18             Identifier="http://schemas.xmlsoap.org/ws/2005/02/trust">
19               <wsdl:definitions name="STS_wsdl" targetNamespace="URL del STS"
20                xmlns:tns="URL del STS" xmlns:xs="http://www.w3.org/2001/XMLSchema"
21                xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
22                xmlns:wsa="http://www.w3.org/2005/08/addressing"
23                xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
24                xmlns:wsid="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity"
25                xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
26                xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
27                xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
28                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
29                xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
30                xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
31                xmlns:ic=ghttp://schemas.xmlsoap.org/ws/2005/05/identity"
32                xmlns:q1="URL del STS">
33
34                  <wsdl:types>
35                     <xs:schema targetNamespace="http://schemas.xmlsoap.org/ws/2005/02/trust/Imports">
36
37                        <xs:import schemaLocation="" namespace="URL del STS"/>
38                     </xs:schema>
39                  </wsdl:types>
40
41                  <wsdl:message name="RequestSecurityTokenMsg">
42                     <wsdl:part name="request" type="q1:MessageBody" />
43                  </wsdl:message>
44                  <wsdl:message name="RequestSecurityTokenResponseMsg">
45                     <wsdl:part name="response" type="q1:MessageBody" />
46                  </wsdl:message>
47
48                  <wsdl:portType name="SecurityTokenService">
49                     <wsdl:operation name="Issue">
50                        <wsdl:input wsaw:Action="http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue"
51                         message="tns:RequestSecurityTokenMsg">
52                        </wsdl:input>
53                        <wsdl:output wsaw:Action="http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue"
                           message="tns:RequestSecurityTokenResponseMsg">
```

```
54                      </wsdl:output>
55                   </wsdl:operation>
56                </wsdl:portType>
57
58                <wsp:Policy wsu:Id="STS_endpoint_policy">
59                   <wsp:ExactlyOne>
60                      <wsp:All>
61                         <ic:RequireFederatedIdentityProvisioning />
62                         <sp:TransportBinding>
63                            <wsp:Policy>
64                               <sp:TransportToken>
65                                  <wsp:Policy>
66                                     <sp:HttpsToken RequireClientCertificate="false" />
67                                  </wsp:Policy>
68                               </sp:TransportToken>
69                               <sp:AlgorithmSuite>
70                                  <wsp:Policy>
71                                     <sp:Basic256/>
72                                  </wsp:Policy>
73                               </sp:AlgorithmSuite>
74                               <sp:Layout>
75                                  <wsp:Policy>
76                                     <sp:Strict/>
77                                  </wsp:Policy>
78                               </sp:Layout>
79                               <sp:IncludeTimestamp/>
80                            </wsp:Policy>
81                         </sp:TransportBinding>
82
83                         <sp:EndorsingSupportingTokens
84                          xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
85                          xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
86                            <wsp:Policy>
87                               <sp:IssuedToken
88                                sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/
89                                 securitypolicy/IncludeToken/AlwaysToRecipient">
90                                  <sp:Issuer>
91                                     <wsa:Address>
                                          http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
92
93                                     </wsa:Address>
94                                  </sp:Issuer>
95                                  <sp:RequestSecurityTokenTemplate>
96                                     <wst:TokenType>
97                                        urn:oasis:names:tc:SAML:1.0:assertion
98                                     </wst:TokenType>
99                                     <wst:KeyType>
                                          http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
100
101                                    </wst:KeyType>
102                                    <wst:Claims
```

```
                                    xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity">
                                      <ic:ClaimType
                                       Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/
                                         claims/privatepersonalidentifier"/>
                                    </wst:Claims>
                                 </sp:RequestSecurityTokenTemplate>
                                 <wsp:Policy>
                                    <sp:RequireInternalReference/>
                                 </wsp:Policy>
                              </sp:IssuedToken>
                           </wsp:Policy>
                        </sp:EndorsingSupportingTokens>

                        <sp:Wss11>
                           <wsp:Policy>
                              <sp:MustSupportRefThumbprint/>
                              <sp:MustSupportRefEncryptedKey/>
                           </wsp:Policy>
                        </sp:Wss11>
                        <sp:Trust10>
                           <wsp:Policy>
                              <sp:RequireClientEntropy/>
                              <sp:RequireServerEntropy/>
                           </wsp:Policy>
                        </sp:Trust10>
                        <wsaw:UsingAddressing wsdl:required="true" />
                     </wsp:All>
                  </wsp:ExactlyOne>
               </wsp:Policy>

               <wsdl:binding name="Transport_binding" type="tns:SecurityTokenService">
                  <wsp:PolicyReference URI="#STS_endpoint_policy"/>
                  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
                  <wsdl:operation name="Issue">
                     <soap12:operation
                      soapAction="http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue"
                      style="document"/>
                     <wsdl:input>
                        <soap12:body use="literal"/>
                     </wsdl:input>
                     <wsdl:output>
                        <soap12:body use="literal"/>
                     </wsdl:output>
                  </wsdl:operation>
               </wsdl:binding>

               <wsdl:service name="STS_0">
                  <wsdl:port name="STS_0_port" binding="tns:Transport_binding">
                     <soap12:address location="URL del STS" />
```

```
152                      <wsa:EndpointReference>
153                         <wsa:Address>URL del STS</wsa:Address>
154                         <wsid:Identity>
155                            <ds:KeyInfo>
156                               <ds:X509Data>
157                                  <ds:X509Certificate>
158                                     Certificado del STS en base64
159                                  </ds:X509Certificate>
160                               </ds:X509Data>
161                            </ds:KeyInfo>
162                         </wsid:Identity>
163                      </wsa:EndpointReference>
164                   </wsdl:port>
165                </wsdl:service>
166
167             </wsdl:definitions>
168          </MetadataSection>
169
170
171          <MetadataSection Dialect="http://www.w3.org/2001/XMLSchema"
172           Identifier="URL del STS">
173             <xs:schema xmlns:tns="URL del STS"
174              xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
175              targetNamespace="URL del STS">
176                <xs:complexType name="MessageBody">
177                   <xs:sequence>
178                      <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"/>
179                   </xs:sequence>
180                </xs:complexType>
181             </xs:schema>
182          </MetadataSection>
183
184       </Metadata>
185    </S:Body>
186 </S:Envelope>
187
```

Servicio de tókenes

FICHERO: `simplesaml/modules/InfoCard/www/tokenservice.php`

URL: `https://.../module.php/InfoCard/tokenservice.php`

DESCRIPCIÓN:

Acceso al servicio de emisión de tókenes. Se recibe una petición RST, se evalúan los credenciales y si todo es correcto se emite una respuesta RSTR on el token.

LÓGICA:

1. Se carga la configuración del módulo.
2. Se pone el estado a no autenticado.
3. Se procesa la petición RST según sea el método de autenticación:
   - Usuario/contraseña.
     Se extraen los credenciales y se validan con la función `sspmod_InfoCard_UserFunctions::validateUser(...)`.
     Si son válidos, se actualiza el estado a autenticado.
   - Tarjeta autoemitida.
     Se obtiene el token autoemitido de la petición RST.
     Se decodifica en base 64 y se descifra con la clave privada del STS.
     Se analiza y se obtiene el claim PPID.
     Se valida el PPID con la función `sspmod_InfoCard_UserFunctions::validateUser(...)`.
     Si el PPID es válido, se actualiza el estado a autenticado.
4. Si el estado es autenticado.
   Se extraen los claims requeridos en la petición RST con la función `sspmod_InfoCard_Utils::extractClaims(...)`.
   Se obtiene la información del usuario (para rellenar los claims) del almacén con la función `sspmod_InfoCard_UserFunctions::fillClaims(...)`.
5. Se genera la respuesta RSTR (el token con la identidad digital del usuario) con la función `sspmod_InfoCard_STS::createToken(...)`.
6. Si el estado no es autenticado.
   - Se genera la respuesta con un mensaje de error con la función `sspmod_InfoCard_STS::errorMessage(...)`.
7. Se devuelve la respuesta.
8. Si está configurado el directorio de depuración, se escribe en éste el fichero con el par RST/RSTR.

Servicio de generación de tarjetas

FICHERO: `simplesaml/modules/InfoCard/www/STS_card_issuer.php`

URL: `https://.../module.php/InfoCard/STS_card_issuer.php`

DESCRIPCIÓN:
Es un servicio del IP con dos funcionalidades.

1. El servidor RADIUS federado puede hacer peticiones de creación de infocards al servidor IP y que éste le devuelve una ruta de un solo uso para que el usuario se descargue la tarjeta.
2. El usuario, a través de la ruta de un solo uso que consiguió cuando se autenticó ante el servidor RADIUS, consigue su infocard.

PROTOCOLO:
El protocolo de comunicación RADIUS-IP es bastante simple y se basa en el modelo petición-respuesta. El servidor RADIUS lanza una petición HTTP a la web del servicio mediante el método GET (todos los parámetros van incluidos en la URL) y el servidor responde. Veamos cómo se compone cada mensaje y qué contiene.

- **Petición RADIUS**
  1. Invoca la URL:
     `https://.../module.php/InfoCard/STS_card_issuer.php?data=X&iv=Y&ident=RADIUS`
  2. El parámetro ident define que es el servidor RADIUS quien invoca el servicio.
  3. El parámetro data son el identificador de usuario y su cardid codificados en un mensaje de la forma `longitud, Id de usuario, longitud, cardID`. Se cifra simétricamente con la clave compartida (parámetro `symmetric_key` del fichero de configuración) y se codifica en base64 seguro para URL.
  4. El parámetro iv es el vector de inicialización para el cifrado simétrico codificado en base64 seguro para URL.
- **Respuesta RADIUS**
  1. La respuesta que recibe el servidor RADIUS es un mensaje codificado en base64, cifrado simétricamente con la clave compartida. Este mensaje contiene la URL para conseguir la infocard del servidor.
- **Petición cliente**
  1. Cuando el cliente tiene su URL de un solo uso, la invoca para conseguir su infocard.
  2. La URL tiene el siguiente patrón:
     `https://.../module.php/InfoCard/STS_card_issuer.php?data=X&iv=Y`.
  3. El parámetro data contiene el identificador interno de petición o uuid cifrado con el parámetro de la configuración `internal_key` y codificado en base64 seguro para URL. El servidor de IP utiliza el identificador de petición para recuperar del almacén la petición hecha por el servidor RADIUS y generar la infocard. Este campo lo cifra el IP para sí mismo y de esa manera evitar que se hagan peticiones arbitrarias.
  4. El parámetro iv es el vector de inicialización para el cifrado simétrico de data, está codificado en base64 seguro para URL.
- **Respuesta cliente**
  1. Una vez hecha la petición por parte del cliente, el servidor le responde con una infocard para que se la descargue.

LÓGICA:

El primer paso después de cargar la configuración global, es discernir qué tipo de petición recibe el servicio.

- **Petición del servidor RADIUS**
  1. Se sabe que la petición es del RADIUS por el parámetro `ident` que va en la URL que invoca al servicio y porque la petición procede la IP configurada en el campo `configuredIP` del fichero de configuración.
  2. Se cargan las claves de cifrados simétrico `symmetric_key` e `internal_key`.
  3. Se decodifica en base64 seguro para URL los campos data e iv de la petición.
  4. Se descifra el campo data con la clave compartida `symmetric_key` y el iv.
  5. Se parsean los atributos del campo data con la función `parse_attributes` para obtener el identificador de usuario y su cardID.
  6. Se invoca a la función `enable_download(...)` que guarda la petición el IP y devuelve el identificador interno de la petición o uuid.
  7. Se cifra el identificador de la petición con la clave `internal_key`, se codifica en base64 seguro para URL y se construye la URL de un solo uso poniendo la transformación del identificador en el campo data y el iv en base64 seguro para URL en el campo iv.
  8. Se cifra la URL de un solo uso con la clave `symmetric_key`, se codifica en base64 y se le envía de vuelta al servidor RADIUS.
- **Petición del cliente**
  1. Se sabe que es una petición de infocard por parte del cliente por fallo al reconocerlo como el servidor RADIUS.
  2. Se toman los campos de la URL `data` e `iv` y se decodifican en base64 seguro para URL.
  3. Se carga la clave interna `internal_key` así como los campos necesarios del fichero de configuración. Campos que se usarán para generar la infocard.
  4. Se descifra el campo data con la clave `internal_key`.
  5. Se comprueba que la petición/tarjeta están disponibles con la función `is_card_enabled(...)`.
  6. Si existe una petición de tarjeta pendiente
     a) Se registra en el almacén al usuario como conectado con la función `DB_update_connected_user(...)`.
     b) Se calcula el PPID de la tarjeta que se va a generar a partir del cardID y los certificados con la función `calculate_PPID(...)`.
     c) Se codifica en base64 el PPID.
     d) Se obtienen los datos del usuario del almacén con el método `sspmod_InfoCard_UserFunctions::fillICdata(...)`.
     e) Se crea la tarjeta con el método `spmod_InfoCard_STS::createCard(...)`
     f) Se borra la petición de infocard que hizo el RADIUS almacenada en el sistema con la función `disable_download(...)`.
     g) Se le muestra la infocard al usuario para que la descargue.
  7. Si no existe petición pendiente, se devuelve un error.

FUNCIONES PROPIAS:

Comento a continuación las funciones específicas de este servicio

1. Codificación base64 segura para URL. Permiten codificar/decodificar información para mandarla de forma segura en las peticiónes URL.
   - `urlsafe_b64encode($string)`
     Codifica. Recibe una cadena de bytes y devuelve una cadena de texto segura para URL.
   - `urlsafe_b64decode($string)`
     Decodifica. Recibe una cadena de texto segura para URL y devuelve una cadena de bytes.
2. Cálculo del PPID.
   - `calculate_PPID($cardid, $rp_cert)`
     Recibe el CardID de la tarjeta autoemitida y la cadena de certificados para generar el PPID asociado al IP que se utilizará en el credencial de la tarjeta gestionada.
   - `calculate_RP_PPID_Seed_2_2007 ($certs)`
     Función de apoyo a la anterior que calcula la semilla de la RP a través de la cadena de certificados que recibe.
3. `curPageURL()`
   Devuelve la URL de la página que se ha invocado. Se usa para generar la URL de un solo uso ya que la URL del servicio es la misma para el servidor RADIUS (cuando hace la petición) que para el cliente (cuando descarga su infocard).
4. `parse_attributes($parsing_string, $num_attrs)`
   Recibe una cadena de la forma longitud.atributo n veces, el número de atributos que espero recibir y devuelve un vector con los atributos parseados. Se usa en la comunicación RADIUS-IP para mandar el identificador de usuario y el cardID.
5. Almacén de peticiones.
   - `enable_download($username, $cardid)`
     Crea un archivo en el directorio temporal `/tmp` con nombre el identificador de petición o uuid único y de contenido los datos referidos a la petición, como el identificador de usuario, el cardid y la fecha en que el servidor RADIUS hizo la petición.
   - `disable_download($uuid)`
     Borra el fichero dado con la petición de infocard.
   - `is_card_enabled($uuid, $delivery_time)`
     Recibe el identificador de petición y la fecha en que el cliente pide su infocard. Comprueba que existe el fichero con esa petición (el servidor RADIUS hizo la petición de URL), parsea el fichero, comprueba que el tiempo entre la petición de URL por parte del RADIUS y la petición de infocard por parte del cliente no supera un cierto intervalo (sesión caduca) y si tiene éxito devuelve los datos almacenados en el fichero, en caso contrario, falso.
6. `DB_update_connected_user($username, $DB_params)`
   Comprueba si el usuario aparece en la tabla de conectados en la base de datos y si no lo está, lo conecta y devuelve verdadero. En caso contrario, devuelve falso.

NOTA: *base64 seguro para URL es una forma de representación de la información basada en base64 que se usa para codificar parámetros para mandarlos en las peticiones HTTP junto con la URL, de forma que no contiene caracteres especiales de las cabeceras como barras, interrogaciones, etc.*

11.3.6. Extras

En esta sección trato temas accesorios que serán de ayuda si se procede a "cacharrear" con el invento.

Lista de ficheros

El módulo oficial se localiza en la carpeta `./simplesaml/modules/InfoCard/` y tiene la siguiente lista de archivos.

- `default-enable o default-disable`
  Fichero que indica si está activo o no el módulo.
- `./config-templates/config-login-infocard.php`
  Configuración de ejemplo.
- `./dictionaries/dict-InfoCard.php`
  Diccionario del módulo, mensajes en varios idiomas.
- `./docs/usage.txt`
  Fichero de ayuda del módulo.
- `./lib/`
  Directorio de las librerías.
  - `./Auth/Source/ICAuth.php`
    Fuente de autenticación, fichero que une la autenticación con infocard en el programa simpleSAMLphp.
  - `./RP/`
    Directorio con la librería de infocard para la RP. Contiene los siguientes ficheros.
    - `InfoCard.php`
      Fichero para el manejo del objeto principal Infocard.
    - `LICENSE.txt`
      Licencia de la librería.
    - `Zend_InfoCard_Claims.php`
      Fichero para el manejo del objeto con los claims.
    - `Zend_InfoCard_Xml_Assertion_Saml.php`
      Manejo de aserciones SAML.
    - `Zend_InfoCard_Xml_Security.php`
      Manejo de la seguridad XML.
    - `Zend_InfoCard_Xml_Security_Transform_EnvelopedSignature.php`
      Manejo de de la firma XML.
    - `Zend_InfoCard_Xml_Security_Transform.php`
      Transformaciones XML.
    - `Zend_InfoCard_Xml_Security_Transform_XmlExcC14N.php`
      Canonicalización XML.
  - `STS.php`
    Funcionalidades del STS.
  - `UserFunctions.php`
    Funciones definidas por el usuario.

  - `Utils.php`
    Funciones útiles varias.

- `./templates/default/`
  Directorio con las vistas (plantillas) básicas.

  - `temp-login-InfoCard.php`
    Vista de inicio de sesión único con infocard.
  - `temp-getcardform.php`
    Vista para conseguir una infocard del IP.

- `./www/`
  Directorio con los controladores.

  - `./resources/`
    Carpeta con las imágenes que se presentarán en las vistas (logo de la infocard para el formulario) e imágenes que contendrán las infocards.
  - `getcardform.php`
    Controlador para que el usuario consiga su infocard.
  - `login-infocard.php`
    Controlador para hacer SSO en simpleSAMLphp con infocard.
  - `mex.php`
    Servicio de intercambio de metadatos.
  - `STS_card_issuer.php`
    Controlador de peticiones de URL de un solo uso para el servidor RADIUS y procesador de esas URLs para expedir tarjetas a los usuarios.
  - `tokenservice.php`
    Emisor de tókenes, servicio STS.

Funciones de usuario

Bajo el nombre de funciones de usuario, he agrupado las diversas funciones que bien por estar poco definidos los requisitos o por ser una prueba de concepto global no he querido extenderlas más allá de lo necesario ya que generaría un acoplamiento excesivo y una forma de trabajar poco adaptable. Como contrapunto a esa situación, las funciones de usuario definen unas cabeceras y una funcionalidad específica que debería de ser suficiente para que cualquiera con conocimientos de php readaptara esta solución a sus necesidades solo reescribiendo la lógica de las funciones. Obviamente el enfoque de universalidad es un poco iluso, pero por lo menos sirve para adaptarse al almacenén usado (SQL VS LDAP VS ficheros en el disco duro).

FICHERO: `./simplesaml/modules/InfoCard/lib/UserFunctions.php`

DESCRIPCIÓN:

Se define el objeto `sspmod_InfoCard_UserFunctions` que definirá una serie de métodos estáticos (no hace falta instanciar el objeto).

MÉTODOS:

1. `validateUser($credential,$type)`
   Recibe un credencial, el tipo de autenticación y si es correcto valida al usuario, devuelve verdadero. Se usa en el formulario de usuario/contraseña de conseguir una infocard o en el STS (tokenservice.php) para validar los credenciales.
2. `fillClaims($user, $configuredRequiredClaims, $configuredOptionalClaims,$requiredClaims)`

   Funcionalidad usada por el STS o tokenservice.php. Recibe el identificador de usuario, la configuración de los claims del sistema y los claims de la petición RST. Lo que hace es comprar los claims de la petición con los que tiene configurados y rellenar la respuesta con los datos del usuario, si se pide un claim que no existe, se pone un valor "unknown" (desonocido). En el código no aparece, pero los valores de los claims del usuario se podrían obtener de cualquier almacén de datos.
3. `generate_card_ID($user)`
   Recibe el identificador de usuario y devuelve su cardID asociado.
4. `fillICdata($user,$UserCredential,$ppid=NULL)`
   Recibe el identificador de usuario, la autenticación que desea usar, opcionalmente el PPID de la tarjeta autoemitida si la autenticación es por ese método y devuelve un vector con la información necesaria para generar su infocard. Opcionalmente se podría hacer que esa información se obtuviera de algún tipo de almacén, etc.

Funciones propias de simpleSAMLphp

SimpleSAMLphp define unas funciones propias que son bastante útiles e incluso necesarias a la hora de programar un módulo. A continuación describo los tipos que hay así como el uso que tienen.

Funciones de configuración y sesión.

- `$config = SimpleSAML_Configuration::getInstance()`
  Instanciar una copia de la configuración, directorio `simplesaml/config`.

  NOTA: *Función obsoleta, usar getConfig()*

- `$autoconfig = $config->copyFromBase('logininfocard', 'config-login-infocard.php')`
  Instanciar un objeto con la configuración del fichero `simplesaml/config/config-login-infocard.php`.
- `$var = $autoconfig->getValue('server_key')`
  Obtener un valor del fichero de configuración, en este caso `server_key`.
- `$var = SimpleSAML_Session::getInstance()`
  Cargar la sesión web del usuario.

Funciones de depuración y excepciones.

- `SimpleSAML_Utilities::fatalError($session->getTrackID(), 'texto')`
  Escribir mensaje de error. En mi caso lo hace en el fichero `/var/log/syslog`.
- `SimpleSAML_Logger::debug('texto')`
  Escribir mensaje de depuración. En mi caso lo hace en el fichero `/var/log/syslog`.
- `throw new SimpleSAML_Error_BadRequest('razón')`
  Lanzar una excepción de petición incorrecta. La razón se escribe en el fichero `/var/log/syslog`.

Funciones para las plantillas.

- `$t = new SimpleSAML_XHTML_Template($config, 'InfoCard:temp-login.php', 'InfoCard:dict-InfoCard')`
  argar una plantilla en la variable t. Está cargando la plantilla `simplesaml/modules/InfoCard/templates/default/temp-login.php` con el diccionario `simplesaml/modules/InfoCard/dictionaries/dict-InfoCard.php`.
- `$t->data['header'] = 'simpleSAMLphp: Infocard login'`
  Establecer un campo de la matriz data para la plantilla, en este caso es el título.
- `$t->show()`
  Mostrar plantilla cargada como página web. Se cede la ejecución a la plantilla.

Instanciación de objetos Se pueden instanciar objetos o invocar sus métodos estáticos de clases, para ello, se define el nombre de la clase como
`sspmod_<nombre del módulo>[_<carpeta>][_<carpeta>]..._<nombre descriptivo>`
y se guardará en un fichero `<nombre descriptivo>.php` en la carpeta `lib` del módulo.

Manejar la autenticación

Ya se ha visto que la funcionalidad del módulo de infocard es la de fuente de autenticación. En el fichero `simplesaml/modules/InfoCard/www/login-infocard.php` se puede apreciar que llega un momento en el que se invoca la siguiente orden.

```
sspmod_InfoCard_Auth_Source_ICAuth::handleLogin($authStateId, $_POST['xmlToken']);
```

Como se vio en el apartado anterior, lo que hace es llamar a la función `handleLogin` del fichero `simplesaml/modules/InfoCard/lib/Auth/Source/ICAuth.php`. La función hace lo siguiente.

1. Recibe el identificador de estado de autenticación en el sistema y el token XML con la identidad digital emitida por el IP.
2. Carga la configuración del módulo (certificados, claves, configuración de las infocards...).
3. Instancia un objeto Infocard y procesa el token.
4. Si el token es válido se procesan los claims y se autentica al usuario en el sistema con la orden `SimpleSAML_Auth_Source::getById($state[self::AUTHID])`. Los claims pasan a ser atributos de su sesión,
5. Si el token no es válido, se devuelve un mensaje de error.

Bibliografía

Recomiendo leer los siguientes documentos.

- simpleSAMLphp modules - Documentación oficial sobre la estructura de un módulo.
  http://simplesamlphp.org/docs/1.5/simplesamlphp-modules
- Creating authentication sources - Cómo funcionan las fuentes de autenticación.
  http://simplesamlphp.org/docs/1.5/simplesamlphp-authsource
- Information cards module for simpleSAMLphp - Documentación oficial del módulo de Infocard.
  http://simplesamlphp.org/docs/1.5/InfoCard:usage
- SimpleSAMLphp Documentation - Página oficial de la documentación de simpleSAMLphp.
  http://simplesamlphp.org/docs/1.5/

Capítulo 12

Modificaciones en eduroam

En este capítulo comentaré detalles prácticos de la infraestructura usada (eduroam) así como todas las decisiones técnicas que he tomado para integrar el metasistema.

Para situarnos mejor diré que voy a tratar el tema del "middleware", en castellano, el material de en medio. Son todos los programas o aplicaciones que no tienen un uso final más que servir de soporte o apoyo a otros programas. Por ejemplo, un programa que te conecta a una red no tiene sentido si no vas a usar esa red (con otro programa), eso es "middleware".

12.1. Conceptos

En la primera sección quiero presentar los conceptos fundamentales así como una visión global para entrar más tarde en modificaciones.

12.1.1. Funcionamiento actual

En el capítulo de eduroam (ver 9.3) expliqué qué es la arquitectura 802.1X, el servidor RADIUS y mencioné los métodos de autenticación que contempla EAP.

En eduroam se utilizan básicamente dos métodos de autenticación EAP basados en túneles TLS para proteger las credenciales.

EAP-TLS

Cliente y servidor se autentican mutuamente mediante certificados. Se requiere de una infraestructura de clave pública para la gestión de las credenciales (certificados). Muy segura pero engorrosa, ya que el uso de certificados por parte de los clientes no es algo "usable", además de que no siempre se conectan a la red desde la misma máquina. Es la opción minoritaria. El protocolo se especifica en:

- RFC 5216 - EAP-TLS Authentication Protocol
  http://www.rfc-editor.org/rfc/rfc5216.txt

EAP-TTLS

Es una extensión del método EAP-TLS, la diferencia estriba en que sólo se maneja un certificado. El cliente debe tener el certificado del servidor y con éste se establece el túnel. Se puede prescindir de

la infraestructura de clave pública y el cliente puede autenticarse con otros credenciales diferentes a un certificado, como con su nombre de usuario y su clave. Es el método más común para autenticarse en eduroam. El protocolo se especifica en:

- RFC 5281 - Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)
  http://www.rfc-editor.org/rfc/rfc5281.txt

12.1.2. Necesidad y opciones

La necesidad principal que justifica este capítulo es tener las dos siguientes funcionalidades.

1. Que el cliente sea capaz de enviar al servidor RADIUS el cardId o identificador de la tarjeta autoemitida que utilizará como credencial de la tarjeta gestionada que le expedirá indirectamente el IP del metasistema y constituirá su identidad digital.
2. Que el servidor RADIUS sea capaz de enviarle al cliente la identidad digital generada específicamente para ese contexto de uso.

Es decir, necesito comunicar cliente y servidor RADIUS de forma segura para intercambiar información sobre los credenciales e identidad digital que se usarán en el metasistema.

A priori, el protocolo EAP-TTLS parece una solución idóndea. Permite crear un túnel TLS por el que fluyen atributos o AVPs y además dentro de ese túnel se me permite usar el método de autenticación que desee.

El método de autenticación usado en la fase 2 es PAP. Se envían el identificador de usuario y su clave por el túnel TLS y el servidor RADIUS (AAA/H) responde con un mensaje de aceptación o rechazo que se traduce en un mensaje de aceptación o rechazo a nivel EAP para el punto de acceso (NAS).

El problema viene a continuación.

Tenga en cuenta que si el AAA/H incluye un mensaje de respuesta (Reply-Message) como parte de un mensaje de aceptación (Access-Accept) o rechazo (Access-Reject), el servidor TTLS no enviará este AVP al cliente. Mejor dicho, este AVP y cualquier otro AVP enviados por el AAA/H como parte de un mensaje de aceptación o rechazo, se mandarán al punto de acceso vía el protocolo de transporte AAA.

[11.2.5 PAP RFC, 2008b, último párrafo]

Es decir, no puedo mandar de vuelta la identidad digital del usuario en un mensaje de aceptación, ya que ese atributo o AVP será para el punto de acceso. Una opción es recurrir a los mensajes de desafío (Access-Challenge) que sí van en el tunel para el usuario, y contestar en el campo de contraseña (User-Password) manteniendo el campo de identificar de usuario (User-Name), pero es una opción poco elegante. Otra opción es usar otro método de autenticación, pero ser tan rígidos y estar estandarizados, me arriesgo a que me suceda lo mismo y acabar haciendo un parche para alguno de ellos. Hay que buscar otra alternativa.

Otro problema más simple, aunque no por ello menos importante es cómo voy a meter la identidad digital del usuario en el mensaje de aceptación. Tengo que decir que la comunicación entre el punto

de acceso y el servidor RADIUS está determinada por el protocolo RADIUS mediante mensajes UDP. Lo ideal sería meter la identidad en el mensaje de aceptación, en uno solo, sin fragmentar ese valor. Este requisito establece unas restricciones muy a tener en cuenta para saber qué características tendrá este campo. Ya sé que al tener un túnel TLS (requisito de seguridad fundamental) la integridad del mensaje de vuelta está asegurado, pero como se vió en su momento, eduroam consta de una estructura jerárquica de servidores y es deseable obtener una solución óptima para minimizar el tráfico global.

12.1.3. Soluciones

Formato de la identidad digital

Primero lo más sencillo, el formato de la identidad digital. A poco avispado que sea el lector, y sobre todo si se ha leído el capítulo anterior, ya sabrá que no es buena idea enviar en el mensaje de vuelta la infocard generada.

Como curiosidad quiero comentar que el tamaño medio de una infocard de las que genero son 26KB, gran parte de ese tamaño corresponde a la imagen que la acompaña (aun siendo un png con los colores inexados) y al material criptográfico (certificados).

Como solución se propone enviar esa identidad digital o infocard por referencia, es decir, enviarle al usuario una dirección URL secreta de un solo uso que indica dónde conseguir su identidad.

Analizo las ventajas.

- Es una forma genérica de enviar información, adaptable a otros contenidos.
- El campo tiene una longitud máxima acotada, en contraposición a una infocard, que puede variar según la imagen que se incluya, certificados, etc.
- Se minimiza el tráfico en los servidores RADIUS así como las modificaciones en los programas relacionados. ¡Hay que manejar la información!
- La generación de la infocard se realiza en un servicio del metasistema, algo más coherente que hacerlo en el acceso a la red.

Inconvenientes.

- Se alarga el ciclo de vida de la autenticación.
- Más comunicaciones y más tiempo es igual a más vulnerabilidades (potencialmente hablando).
- Conceptualmente se entiende peor el uso de una referencia que el de el objeto propio.

En realidad las ventajas superan con creces a los inconvenientes, tanto en un sentido práctico como estratégico.

Comunicaciones con PEAP

Para la comunicación cliente-servidor, recurriré a una extensión de EAP llamada PEAP (Protected Extensible Authentication Protocol). Aparentemente el protocolo PEAP no difiere mucho de EAP-TTLS, pero su principal característica es que permite tunelizar atributos en forma de TLVs (Tipo, Longitud, Valor) en contraposición a los AVPs de EAP-TTLS que podían ser o no ser filtrados por el

túnel TLS y variar así su destinatario, como pasaba en la autenticación PAP.

La decisión de utilizar PEAP se tomó después de una de mis visitas a RedIRIS y comentar los problemas que tenía con el servidor RADIUS. La recomendación fue la de ojear el proyecto DAMe (ver 9.4.2) y basarme en su código para desarrollar la funcionalidad requerida.

Realmente no soy un experto en la materia de EAP y las circunstancias de que no fuera capaz de conseguir el consejo de ningún gurú del tema así como las restricciones temporales para la entrega del código, han motivado que la única razón para haber utilizado PEAP es que funciona y ya se había hecho algo parecido con DAMe. El que mis intentos por haberlo hecho inicialmente con EAP-TTLS no fueran exitosos no son motivos suficientes para descartar esa opción, ya que potencialmente es posible la modificación.

En la figura 12.1 muestro una tabla comparativa entre PEAP, EAP-TTLS y EAP-TLS [diapositiva 28 Koren, 2003].

	PEAP	**EAP-TTLS**	**EAP-TLS**
Server Authentication	certificate	certificate	certificate
User identity protection	Yes, TLS	Yes, TLS	No
Cipher-Session negotiation	No	Yes, TLS	No
EAP Attacks: Session hijacking, Man-in the middle, Dictionary attack	Protected (TLS)	Protected (TLS)	Protected (TLS)

Figura 12.1: Comparativa entre PEAP, EAP-TTLS y EAP-TLS

Los aspectos técnicos que explico a continuación son un breve resumen del documento *[MS-PEAP]:Protected Extensible Authentication Protocol (PEAP) Specification* [Microsoft Corporation, 2010] y que recomiendo leer para profundizar en el tema.

Flujo de autenticacion

La mejor forma de familiarizarse con el protocolo es ver los pasos básicos que se llevan a cabo en una autenticación con PEAP.

1. Se establece una sesión EAP entre el cliente y el servidor.
2. Cliente y servidor negocian qué método EAP usar. Se acuerda usar PEAP versión 0.
3. PEAP entra en la fase 1. El propósito de la fase 1 es autenticar al servidor EAP y establecer una sesión TLS.
   a) Cliente y servidor intercambian mensajes TLS estableciendo registros TLS como datos de los mensajes PEAP.
   b) Se intercambiarán estos mensajes PEAP hasta que se establezca correctamente una sesión TLS entre el cliente EAP y el servidor EAP. Se completa la fase 1.

4. PEAP entra en la fase 2. Cliente y servidor continúan intercambiando mensajes PEAP con información TLS que es la que tuneliza y protege la información confidencial. El propósito de la fase 2 es permitir al servidor EAP autenticar al cliente dentro de la sesión TLS establecida en la fase 1.
   a) El servidor establece una nueva negociación EAP para autenticar al cliente. Este nuevo "método EAP interno" va dentro de los mensajes TLS que intercambian cliente y servidor.
   b) Cliente y servidor negocian y aceptan un método interno.
   c) Cliente y servidor intercambian mensajes internos hasta que el cliente es autenticado. Se completa la fase 2.
5. PEAP termina cuando se completa la fase 2.

La seguridad que ofrece la sesión TLS establecida en la fase 1 protege la autenticación del cliente en la fase 2, así que las contraseñas u otra información vulnerable a ataques por diccionario puede ser usada confidencialmente.

La figura 12.2 muestra un caso típico de arquitectura que usa PEAP. El cliente se autentica contra un servidor usando EAP a través de un punto de acceso o NAS. Los mensajes PEAP se envían del cliente al NAS usando protocolos de capa de enlace como PPP o IEEE 802.1X (eduroam), y del NAS al servidor usando el protocolo RADIUS.

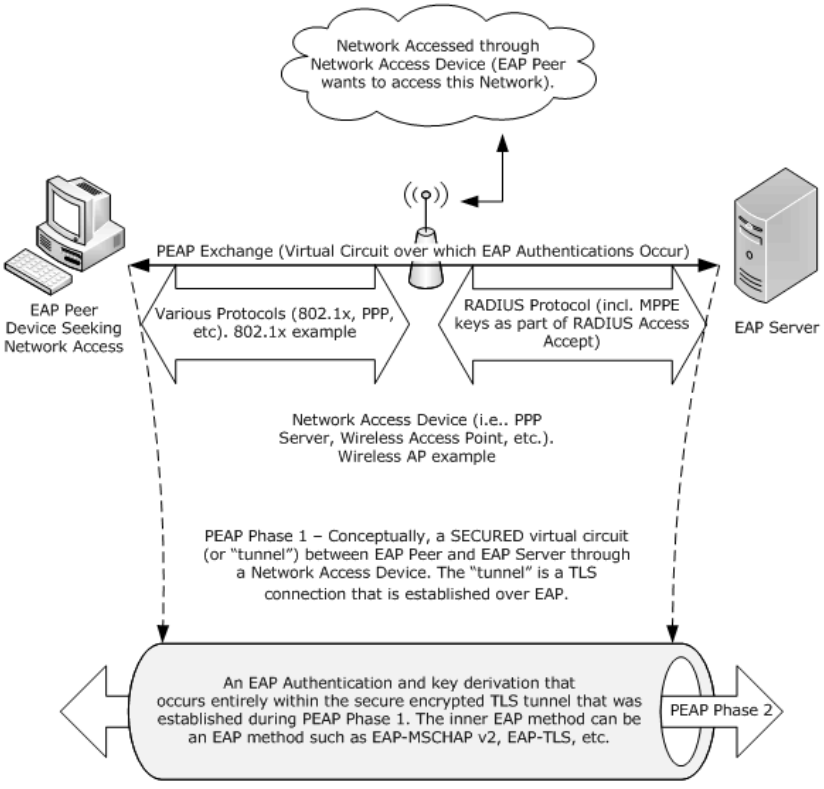

Figura 12.2: Caso de uso PEAP

Pila de protocolos

La figura 12.3 muestra la pila de protocolos que intervienen en cada una de las fases.

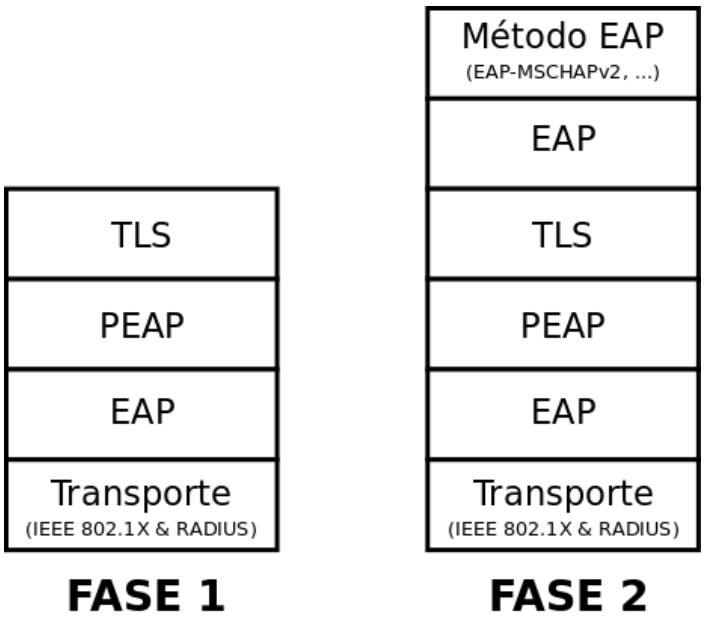

Figura 12.3: Pila de protocolos en las fases de PEAP.

Formato de los mensajes

Procedo a detallar el formato de los mensajes, algo que será de gran ayuda cuando haya que tocar código para mandar la información extra (cardId y URL de un solo uso).

Escribo los campos en el orden en que se mandan.

Detallo los formatos de los dos paquetes principales que se mandan durante la fase1.

- **Paquete EAP**
  - Código (1 byte): indica si el paquete es una petición o una respuesta.
  - Identificador (1 byte): identificador del paquete.
  - Longitud (2 bytes): longitud (en bytes) del paquete.
  - Tipo (1 byte): método EAP utilizado. Como el método es PEAP, este valor se fija a 25.
  - Datos del método (longitud variable): contiene la información referida al método, en este caso, un mensaje PEAP

- **Paquete PEAP**
  El paquete PEAP va dentro del campo `datos del método` del paquete EAP.

  - Banderas (6 bits):
    - L (1 bit): indica la presencia del campo `longitud del mensaje TLS`. Debe activarse en los mensajes TLS no fragmentados y en el primer fragmento de un mensaje fragmentado.
    - M (1 bit): se activa en todos los fragmentos excepto el último de un mensaje TLS fragmentado. Si el mensaje TLS encapsulado en PEAP no está fragmentado, no se activa.
    - S (1 bit): activado si el mensaje PEAP es de comienzo. Diferencia un mensaje de comienzo PEAP de otro de recibo (ACK). Sólo lo activa el servidor EAP en el primer mensaje que manda al cliente. El mensaje PEAP de inicio lleva la información de inicio de sesión (handshake) TLS.
    - R1 (1 bit): reservado. Desactivado, puesto a cero, e ignorado en la recepción.
    - R2 (1 bit): reservado. Desactivado, puesto a cero, e ignorado en la recepción.
    - R3 (1 bit): reservado. Desactivado, puesto a cero, e ignorado en la recepción.
  - Versión (2 bits):
    - R (1 bit): desactivado, puesto a cero, e ignorado en la recepción.
    - V (1 bit): indica la versión de PEAP, en este caso, 0.
  - Datos (longitud variable): si la bandera L está puesta a 0, este campo no debe estar presente. En otro caso, el campo se formatea como se indica a continuación.
    - Longitud del mensaje TLS (4 bytes): entero sin signo de 32 bits en representación de red (network-byte order) que indica el tamaño en bytes del campo `Mensaje TLS`.
    - Mensaje TLS (Longitud variable): paquete TLS (completo o fragmentado).

Una vez completada la fase 1, se crea un túnel TLS y se utiliza un método EAP interno. Dentro de estos mensajes se pueden incluir atributos en formato TLV así como métodos de autenticación (MSCHAPv2).

- **TLV**

  - M (1 bit): puede tener dos valores.
    - 0 Si el TLV no es obligatorio. El receptor puede ignorarlo si no es capaz de manejarlo.
    - 1 Si el TLV es obligatorio. Si el receptor no lo soporta, debe descartar el paquete PEAP completo (aunque contenga más TLVs).
  - R (1 bit): reservado. Desactivado, puesto a cero, e ignorado en la recepción.
  - Tipo de TLV (14 bits): entero sin signo de 14 bits en representación de red (network-byte order) que indica el tipo de información que contiene el campo `valor`. Si el campo `valor` contiene un TLV específico del vendedor, el tipo de TLV se fija a 7.
  - Longitud (2 bytes): entero sin signo de 16 bits en representación de red (network-byte order) que indica la longitud en bytes del campo `valor`.
  - Valor (variable): contiene la información que se quiere transmitir con el TLV y debe estar formateado en concordancia con el tipo especificado en el campo `tipo de TLV`.

- **TLV específico del vendedor**
  El cliente mandará a través del suplicante (programa con el que se conecta a eduroam) con el método PEAP y ya en la fase 2 de la conexión, un TLV específico con el cardId de su tarjeta autoemitida que se usará como credencial de la infocard generada y que representará su identidad digital como miembro de eduroam.

  Este TLV específico se construye como un TLV normal, siendo la única diferencia el cómo se formatea el campo `valor`.

  - Identificador del vendedor (4 bytes): identifica al creador/empresa/organización que utiliza ese TLV. En este caso el identificador de vendedor será `0x00 53 4D 48` que en en representación ASCII corresponde con mis iniciales, SMH (una pequeña licencia de autor).
  - TLV del vendedor (variable): TLV o conjunto de los mismos que pertenecen a ese vendedor.

12.2. El servidor RADIUS

Como servidor de autenticación RADIUS utilizaré el programa freeRADIUS. Es un proyecto de código abierto liderado por Alan DeKok y su página oficial es http://freeradius.org/.

Las principales características que han condicionado mi decisión por esta opción han sido las siguientes:

- Recomendaciones de gente versada en el tema.
- Es un software maduro que soporta multitud de protocolos EAP, incluídos EAP-TTLS y EAP, así como métodos de autenticación (PAP, CHAP, MSCHAPv2, ...).
- Ahora mismo es el servidor RADIUS más utilizado en eduroam.
- Posee una lista de correo para que los desarrolladores se comuniquen.
  Disponible en freeradius-devel@lists.freeradius.org.
- Es capaz de cargar guiones PERL para gestionar las peticiones. Sin duda esta es la opción más interesante ya que la productividad en este lenguaje es mucho más alta que la de C debido a su mayor abstracción.

También he encontrado ciertas desventajas, todo hay que decirlo, a posteriori.

- La documentación es escasa y confusa. Se centra sobre todo en la configuración para usarlo, nada para desarrolladores. Se da por supuesto que el usuario/desarrollador que utilice freeRADIUS es un experto en la materia. Esto jugó en mi contra y me llevó a "perder" casi un mes leyendo RFCs, consultando documentación ajena y realizando pruebas para lograr una configuración satisfactoria, ni siquiera óptima, sólo que funcionara.
- El soporte para el módulo de cargar guiones PERL estaba (¿está?) como algo experimental, por lo que no venía compilado en la distribución general. Tuve que compilar todo el programa con soporte para dicho módulo.
- No encontré muy útil la lista de desarrolladores. Formulé una pregunta un tanto compleja (agradezco que me contestaran) y la respuesta no fue muy satisfactoria, algo del estilo de: "Está en el código". Se puede ver el hilo aquí
  http://lists.freeradius.org/pipermail/freeradius-devel/2009-April/013030.html.

- El código es complejo, sin documentación ni comentarios, lo que propicia que no se unan nuevos desarrolladores al proyecto y el programa sea difícil de mantener.
- Existen planes para portar los servidores RADIUS de eduroam de freeRADIUS a RADIATOR http://www.open.com.au/radiator/. RADIATOR está ganando bastante inercia.

12.2.1. Configuración de freeRADIUS

Doy por supuesto que el lector sabe instalar un servidor freeRADIUS (en caso contrario recomiendo ojear el capítulo de puesta en marcha, concrétamente la parte 16.1.9 y procedo a comentar la configuración que usaré.

Existe documentación en la red de cómo configurar un servidor de este tipo, sin ir más lejos he encontrado el siguiente documento.

- Using FreeRADIUS with Eduroam
  http://www.gdt.id.au/~gdt/presentations/2008-05-29-eduroam-workshop/glenturner-eduroam-freeradius.pdf

Sin embargo y puesto que este proyecto es una prueba de concepto, hay parámetros que pueden no responder a una configuración estándar. Veamos qué consideraciones tengo que tomar en cuenta.

- El método EAP será PEAP.
- PEAP necesita crear un túnel TLS.
- Dentro del túnel se usará el método de autenticación MSCHAPv2.
- Usará el módulo de PERL para cargar un guión que gestione las peticiones. Esto me permitirá manejar los TLVs extras (cardId y URL de un solo uso) así como las conexiones con el almacén de datos.
- Necesita una comunicación segura con el punto de acceso.

Lo más sencillo es crear una carpeta para la configuración que se usará como parámetro a la hora de lanzar el servidor. Esta carpeta será una copia de la carpeta original de la configuración del servidor freeRADIUS que contiene una serie de ficheros que ofrecen una configuración por defecto que funciona. Veamos qué ficheros hay que modificar y cómo queda la configuración final.

NOTA: *Las líneas que empiezan por "#" en los ficheros de configuración, son comentarios, no tienen ningún tipo de efecto sobre la configuración.*

radiusd.conf

Se puede encontrar una copia la configuración utilizada en el anexo B.1.
Configuración general del servidor, enumero las opciones más interesantes.

- `confdir`: directorio de la configuración.
- `user/group`: usuario y grupo con el que se ejecutará el servidor.
- `listen`: especifica en qué direcciones IP y puertos se escucharán los servicios de autenticación, contabilidad...

- `log`: opciones de registro.
- `$INCLUDE $confdir/clients.conf`: carga el fichero de configuración de los clientes `clients.conf`.
- textttmodules: configuración de lo módulos que se cargarán.
  - `mschap`: módulo de autenticación por MS-CHAP.
  - `$INCLUDE eap.conf`: carga el fichero de configuración de EAP `eap.conf`.
  - `files`: carga diversos ficheros que contienen información almacenada sobre credenciales de usuarios, contabilidad, etc. Para este caso no es necesario.
  - `perl`: carga el módulo que interpreta guiones PERL con las siguientes opciones.
    - `module`: dirección del guión PERL con las instrucciones.
    - `func_authenticate`: nombre de la función del guión que se usará para manejar la autenticación.
    - `func_authorize`: nombre de la función del guión que se usará para manejar la autorización.
    - `func_preacct`: nombre de la función del guión que se usará para empezar la contabilidad.
    - `func_accounting`: nombre de la función del guión que se usará para manejar la contabilidad.
- `authorize`: lista de módulos que se invocarán para autorizar a un usuario. Cuando un usuario está en la fase de autenticación, se maneja la petición con el primer módulo, si falla se continúa con el siguiente y así hasta que uno lo autorice o todos fallen y se rechace al usuario.
- `authenticate`: lista de módulos que se invocarán para autenticar a un usuario.
- `preacct`: lista de módulos que se invocarán antes de comenzar la contabilidad.
- `accounting`: lista de módulos que se invocarán para llevar la contabilidad de un usuario.

clients.conf

En este fichero se especifican los clientes (NAS, puntos de acceso, etc) con los que se comunicará el servidor mediante el protocolo RADIUS.

En mi caso, por ser una prueba de concepto y utilizar una arquitectura canónica, sólo tendré un cliente. La sintaxis es la siguiente.

- Se define un cliente con un nombre, `client Infocard {}` que designará al NAS.
- Dentro de las llaves se establecerán dos campos
  1. `ipaddr`: es la dirección IP del cliente, sólo se aceptarán mensajes provinientes de IPs de clientes.
  2. `secret`: es el secreto compartido con el cliente. Sirve para cifrar las comunicaciones.

NOTA: *El protocolo RADIUS está basado en UDP, es un protocolo no conectivo que envía la información en claro. Puesto que estamos manejando información sensible, el método de filtrado por IP el secreto compartido ofrecen cierta privacidad y autenticación en las comunicaciones. El uso de la criptografía simétrica acarrea el problema de gestionar una gran cantidad de claves, y el filtrado por IP es una solución un poco chapucera a la autenticación (más aún en UDP). Por todo esto, se tiende hacia el uso de del protocolor RadSec que es un protocolo para transportar datagramas RADIUS utilizando TCP (conectivo) y TLS (privacidad y autenticación). Esta es una de las razones por las que RADIATOR se está convirtiendo en una opción a tener en cuenta.*

A continuación se muestra el contenido del fichero utilizado.

```
client InfoCard {
   ipaddr = 192.168.0.119
   secret = claveinfocard
}
```

eap.conf

En este fichero se especifica todo lo referente a la configuración EAP. Se puede encontrar una copia la configuración utilizada en el anexo B.2.

- `default_eap_type`: tipo eap a utilizar por defecto. En este caso, PEAP.
- `mschapv2`: carga del módulo para el manejo de MSCHAPv2.
- `tls`: carga del módulo para la gestión del túnel TLS:
  - `certdir`: directorio de los certificados.
  - `cadir`: directorio del certificado de la autoridad de certificación.
  - `private_key_file`: clave privada del certificado del servidor.
  - `certificate_file`: certificado del servidor.
  - `CA_file`: certificado de la CA.
- carga del módulo del método EAP, PEAP.
  - `default_eap_type`: método de autenticación que usará PEAP. En este caso, MSCHAPv2.

Vendor ID

Para facilitar el manejo de los nuevos atributos, se ha creado el diccionario eduroam. Es un fichero `/usr/local/share/freeradius/dictionary.eduroam` localizado en la carpeta de diccionarios de la instalación de freeRADIUS que contiene las características del los AVPs que utilizaré.

En un principio se pensó en un diccionario como el de la figura 12.4, pero viendo las características del método EAP-PEAP, el único atributo realmente usado es `EDUROAM-Card-URL` que se usa para meter la URL de un solo uso de vuelta al usuario en la respuesta de aceptación del servidor.

```
VENDOR eduroam      250

BEGIN-VENDOR   eduroam
ATTRIBUTE   EDUROAM-Card-ID     191 string
ATTRIBUTE   EDUROAM-Card-URL    192 string
ATTRIBUTE   EDUROAM-Auth-State  193 integer
END-VENDOR   eduroam
```

Figura 12.4: Diccionario con los AVPs de eduroam para RADIUS

12.2.2. Modelo de datos

Puesto que freeRADIUS es un servidor AAA (Autenticación, Autorización y Contabilidad), es sencillo pensar que debe tener conexión con algún almacén de datos que gestione los credenciales, permisos así como la información de los usuarios conectados.

El propósito del proyecto es el de desarrollar una prueba de concepto de integración de dos tecnologías, no he puesto un gran esfuerzo en desarrollar un esquema de datos sólido, genérico y que cubra todas las necesidades. Sin embargo y como acto simbólico el esquema desarrollado es suficiente para demostrar el potencial de la solución: manejar credenciales, controlar a los usuarios conectados y sus infocards, y demostrar que es posible controlar el almacén de datos desde el módulo de PERL de una forma más sencilla y potente que usando los módulos sql que dispone freeRADIUS.

El modelo de datos consta de las siguientes tablas.

- Una tabla de usuarios en la que se almacenan sus credenciales: nombre de usuario y contraseña. La clave primaria es el nombre de usuario.
- Una tabla de usuarios conectados a la red en la que se agregan usuarios cuando son aceptados en la red y se borran cuando se desconectan. Los atributos de esta relación son: el nombre de usuario (clave primaria), el cardId de la tarjeta autoemitida que se usará como credencial, el cardId de la tarjeta gestionada (identidad digital), el tiempo en que se inició la sesión y el método de autenticación.

NOTA: *No se debería almacenar el cardId de la tarjeta autoemitida, sino su PPID para que el STS lo utilizara al autenticar cuando el usuario usara su infocard. Error de perspectiva.*

En el anexo B.3 se muestra el código SQL para la creación del modelo de datos.

12.2.3. Script de PERL

Lo último que me queda por explicar del servidor RADIUS es el guión PERL que se encarga de gestionar las peticiones de autenticación.

El guión se almacena en un fichero (como se vió en el fichero de configuración (referencia)`radiusd.conf`) en el directorio de configuración del servidor bajo el nombre de `perl_IC.pm`. Se puede ver el código fuente en el anexo B.4.

Para comprender mejor los entresijos técnicos de cómo funciona el módulo (no así la lógica), recomiendo leer la documentación oficial de freeRADIUS sobre el módulo `Rlm perl`. En ella se explican las funciones principales, constantes, códigos de retorno, configuración, y cómo manejar los atributos. Está disponible en la siguiente dirección:
http://wiki.freeradius.org/Rlm_perl

Funciones auxiliares

Antes de entrar en la lógica principal del módulo, quiero explicar las funciones auxiliares que he desarrollado explícitamente para ganar abstracción.

get_InfoCard

PARÁMETROS DE ENTRADA:

- 0 - Nombre o identificador de usuario.
- 1 - CardId.

DESCRIPCIÓN

Lanza una petición de infocard al servicio de generación de tarjetas (ver 11.3.5) y devuelve la URL de un solo uso.

LÓGICA:

- Crea la petición con la forma `longitud.nombre de usuario.longitud.cardID`. Donde "." es el operador de concatenación y `longitud` es la longitud del campo que le precede.
- Cifra la petición con la clave simétrica compartida con el IP del metasistema.
- Codifica en base64 seguro para la web la petición y crea la URL para invocar el servicio, pasando la petición como parámetro. Tiene la forma:
  `https://.../module.php/InfoCard/STS card issuer.php?data=X&iv=Y&ident=RADIUS`.
  Donde X es la petición, Y es el vector de inicialización para descifrar el mensaje e ident indica que el emisor es el servidor RADIUS.
- Recibe la respuesta del servicio, la decodifica en base64 seguro para la web y la descifra usando la clave simétrica.
- Devuelve la respuesta.

get_Card_Id

PARÁMETROS DE ENTRADA:

- Toma el parámetro global `EAP-Message`

DESCRIPCIÓN

Procesa el mensaje EAP de dentro del túnel para obtener el TLV con el cardId enviado por el cliente. Solución muy dependiente de cómo mande el cliente la información.

LÓGICA:

- Se compara el mensaje EAP recibido `$RAD_REQUEST{'EAP-Message'}` con la siguiente expresión regular `/.{3}7.{4}00534D48.*/`. La expresión regular dice que busco un mensaje (TLV) con un 7 en la 4ª posición, indica que es de tipo TLV específico de vendedor y con `SMH` como identificador de vendedor. Si coincide es que me están mandando el cardId y continúo, si no, devuelvo nulo.
- Como vimos en 12.1.3, formateo el mensaje.
  - 2 bits M y R. No los tengo en cuenta.
  - 14 bits tipo de TLV. Ya he comprobado que es 7.
  - En verdad la información anterior la trato como 2 bytes.
  - 2 bytes de longitud del campo
  - 4 bytes del identificador de vendedor " SMH" o 0x00534D48.
  - CardID.
  - Devuelvo el cardId.

get_Password

PARÁMETROS DE ENTRADA:

- 0 Identificador o nombre de usuario.

DESCRIPCIÓN

Consulta en la base de datos la contraseña correspondiente al usuario y la devuelve.

connect_user

PARÁMETROS DE ENTRADA:

- 0 Identificador de usuario.
- 1 CardId de la tarjeta autoemitida.
- 2 Método de autenticación.

DESCRIPCIÓN

Inserta en la tabla de usuarios conectados al usuario junto con los datos proporcionados. Si el usuario figura como conectado, machaca ese registro y solo tendrá validez el nuevo. Ej: si existiera una reconexión.

disconnect_user

PARÁMETROS DE ENTRADA:

- 0 Identificador de usuario.

DESCRIPCIÓN

Borra de la tabla de usuarios conectados al usuario indicado.

Función principal

Realmente la única función que maneja la gestión de las peticiones es la de autorización.

authorize

PARÁMETROS DE ENTRADA:

- Parámetros del servidor RADIUS correspondientes a la petición.

DESCRIPCIÓN

Gestiona la petición de acceso a eduroam del usuario. Basándose en una máquina de estados.

LÓGICA:

- Es una máquina de estados que avanza según el RADIUS va evaluando la petición. A medida que se consiguen datos de la petición se van haciendo comprobaciones.
  Por ejemplo, cuando la petición entra en la fase 2 en MSCHAPv2 se obtiene la contraseña de usuario (función `get_Password`) y se fija en una variable propia del RADIUS para que la compare y se actualiza la máquina de estados. Se devuelve la constante `RLM_MODULE_UPDATED` para que se vuelva a invocar al módulo, ya que si falla se pasaría al siguiente, una y otra vez hasta que rechaza al usuario o lo acepta.
- Cuando se alcanza el estado final, se parsea el TLV `get_Card_Id` para obtener el cardId, se conecta al usuario con `connect_user` y se obtiene su identidad digital con `get_InfoCard`.
- El servidor le envía al usuario la URL de un solo uso para que se descargue su infocard (identidad digital) en el mensaje de aceptación usando el AVP `EDUROAM-Card-URL` definido en el diccionario de eduroam.

NOTA: *Codificar esta función fue un auténtico suplicio ya que no existe documentación de cómo funciona internamente la invocación del módulo, manejo de atributos, etc. El uso de módulos en PERL o Python está pensado para tareas sencillas, algo que entre al módulo y salga, no algo que se tenga que autoinvocar en cada fase.*
La metodología de trabajo consistió en una evaluación con ingeniería inversa sobre los atributos que el servidor añade, modifica o comprueba en cada fase. A partir de ahí desarrollé la máquina de estados, la reduje y codifiqué la función. Con esto quiero decir que aunque las comprobaciones puedan parecer arbitrarias o fuera de tiempo, no lo son, ya que realmente la invocación del módulo no se corresponde con el flujo de mensajes con el cliente y no se sabe exactamente en que punto de la comunicación se encuentra cuando se invoca.

Otras consideraciones

Funciones no relevantes.

Existen otras funciones en el módulo que aunque no se invocan, las he dejado por cuestiones estéticas. Estas funciones son: `authenticate`, `preacct`, `accounting`, `checksimul`, `pre_proxy`, `post_proxy`, `post_auth`, `xlat`, `detach`, `test_call`, `log_request_attributes`.

Parámetros de configuración
Al comienzo del módulo existen unos parámetros de configuración. Habrá que tenerlos en cuenta si se desea ajustar el funcionamiento del módulo.

- Configuración de la conexión con la base de datos:
  - `DB_name`: nombde de la base de datos que contiene las tablas mencionadas anteriormente.
  - `DB_username`: nombre del usuario con que se efectuará la conexión.
  - `DB_password`: contraseña del usuario con el que se se efectuará la conexión.
- Configuración del cifrado (define la comunicación con el servicio de generación de tarjetas del metasistema):
  - `key`: clave simétrica para cifrar las comunicaciones. Debe ser la misma que la configurada en el servidor en el campo `symmetric_key`. Ver 11.3.2.
  - `preurl`: dirección URL del servicio de generación de tarjetas, sin parámetros. Debería de tener la siguiete forma:
    `https://.../module.php/InfoCard/STS_card_issuer.php`.

Módulos usados
Para cerrar la sección quiero comentar los extra módulos usados y cuál es la funcionalidad que aportan.

- `MIME::Base64`: codificación/decodificacion en base64 de las respuestas del servidor de tarjetas.
- `MIME::Base64::URLSafe`: codificación/decodificacion en base64 seguro para la web de las peticiones del servidor RADIUS cuando las hace a través de la URL (método GET).
- `LWP::Simple`: funcionalidades HTTP para comunicaciones con el servidor de tarjetas.
- `Crypt::CBC`: criptografía simétrica para cifrar las comunicaciones entre servidores.
- `String::Random`: números aleatorios para crear vectores de inicialiación para el cifrado.
- `Digest::SHA`: funciones de resumen SHA. Para la clave de cifrado.
- `DBI`: funcionalidades para la comunicación con el sistema gestor de base de datos.

12.3. Suplicante

El suplicante es el programa que utilizará el usuario para conectarse a eduroam. Este software ofrece capacidades para manejar el protocolo 802.1X (autenticación WPA) y utilizar métodos de autenticación EAP como EAP-TLS o PEAP.

La elección del suplicante para Linux no fue una tarea muy difícil ya que solo contaba con dos candidatos.

- wpa_supplicant
  http://hostap.epitest.fi/wpa_supplicant/
- Open1X (anteriormente conocido como XSupplicant)
  http://open1x.sourceforge.net/

Para tomar la decisión tuve en cuenta ciertas características:

- Ambos soportan EAP-PEAP.
- Ambos son de código abierto.
- Ambos tienen licencia GPL/BSD. Mis intenciones no son comerciales.
- Open1X tiene versiones para Windows y para Linux, sin embargo, la versión para Linux era anterior y el proyecto parecía abandonado en ese sentido. Gran desventaja.
- wpa_supplicant es la opción mayoritaria en Linux y tiene una versión portada a Windows. Es el suplicante que siempre he usado y puedo manejarme con él con cierta soltura. Nunca he tenido problemas a la hora de compilarlo y es el que más promete. Gran ventaja.
- Open1X tiene interfaz gráfica mientras que wpa_supplicant funciona en modo texto (consola). Ventaja para wpa_supplicant porque aunque la usabilidad decae (luego veré como lo soluciono), las modificaciones en el código serán menores.
- Open1X está respaldado por la OpenSEA Alliance http://www.openseaalliance.org y se está impulsando para ser el suplicante por defecto en eduroam.

  - El cliente que se recomendaba para Windows, SecureW2, ha cambiado la licencia.

    Debido a un cambio en la licencia del software SecureW2, en la que la última versión ya no es gratuita y que se prohíbe la distribución pública y mientras la Comunidad Española de Eduroam busca alternativas, le proponemos que pruebe el programa Open1x. Dado que el programa está en constante desarrollo, en estos momentos (diciembre de 2009, fecha de creación de este documento), solo funciona correctamente con sistemas Microsoft Windows XP/32 bits, esperando que próximamente pueda funcionar en otros sistemas operativos posteriores a XP y también en versiones de 64 bits.

    [Universidad de Sevilla]

  - Recomiendo mirar la encuesta que hace Miroslav Milinovic (eduroam Croacia) sobre suplicantes y herramientas y en la que emite juicios como el siguiente.

    I think most institutions will pay for the license. It's something that the institutions decide for themselves. Some of them are upset a little, and it was not very well communicated by SecureW2 - but that might just have been easy anyway. In any case: there are more people interested in alternatives (like OpenSEA, or TLS) then ever.

    [Milinovic, 2010]

- Por último cito una presentación de RedIRIS en la que se trata la problemática sobre SecureW2 y qué soluciones tomar.

  Y nosotros, ¿qué hacemos?

  - *De momento habría que centrarse en estudiar las dos alternativas más a mano: SecureW2 1.x, y Open1X.*
  - *Investigar otras alternativas.*

  [Macías Luna, 2009]

Por todo lo anterior y ya que al ser este proyecto una prueba de concepto los aspectos técnicos priman sobre los políticos, mi elección es `wpa_supplicant`, la versión es la `0.6.8`.

12.3.1. Modificaciones

Antes de comentar las modificaciones que he realizado en el código, quiero dejar bien claro qué funcionalidades extras necesito en el suplicante.

- Tiene que ser capaz de recibir el cardId como parámetro.
- Tiene que ser capaz de enviar el cardId al servidor RADIUS; con el método EAP-PEAP, dentro de un túnel TLS en la fase 2 y como un TLV específico del vendedor.
- Tiene que recibir la respuesta del servidor RADIUS, la URL de un solo uso para conseguir la infocard, y mostrar esa información de alguna manera. ¿Por pantalla?

Metodología

La metodología que se ha seguido para implementar las funcionalidades ha sido la de realizar un proceso de ingeniería inversa sobre el código fuente del suplicante. La tarea fue larga y tediosa y se completó con ciclos de prueba-error hasta que todo funcionó correctamente.

Se pude acceder a la documentación del proyecto wpa_supplicant (código fuente, funciones, estructuras de datos, etc) en la siguiente dirección.
http://hostap.epitest.fi/wpa_supplicant/devel/index.html

Importar el parámetro cardId

La primera funcionalidad a implementar es la de tomar el parámetro cardId del usuario y suministrárselo al programa.

El suplicante admite un fichero de configuración para conectarse a la red en el que se especifican atributos como el ssid de la red (eduroam), el método de autenticación, o la interfaz de red a utilizar. Es en este fichero donde pretendo crear un campo `card_ID` para suministrarle dicho atributo al programa. En la figura 12.5 se muestra un ejemplo de como sería un fichero de configuración con el nuevo campo.
Se invocaría el suplicante con la siguiente orden.

```
eapol_version=1
ap_scan=1
fast_reauth=1
card_ID="CardId de la tarjeta autoemitida"
network={
   ssid="eduroam"
   scan_ssid=1
   key_mgmt=WPA-EAP
   pairwise=CCMP TKIP
   group=CCMP TKIP
   eap=PEAP
   phase1="peaplabel=0"
   phase2="auth=MSCHAPV2"
   identity="identificador del usuario en eduroam"
   password="contraseña del usuario en eduroam"
    ca_cert="ruta al certificado de la CA"
}
```

Figura 12.5: Ejemplo de fichero de configuración de wpa_supplicant

consola

```
# wpa_supplicant -dd -K -D <driver> -i <interfaz> -c <ruta al fichero de
configuración>
```

Donde:

- `-dd`: es el modo más verboso para sacar mensajes de depuración.
- `-K`: muestra las claves en el modo de depuración.
- `-D`: indica el controlador de la tarjeta WiFi.
- `-i`: indica qué interfaz de red utilizar.
- `-c`: indica la ruta del fichero de condiguración a utilizar.

Comento la traza principal de llamadas que sigue el programa hasta cargar la configuración y qué estructura de datos maneja. El formato es el siguiente `función [fichero]` y la ruta base es `wpa_supplicant-0.6.8/wpa_supplicant`.

- `main()` [main.c]
  Es la función principal del suplicante, la primera que se invoca. Parsea los parámetros de entrada, crea la variable `*global` que es una estructura de tipo `wpa_global` e invoca a:
  - `wpa_supplicant_add_iface(global, &ifaces[i])` [wpa_supplicant.c]
    Se encarga de agregar una interfaz de red con su configuración. Crea la variable `*wpa_s` que es una estructura de tipo `wpa_supplicant` y la pone como campo de esta variable `global`. Devuelve `wpa_s`. Invoca a:

- `wpa_supplicant_init_iface(wpa_s, iface)` [wpa_supplicant.c]
  Es la primera fase para iniciar la interfaz. De vuelve 0 o -1 según tenga éxito o error, pero lo que interesa es que modifica el parámetro `wpa_s` pasado por referencia. Cambia el campo `conf` de `wpa_s` a lo que devuelva la siguiente función.
  - `wpa_config_read(wpa_s->confname)` [config_file.c]
    Recibe el nombre del fichero de configuración. Crea la variable `*config` que es una estructura de tipo `wpa_config`. Va tomando líneas del fichero de configuración y las trata con diferentes funciones. La que me interesa es la siguiente función.
    `wpa_config_process_global(config, pos, line)` [config_file.c]
    Recibe la variable de configuración `config` por referencia y la modifica. Lo que hace es comparar la línea a tratar con una instancia del parseador tomando una variable del vector `global_fields` de tipo `global_parse_data`. Si hay éxito y se reconoce la línea, se asocia en la estructura el par tipo-valor.
- `wpa_supplicant_init_iface2(wpa_s)` [wpa_supplicant.c]
  Es la segunda fase para iniciar la interfaz. Recibe la variable `wpa_s` con el fichero de configuración parseado y la modifica poniendo el campo card_ID al valor obtenido del fichero. Acto seguido muestra por la salida de depuración el valor que tendrá el cardId. A continuación se muestra la modificación en la función.

```
1       wpa_s->card_ID = wpa_s->conf->card_ID;
2       wpa_printf(MSG_DEBUG, "CardID2 = %s", wpa_s->conf->card_ID);
```

- `wpa_supplicant_run(global)`
  Comienza la ejecución del programa con el cardId cargado en la variable global.

Las estructuras de datos usadas son las siguientes.

- `wpa_global` [wpa_supplicant_i.h]
  Información interna y global para todas las interfaces de wpa_supplicant.

- `wpa_supplicant` [wpa_supplicant_i.h]
  Contiene la información interna para el código del núcleo del suplicante, es donde me interesa tener la información ya que es accesible desde todas las funciones, incluída la de autenticación por PEAP.
  La he modificado añadiéndole el campo card_ID que es un puntero a caracter (cadena).

```
1   struct wpa_supplicant {
2       ...
3       //InfoCard
4       char* card_ID;
5   };
```

- `wpa_config` [config.h]
  Contiene la información de la configuración.

- `global_parse_data` [config_file.c]
  Estructura del parseador.

- `global_parse_data global_fields[]` [config_file.c]
  Realmente no es una estructura, sino información para la estructura del parseador. Aquí modifiqué el código para añadir la referencia al cardId.

```
1    static const struct global_parse_data global_fields[] = {
2       ...
3       //InfoCard
4       { STR(card_ID) }
5    }
```

Y ya está el carID cargado en la estructura principal y accesible desde los módulos.

Envío del CardId

La segunda funcionalidad es la de enviar el cardId al servidor RADIUS.

Para afrontar este reto, es necesario indagar en el código fuente hasta encontrar el fichero encargado de manejar método EAP-PEAP. Este fichero se localiza en `wpa_supplicant-0.6.8/src/eap_peer/eap_peap.c`.

Ya localizado, busco la funciones encargadas de gestionar la fase 2 y doy con una que parece bastante interesante.

```
1  static int eap_peap_phase2_request(struct eap_sm *sm, struct eap_peap_data *data,
2                struct eap_method_ret *ret, struct wpabuf *req, struct wpabuf **resp)
3  {
4     ...
5  }
```

Antes de ponerme con las modificaciones, necesito saber si la función hace realmente lo que creo, ya que no tiene comentarios ni documentación alguna. Hago un trazado inverso de llamadas y llego a la siguiente línea de invocación de funciones dentro del propio fichero.

- `eap_peap_process(struct eap_sm *sm, ...)`
  Parece la función de entrada del módulo PEAP. Muestra mensajes como *EAP-PEAP: Start...*, *EAP-PEAP: Using PEAP version ...* o *EAP-PEAP: TLS done, proceed to Phase 2.* La siguiente función a la que invoca en la traza es:

  - `eap_peap_decrypt(sm, data, ret, req, &msg, &resp)`
    Se encarga de establecer el túnel TLS así como de las comunicaciones internas. Es de suponer que la variable `resp` al ser pasada por referencia sea un valor a modificar con el mensaje a enviar al servidor. Cuando recibe un código `EAP_CODE_REQUEST` invoca a la función que estoy buscando.

    - `eap_peap_phase2_request(sm, data, ret, in_decrypted,&resp)`
      Se encarga de hacer la peticiones en la fase 2. Las variables que maneja y me resultan útiles son `*sm` que es un puntero a una estructura del tipo `eap_sm` (controla la máquina de estados de EAP) y `resp`, que será el mensaje de respuesta. Gracias a mensajes como *EAP-PEAP: Phase 2 Request: ...* puedo estar seguro de que estoy en la función correcta.

Haré la suposición de que puedo obtener el parámetro `card_ID` que cargo del fichero de configuración de la estructura `*sm`. Más tarde explicaré cómo lo hago.

Ya dentro de la función `eap_peap_phase2_request()` encuentro bifurcaciones según el el mensaje EAP recibido (de identidad, TLVs, tipo expandido...). Me sitúo en el caso por defecto del condicional `switch` y justo después de que se construya la nueva respuesta en `resp` y antes de salir del caso del condicional, introduzco la modificación de código.

```
...
//InfoCard
//TLV con el card id aquí
eap_tlv_add_cardid(resp, sm->card_ID, os_strlen(sm->card_ID));
...
```

Esta línea de código toma la respuesta (antes de que se envíe), el cardId obtenido de la estructura sm, su longitud y las manda a una función creada expresamente para la ocasión. Veamos el código de la función `eap_tlv_add_cardid(...)`.

```
/**
DE PRUEBAS
*/
static int eap_tlv_add_cardid(struct wpabuf **buf, u8 *card_id, u16 card_id_size)
{
   u16 i;

   //Room for a new TLV
   wpabuf_resize(buf, (size_t) 8 + card_id_size);

   //TLV Type (2 bytes: 0,7)
   wpabuf_put_be16(*buf, EAP_TLV_VENDOR_SPECIFIC_TLV); //Non mandatory

   //Length (2 bytes)
   wpabuf_put_be16(*buf, card_id_size);

   //Vendor_id (4 bytes: NULL,S,M,H)
   wpabuf_put_be16(*buf, 0x0053);
   wpabuf_put_be16(*buf, 0x4D48);

   //Data (Card ID, 64KB max)
   for(i=0; i<card_id_size; i++){
      wpabuf_put_u8(*buf, card_id[i]);
   }
   wpa_hexdump(MSG_MSGDUMP, "EAP-PEAP: Compound CardID TLV %s %d", card_id, card_id_size);

   return 0;
}
```

Lo que hace la función es tomar la variable de respuesta y agregarle el nuevo TLV con el cardId según el formato visto en 12.1.3. También muestra por pantalla, como mensaje de depuración, el valor del cardId. Lo curioso de la función es que utilizo funciones propias del programa como `wpabuf_resize(...)`, `wpabuf_put_be16(...)` o `wpa_hexdump(...)` para manejar tipos de datos internos, lo que indica que he llegado a tener cierta comprensión del código del programa.

Para dar por zanjada esta funcionalidad debo explicar cómo consigo obtener el `card_ID` de la estructura `sm`. Empieza el juego de asignaciones y punteros.

La variable `sm` es una estructura de tipo `eap_sm`. Dicha estructura está definida en el fichero `wpa_supplicant-0.6.8/src/eap_peer/eap_i.h`. La modificación a realizar es añadirle un campo para almacenar el cardId, como se muestra a continuación.

```
/**
 * struct eap_sm - EAP state machine data
 */
struct eap_sm {
   ...
   //InfoCard
   char* card_ID;
};
```

Ahora sólo hay que rellenar ese campo cuando se inicialice la estructura. Siguiendo la metodología, expongo una traza de las funciones que se invocan hasta que se llega a ese punto. La carpeta de referencia de los archivos será `wpa_supplicant-0.6.8`.

- `main()` [wpa_supplicant/main.c]
  Función principal del suplicante.

  - `wpa_supplicant_add_iface(global, &ifaces[i])` [wpa_supplicant/wpa_supplicant.c]
    Función que crea y configura una nueva interfaz de red.
    Crea la variable `wpa_s`.

    - `wpa_supplicant_init_iface2(wpa_s)` [wpa_supplicant/wpa_supplicant.c]
      Segunda fase de la inicialización de la interfaz.
      Añade card_ID a la estructura wpa_s (como se vio anteriormente).

      ```
      wpa_s->card_ID = wpa_s->conf->card_ID;
      ```

      Justo después de la orden anterior y antes de terminar, inicia la configuración de EAPOL.

      - `wpa_supplicant_init_eapol(wpa_s)` [wpa_supplicant/wpas_glue.c]
        Crea la variable `ctx` y hace la siguient asignación.

        ```
        ctx->ctx = wpa_s;
        ```

        Invoca a la función de inicialización de la máquina de estados de EAPOL.
        `eapol_sm_init(ctx)` [src/eapol_supp/eapol_supp_sm.c]
        Crea la variable `sm`, estructura de tipo `eapol_sm` y realiza la siguiente asignación.

        ```
        sm->ctx = ctx;
        ```

        Invoca a la función de inicialización de la máquina de estados de EAP.
        `eap_peer_sm_init(sm, &eapol_cb, sm->ctx->msg_ctx, &conf)` [src/eap_peer/eap.c]
        Crea la variable `sm`, estructura de tipo `eap_sm` que recibirá la función
        `eap_peap_phase2_request(...)`
        y de donde obtendremos el card_ID para enviarlo como TLV.

Dentro del fichero `src/eap_peer/eap.c` se llevan a cabo las modificaciones.

Para manejar los tipos de datos intermedios, incluyo los ficheros de cabeceras que contienen la definición de esos tipos.

```
//InfoCard
#include "../eapol_supp/eapol_supp_sm.h"
#include "../../wpa_supplicant/wpa_supplicant_i.h"
```

En la función `eap_peer_sm_init(...)`, una vez creada la variable `sm`, sólo tenemos que igualar su atributo `card_ID` con el de la estructura `wpa_s`. He añadido el siguiente código teniendo en cuenta las asignaciones comentadas en la traza de llamadas de funciones.

```
1   //InfoCard - Retrieving the card_ID value set prefiously in the parser
2      struct eapol_sm *pol_sm = (struct eapol_sm *) eapol_ctx;
3      struct eapol_ctx *ctx = pol_sm->ctx;
4      struct wpa_supplicant *wpa_s = (struct wpa_supplicant *)(ctx->ctx);
5
6      if (!wpa_s->card_ID) {
7         wpa_printf(MSG_WARNING, "WARNING: no InfoCard");
8      } else {
9         sm->card_ID = (char *)malloc(os_strlen((char*)wpa_s->card_ID));
10        memcpy(sm->card_ID,wpa_s->card_ID, os_strlen(wpa_s->card_ID));
11     }
```

Recepción de la URL de un solo uso

La última funcionalidad requerida es que el suplicante sea capaz de recibir el TLV con la URL de un solo uso del servidor RADIUS y mostrarlo de alguna manera.

Continuando con el módulo de PEAP, fichero `wpa_supplicant-0.6.8/src/eap_peer/eap_peap.c`, reviso la función encargada de gestionar la fase 2, `eap_peap_phase2_request(...)`, y me llama la atención los siguientes detalles:

- La función recibe como parámetro la estructura `wpabuf` bajo la variable `*req`.
- Dicha variable se analiza. Por lo que seguramente contiene un mensaje EAP proviniente del servidor RADIUS.
- Durante el análisis condicional, se evalúan los casos en que `req` sea un mensaje de tipo identidad, tipo TLV, tipo expandido, y por defecto (cuando incluyo el cardId). Ambos tipos corresponden a mensajes EAP tunelizados en fase 2.
- En el caso de que el mensaje EAP sea de tipo TLV, te invoca a la siguiente función:

  - `eap_tlv_process(sm, ... , req, resp, ...)`
    Se encarga de procesar los TLVs del mensaje EAP. Tiene un condicional que distingue entre el tipo de TLV. Las constantes para definir el tipo de TLV están declaradas en el fichero `wpa_supplicant-0.6.8/src/eap_common/eap_tlv_common.h`. Se puede ver que el valor de `EAP_TLV_VENDOR_SPECIFIC_TLV` es 7. Todo correcto.

    Creo un caso para el condicional en el que el tipo de mensaje EAP contenga un TLV específico del vendedor. Muestro a continuación el código añadido a la función.

```
1    //Parse InfoCard-URL
2    case EAP_TLV_VENDOR_SPECIFIC_TLV:
3       printf("\n\nEAP-TLV: Vendor Specific\n");
4       printf("\tSIZE: %d\n", tlv_len);
5       printf("\tVENDOR ID: %X%X%X%X\n", pos[0], pos[1], pos[2], pos[3]);
6       pos +=4;
7       left -=4;
8       char *InfoCardURL = malloc(tlv_len+1); //+1 because \0 end string character
9       memcpy(InfoCardURL,pos,tlv_len);
10      InfoCardURL[tlv_len] = '\0';
11      printf("\tDATA: %s\n", InfoCardURL);
12      free (InfoCardURL);
13      break;
```

El código muestra unos mensajes por pantalla como que ha recibido un TLV específico de vendedor, pone su tamaño, el identificador de usuario y por último la información.

NOTA: *Puesto que estamos en una prueba de concepto, esta no es una solución genérica, aunque se puede adaptar. No obstante, sabré que he tenido éxito cuando al ejecutar el suplicante, se muestre en la pantalla del ordenador del cliente que se ha recibido un TLV específico del vendedor, que el identificador de vendedor es "SMH" y por último, como campo DATA, la URL de un solo uso para descargar la infocard generada por el IP y enviada a través del servidor RADIUS.*

12.4. Conclusiones

La moraleja que he sacado de este capítulo es que siempre tiene más reconocimiento y es tarea más reconfortante el desarrollar una idea *desde cero* que el modificar o colaborar en una ya existente. Aunque no lo parezca, las modificicaciones en los programas que he tratado en este capítulo, me llevaron más tiempo que desarrollar desde la nada y sin conocimientos previos el módulo de Infocard para simpleSAMLphp. Sin embargo, la expectación y novedad se la lleva el metasistema aunque comparte título a partes iguales con eduroam.
Como colofón diré que:

- Los proyectos sobre los que he trabajado no tienen documentación más que la que se puede generar con herramientas de documentación automática como como doxygen.
- Ajustar los ficheros de configuración del servidor RADIUS es una tarea solo para expertos. La documentación trata los casos clásicos, pero un funcionamiento personalizado requiere algo más que la intuición clásica del ingeniero.
- El saber cómo funciona un modo no siempre es posible en un periodo de tiempo prudencial, de todas formas el proceso de ingeniería inversa sobre el código es tarea obligatoria.
- No existen diagramas con las llamadas entre módulos, funciones, o algo que sea verdaderamente útil.
- Cuando los porqués del funcionamiento de una aplicación se encuentran únicamente en la cabeza del autor de la misma, se producen dos efectos indeseables: la sobresaturación de peticiones de ayuda al autor, lo que acrecienta su ego y desprecio hacia la humanidad (vease el caso

RADIUS); y la escasez de desarrolladores que continúen con el proyecto por su complejidad (véase el `wpa_supplicant`, es un software maduro, bien estructurado, bueno en mi opinión, multiplataforma, pero con apoyo muy escaso y poca expectación a su alrededor en comparación a `Open1X`).

- A veces, las pruebas de concepto no distan tanto de lo que podría ser un producto final. Esta afirmación viene por la razón de que primero construí el producto y ahora que estoy redactando la memoria, esperaba encontrarme con algo bastante más chapucero de lo que hay.
- *Eppur si muove (y sin embargo se mueve)*

Sea o no de Galileo Galilei

.

Capítulo 13

El conector

13.1. Sobre la usabilidad

Situándonos en el desarrollo actual del proyecto, se puede decir que lo difícil ya está hecho. Hay un metasistema como módulo de una herramienta de SSO, se comunica con eduroam, y se han realizado los cambios necesarios tanto en el suplicante como en el servidor RADIUS para manejar los nuevos datos. La prueba de concepto está terminada.

¿Entonces cuál es el problema?

La aceptación de una tecnología por el gran público reside en gran medida no en lo avanzada que sea sino en lo sencilla que resulte (tanto en uso como conceptualmente) en la primera impresión. En una palabra, usabilidad.

El problema que existe ahora mismo es para el usuario. ¿Cómo va a aceptar el gran público esta propuesta cuando es mucho más compleja de utilizar que la que existía anteriormente? Veamos los pasos que debe dar el usuario.

- Tener configurados los certificados de su selector de identidad.
- Saber el cardId de la tarjeta autoemitida que usará como credencial de la gestionada.
- Crear un fichero de configuración apropiado para el suplicante.
- Lanzar el suplicante.
- Leer la URL de un solo uso de la salida del suplicante.
- Obtener la infocard gestionada usando la URL anterior.
- Importar la nueva infocard que será su identidad digital en el selector de identidad.

Claramente la secuencia anterior no es una opción viable para la comunidad universitaria. La gente lo que quiere es conectarse, no realizar un ritual de conexión.

Basándome en las necesidades acumuladas, creo que la solución óptima es realizar una pequeña pieza de "software", llamémosle **conector**, que automatice lo máximo posible el proceso anterior.

Quiero que el caso de uso sea el siguiente:

1. Un usuario abre su portátil y descubre un punto de acceso marcado como `eduroam`. Está en el área de cobertura.

2. El usuario lanza el conector, este le hace unas preguntas sencillitas, y poco después ya está conectado a eduroam y con su infocard gestionada cargada en el selector de identidad.

13.2. Sobre Perl

El lenguaje de programación usado para implementar el conector es Perl (http://www.perl.org/). Las razones para su elección han sido las siguientes.

- Perl es un lenguaje interpretado, el ciclo de prueba y error es más corto.
- El intérprete de Perl viene por defecto en muchas distribuciones de Linux, además es de código abierto y por consiguiente portable y multiplataforma.
- La abstracción que ofrece el lenguaje redunda en la productividad.
- Está pensado para manipular texto, expresiones regulares, parseadores...
- Contiene módulos suficientes (más de 18.000) como para cubrir cualquier necesidad.

Además ya usé este lenguaje para programar el módulo de autenticación del servidor FreeRADIUS y sintácticamente se parece a PHP.

13.3. Guión de ejecución

Procedo a comentar los entresijos y funcionalidades del guión Perl que automatiza la conexión. Se puede ver el código fuente completo en el apéndice C.1.

13.3.1. Programas necesarios

Lo que hace el guión es recopilar información de programas y del usuario, y aprovecharla para establecer la conexión y configurar la identidad digital. Los programas requeridos son los siguientes.

- **Perl:** es el intérprete del guión.
- **ifconfig**: es el configurador de interfaces de red del sistema. Se usa para activar la interfaz.
- **iwconfig**: es el configurador de interfaces inalámbricas del sistema. Se usa para saber qué interfaces tienen capacidades IEEE 802.11.
- **gksu**: se utiliza para lanzar procesos como superusuario (configuración de certificados, configuración de interfaces de red, etc).
- **zenity**: múestra ventanas de diálogo por pantalla. Sirve para interactuar gráficamente con el usuario. Ej: pedir credenciales, mostrar mensajes, etc.

13.3.2. Variables

Las variables principales son las siguientes.

- Opciones de usuario
  - `$LANG`: lenguaje configurado en el sistema.
  - `$ssid`: nombre del punto de acceso. Como es una arquitectura de pruebas, se llama `INFOCARD`, pero en producción sería `eduroam`.
  - `$iface`: interfaz de red WiFi de conexión.
  - `$driver`: controlador de la interfaz de red a usar por el suplicante.
  - `$WIFI_DRIVERS`: lista de controladores que soporta el suplicante.
- Credenciales de conexión y seguridad
  - `$username`: nombre de usuario en eduroam.
  - `$password`: contraseña en eduroam.
  - `$CONF_CA_CERT`: certificado de la CA que autoriza al IP que firma la infocard que importará el usuario.
  - `$card_ID`: cardId de la tarjeta autoemitida que servirá de credencial a la generada y que se enviará al servidor RADIUS.
- Rutas a programas de apoyo
  - `$DM_path`: ruta al selector de identidad
  - `$WPA_SUPPLICANT_path`: ruta al suplicante `wpa_supplicant`.
  - `$IFCONFIG_path`: ruta al programa `ifconfig`.
  - `$IWCONFIG_path`: ruta al programa `iwconfig`.
- Ficheros temporales
  - `$WPA_SUPPLICANT_CONF_file`: ruta para el fichero temporal de configuración del suplicante que creará el guión.
  - `$IC_file`: ruta al fichero temporal donde se almacenará la infocard obtenida de eduroam.
  - `$DM_pipe`: nombre de la tubería para conectar el guión con el registro del selector de identidad. Sirve para comprobar los certificados y conseguir el cardId.
  - `$WPA_pipe`: nombre de la tubería para conectar el guión con la salida del suplicante. Sirve para conseguir la URL de un solo uso.
- Plantillas
  - `$WPA_SUPPLICANT_CONF`: plantilla de la configuración para del suplicante.

13.3.3. Módulos

Ya que casi todo lo que manejo es texto, registros y mensajes de programas, solo he necesitado usar un módulo.

- LWP::Simple
  http://search.cpan.org/~gaas/libwww-perl-5.834/lib/LWP/Simple.pm
  Ofrece la funcionalidad necesaria para manejar peticiones HTTP. (EJ: descargar la infocard a través de la URL de un solo uso).

13.3.4. Lógica

El guión se divide en una secciones muy estrictas, veámoslas.

Comentarios, carga de variables y selección de diccionarios.

Líneas: 1-68

Description:
Se inicializan o definen las variables. La sección de diccionarios (líneas 64-68) es una idea de cómo se podría realizar una internacionalización de los mensajes para adecuarlos a la configuración del idioma del sistema.

Declaración de funciones.

Líneas: 72-79

Description:
Se declara la función `checkcert` que comprueba si un certificado cualquiera es el de la CA (variable `$CONF_CA_CERT`).

Invocación del selector de identidad.

Líneas: 83-163

Description:

- Se crea la tubería especificada en `$DM_pipe` (línea 85).
- Se le muestra un mensaje informativo al usuario para que sepa que tiene que seleccionar una tarjeta autoemitida (línea 87).
- Se duplica la ejecución del programa con un *fork* (línea 90). Se invoca al selector de identidad en el proceso hijo y se redirige su salida de depuración (parámetro `--loglevel=debug`) a la tubería (línea 94).
- En el proceso padre se espera hasta que el selector de identidad empieza a escribir en la tubería (línea 103). Sé que es una espera activa, no pasa nada.
- Hago uso de las capacidades de Perl para manejar texto y expresiones regulares y con una máquina de estados (líneas 102-135) voy comprobando los certificados que carga el selector de identidad y los voy comprobando con el de la CA con la función `checkcert`. Si la función devuelve verdadero, se sale de la máquina de estados.
  - Si se llega en el log al final de la zona de certificados (línea 104), es porque no se ha importado el certificado de la CA. El usuario no lo tiene importado.
    Se lanza un mensaje de error diciendo que no dispone de dicho certificado y el guión se ofrece para instalarlo en el sistema (línea 108). ¡Automatización al poder! El usuario puede recharzlo y acabará la ejecución del guión o aceptarlo, momento en el que se instalará el certificado en el fichero `/usr/share/digitalme/certs/ca-bundle.crt`.
    Ver 11.1.2.
  - Los valores de la máquina de estados son los siguientes.
    - **0:** Estado inicial. No se ha tomado ningún certificado.

    - **1:** Estado de validación. Se está leyendo un certificado del log, cuando se acaba, se comprueba si es el de la CA. Si coincide, se sale de la máquina, estado 2, si no, se pasa al estado inicial 0.
    - **2:** Estado de finalización, el usuario tiene cargado el certificado de la CA.
    - **3:** Estado de finalización, el usuario no tiene cargado el certificado de la CA.
- En este punto está solucionado el tema del certificado de la CA. Se supone que si todo ha ido bien, el caso genérico es que el usuario no reciba mensaje alguno en el paso anterior, el usuario tendrá el selector de identidad abierto junto con sus tarjetas. Sólo tiene que seleccionar una autoemitida y darle a ver los valores o pulsar sobre ella dos veces (hace lo mismo).
- Lo que sucede es que mientras el usuario realiza la acción anterior es que el guión sigue analizando el registro del selector con una nueva máquina de estados (líneas 143 a 158). Intenta conseguir el cardId de una tarjeta autoemitida.
  Existen tres estados.
  - **0:** Estado inicial.
  - **1:** Se recibió una línea del tipo *"Importing unmanaged token card ..."*, se está cerca del carId esperado. Se vuelve al estado 0 si existe algún problema como que se seleccione una tarjeta gestionada o se realice otra acción que no es la esperada.
  - **2:** Estado final, se consiguión el cardId y se sale de la ejecución.
- Una vez conseguido el cardId, se cierra el selector, se cierra la tubería y se borra la misma (líneas 161-163).

Obtención de las credenciales de autenticación.

Líneas: 168 y 169

Description:
Se le muestra al usuario dos ventanas, una para obtener su nombre de usuario en eduroam y otra para obtener su contraseña y así rellenar las variables `$username` y `$password`.

Configuración de la interfaz inalámbrica.

Líneas: 171-190

Description:

Se invoca al programa especificado en `$IWCONFIG_path` y se analiza la salida para obtener una lista de las interfaces que soportan el estándar IEEE 802.11.

Si la lista contiene más de una interfaz, se le pregunta al usuario en una ventana cuál quiere utilizar.

Se le muestra una lista (configurada en `$WIFI_DRIVERS`) al usuario para que seleccione el controlador de tarjeta a usar con el suplicante.

Punto de control.

Líneas: 193

Description:
Ya se tiene el cardId, las credenciales de autenticación en eduroam (usuario/contraseña) y la configuración de la tarjeta inalámbrica. Se muestra un mensaje de depuración con estos datos.

Invocación del suplicante y carga de tarjeta.

Líneas: 201-282

Description:

- Se configura la plantilla de configuración del suplicante `$WPA_SUPPLICANT_CONF` con los datos obtenidos anteriormente (líneas 203-222).
- Se crea la tubería especificada en `$WPA_pipe` (línea 225).
- Se duplica la ejecución del programa con un *fork* (línea 227). Se invoca al suplicante en el proceso hijo y se redirige su salida a la tubería (línea 234).
- En el proceso padre se procesa la información que llega a la tubería con una máquina de estados.
- En algún momento el suplicante escribirá en la tubería algo como:

```
1 EAP-TLV: Vendor Specific
2 VENDOR ID: 0534D48
3 DATA: http://...?data=...&iv=...
```

- Cuando se llega a la última línea (DATA:...), se alcanza el estado final (línea 260), se descarga la infocard con la URL de un solo uso (línea 265) y se guarda en un fichero (líneas 269-271). Acto seguido se invoca al selector de identidad con la infocard para que la importe y se borra el fichero de la infocard (líneas 272 y 273).

Borrado de archivos temporales.

Líneas: 281 y 282

Description:
Se borra el fichero de configuración y la tubería que conecta el guión con el suplicante.

13.4. Conclusiones

El conector es una buena idea, demuestra que es posible crear una herramienta que facilite enormemente al usuario la tarea de la conexión. No obstante, su principal desventaja es que es dependiente de los mensajes y formato que emiten los programas, lo que provoca un serio acoplamiento. Recuerde el lector que este proyecto es una prueba de concepto, las soluciones tomadas están pensadas para que funcionen en el marco contextual del mismo.

Capítulo 14

Conclusiones

Es momento de analizar globalmente el proyecto y emitir un juicio.

14.1. Seguridad

La seguridad es un aspecto a tener en muy en cuenta (incluso más que la usabilidad) ya que ahí reside la confianza en toda la arquitectura que he montado. Explico las posibles pifias cometidas, potenciales vulnerabilidades y recomendaciones.

- El aspecto más vulnerable es cómo se maneja el cardId de la tarjeta autoemitida que servirá de credencial para usar la gestionada. Este valor se extrae del registro de depuración del selector de identidad, se maneja con el guión, se introduce en un fichero de configuración que usará el suplicante, se manda a un punto de acceso, recorre la jerarquía RADIUS hasta llegar al servidor y de ahí se reenvía al servicio correspondiente del IP para generar el PPID asociado. Sin duda es un camino demasiado largo.
  ¿Por qué no generar el PPID en el cliente y enviar ese valor? En mi caso lo hice del modo anterior por comodidad (manejo de certificados) y evitar complejidad en el guión Perl.

- Siguiendo con la tarjeta autoemitida y aumentando la seguridad... Se recomienda no usar el PPID como único credencial de una tarjeta gestionada. Esto es que aunque en la tarjeta gestionada aparezca el PPID como como referencia a la infocard a usar, el IP no sólo debe comprobar el PPID, sino usarlo como un índice para acceder a la clave pública que almacenó cuando registro la infocard autoemitida. La seguridad de una infocard autoemitida reside en el par de claves RSA asociado que se genera cuando se crea la tarjeta, de forma que al registrar una tarjeta, se envía el certificado público de la misma.

- Otro aspecto que me preocupa es la seguridad en la configuración del servidor RADIUS y las modificaciones realizadas en el código. Sospecho que la URL de un solo uso se envía como AVP no tunelizado y podría ser vulnerable a escuchas en el canal. No puedo garantizarlo porque no he realizado las pruebas de seguridad pertinentes (o un ataque práctico) y no he encontrado a nadie que realmente supiera cómo funciona el servidor RADIUS y mucho menos que hubiera usado el módulo de Perl.

- Se supone que el sistema no es vulnerable en el tema de las comunicaciones web ya que todas hacen uso de canales TLS (https).

- Tampoco es posible importar una tarjeta maliciosa puesto que van firmadas. Lo mismo sucede con los tokens que admiten las RPs. Sobre la confianza entre una RP y un STS recomiendo mirar la siguiente página
  http://dev.eclipse.org/mhonarc/lists/higgins-dev/msg03938.html.

- Existen dos problemas asociados a eduroam que aunque caen fuera del ámbito del proyecto, es necesario tener en cuenta.

  1. ¿Cómo saber cuándo se desconecta un usuario?
  2. ¿Cómo detectar un abuso en la red por parte de un usuario?

  Ambos problemas fueron cuestionados por Tomasz Wolniewicz en la presentación de esta arquitectura en la TNC2009. Su ponencia fue inmediatamente anterior a la de Enrique de la Hoz y no desaprovechó la oportunidad. Recomiendo ojear su página de la TNC2009
  http://tnc2009.terena.org/schedule/presentations/show276c.html?pres_id=43
  así como sus transparencias sobre el tema [Wolniewicz and Górecka-Wolniewicz].

  Identity management of users in eduroam
  Roaming users in eduroam are able to hide their identity from the visited institution. eduroam service as a whole has sufficient means to identify offending users in serious incidents, however there are cases where visited institutions should be able use available authentication data to distinguish users and recognise returning ones. We discuss a possible, privacy preserving, solution through the use of the Changeable-User-Identity RADIUS attribute and describe an implementation in FreeRADIUS server and some testing tools.

  TNC2009

  La segunda pregunta excede los límites de este proyecto, pero la primera se planteó durante la finalización del prototipo.
  Las soluciones que propuse fueron:

  1. Implementar un demonio en el punto de acceso que registrara los usuarios que había, y cuando uno se desconectara, lo notificaba al RADIUS. Algo poco práctico porque existe itinerancia ("roaming") en la red, reautenticaciones, y un montón de puntos de acceso con características diferentes. En el router liksys que usé era viable hacerlo porque reprogamé su memoria interna para que su sistema operativo fuera Linux.
  2. Establecer un periodo de caducidad de las infocards. Opción más elegante para la gestión de la identidad pero que no resuelve realmente el problema de cuándo darlas de baja.

- Es curiosa la recomendación del equipo operacional de eduroam de no usar los credenciales de conexión en formularios web.

  If you come across a web page that asks you to login to a wireless network and it includes the eduroam name or logo, please be aware that this is in violation of the eduroam policy. In this case you are advised not to use the web page and instead to contact the relevant institution?s helpdesk for assistance with connecting to eduroam.

  [eduroam, c]

- Por último quiero citar el trabajo sobre vulnerabilidades en los selectores de identidad de Rick Smith en la Internet Identity Workshop del 2009. *Four risk areas:*

  1. *Native code running on client systems, and/or plug-ins on a browser. Attackers can substitute subverted code and intercept personal memorized secrets that secure the cards, or that are used with IDPs to authenticate a managed identity.*
  2. *Network based attacks - forged transactions or modified transactions used to spoof identity. Most implementations rely on SSL to protect on these. Not all protocols require SSL in all circumstances where it is needed.*
  3. *Subverted or malicious relying party - can the RP turn around and exploit the user's identity to masquerade to another RP?*
  4. *Spoofed IDP - a variant of the network based attack - can an attacker trick the RP into authenticating a user by intercepting IDP transactions and providing a bogus response*

  [página 10 Internet Identity Workshop 9, 2009]

¿Por qué no he corregido el proyecto con todas estas recomendaciones? Por la sencilla razón de que es un trabajo de final de carrera y una prueba de concepto. No me puedo pasar toda la vida haciéndolo, tengo que vivir de algo.

14.2. Predicciones y trabajo futuro

Es momento de lanzar unas valoraciones de lo que habría que desarrollar para potenciar esta tecnología.

- Quizá el primer paso sea el desarrollo de unas librerías más genéricas que las que propongo para poder manejar los aspectos necesarios relacionados con las infocards. Partiendo de esas librerías se podrían desarrollar herramientas tales como:
  - Plugins para la autenticación con infocard para plataformas como Wordpress, Drupal, Joomla, etc.
  - Proveedores de identidad con todas las capacidades que ofrece la tecnología.
  - Aplicaciones web para la gestión de la identidad digital.
- Selector de identidad. Aquí me desmarco del concepto del CardSpace y propongo un selector de identidad de verdad.
  - Multiplataforma, de código abierto. Debería ser capaz de funcionar en dispositivos móviles, ahí está el futuro.
  - Compatible con Information Cards, OpenId, y otras formas del manejo de la identidad digital.
  - Al que se le puedan agregar extensiones o complementos (plugins) para ampliar las funcionalidades.
    - Complemento para autenticación con códigos bidimensionales. Autenticación al estilo http://bidikey.com con códigos QR de un solo uso.

- Complemento para gestionar entradas, facturas, billetes de transporte, pagos, etc.
- Complemento para comunicación con otros programas.
  - Con el suplicante para obtener el cardId (o el PPID).
  - Con un programa que gestione todos los certificados instalados en el sistema.
  - Con el navegador para iniciar automáticamente el SSO sin infocard.

- Suplicante. Lo interesante sería tener un suplicante que soportara completamente PEAP para gestinar el envío del cardId y la recepción de la URL de un solo uso, y en cooperación con el selector de identidad, conectar a eduroam. El proceso se llevaría a cabo desde el escritorio mediante una interfaz gráfica (con su respectivo complemento para eduroam). Sin necesidad del conector. Personalizando la experiencia de usuario y cumpliendo la máxima de *abre el portatil y conéctate*.

- Otro aspecto interesante es el de simpleSAMLphp. El reciente módulo `self-register` http://rnd.feide.no/content/new-simplesamlphp-module-selfregister que permite al usuario registrarse y gestionar los datos que almacena de él un Idp, en combinación con OpenId e Infocard como fuentes de autenticación más un generador de tókenes (rol de IP en el metasistema) y otro de infocards, lo hacen una opción bastante potente para cubrir casi cualquier necesidad en el campo de la identidad digital en la web. Sin duda tiene un futuro prometedor.

- Como última predicción, quiero mencionar que el futuro de la identidad digital, como el futuro de todo, estará marcado por el beneficio que sea capaz de generar. En términos prácticos se traduce en dinero (seguridad=confianza=dinero, automatización=tiempo=dinero, facilidad de uso=tiempo=dinero). No es por tanto extraño que con el auge de las plataformas móviles (IPhones, Androids y demás sistemas) y las ultravulnerables tarjetas de crédito, lleguemos a ejemplos como el siguiente.

  Ejemplo(22) Estoy en un supermercado y me toca pagar en la caja registradora. La cajera me pregunta: -"¿En efectivo, tarjeta o transacción virtual?".

  Nótese que "transacción virtual" es el nuevo método que usaré y que aunque ahora es un nombre desafortunado, es tarea de los que se dedican al "marketing" el bautizarlo como se merece.

  Elijo transacción virtual y la cajera me presenta un código bidimensional cifrado por el supermercado con el resumen de mi compra, la factura. Lo escaneo con mi teléfono, se descodifica, interpreta y si estoy conforme, se emite orden de pago al banco a través de internet y haciendo uso del selector de identidad instalado en mi terminal (puedo tener varias cuentas con diferentes entidades). Me aparece otro código bidimensional que escanea la cajera y contiene la orden de pago firmada por el banco y todo firmado por mi. El sistema podría ser tan rápido como dejar el teléfono móvil unos segundos sobre un mostrador específicamente adaptado en caja. Rápido, seguro, no hay repudio posible, he hecho uso de la identidad digital y todo está automatizado. Esto es el futuro.

  Es más, podría tener una aplicación de contabilidad que registrara dichas facturas y me ayudara a planificar los gastos. Pero esto ya es otro tema.

14.3. Conclusión final

Después de haber hecho este proyecto y haber visto durante dos años cómo ha ido evolucionando el tema, diré que la tecnología de Information Cards, es potente, buena para los desarrolladores, cubre bastantes necesidades, es extensible pero no va a triunfar por las siguientes razones.

- Se concibe como un artificio de Microsoft aunque en verdad es un estándar abierto. El problema reside en la política de la compañía de **Adoptar, extender y extinguir** (véase http://es.wikipedia.org/wiki/Adoptar,_extender_y_extinguir).
  Lo que sumado a que el único selector de identidad que funciona en condiciones es su CardSpace, no hace sino suponer que sacarán alguna funcionalidad no reflejada en el estándar y fastidie al resto de usuarios. Típico caso de monopolio.
- Otro problema es que se requiere un selector de identidad. Un programa con el que el usuario no está familiarizado, y si encima tomamos "selector de identidad" como eufemismo de "gestor de contraseñas" e implicamos a Microsoft, el rechazo se magnifica (véanse los comentarios de http://www.meneame.net/story/microsoft-esta-desarrollando-forma-oficial-extension-para-firefox).
- Asimilar el concepto de que haya una tercera parte que confirme una identidad no es algo que produzca excesiva confianza entre la gente no versada en el tema de la seguridad.
- OpenId se empieza a establecer como la opción ganadora: tiene un gran apoyo en la red, no requiere de programas adicionales, pero no ofrece tantas funcionalidades.
- Creo firmemente que si infocard quiere tener alguna posibilidad, debe centrar sus esfuerzos en las plataformas móviles. En los ordenadores de sobremesa tiene la desventaja de que las tarjetas no van de pc a pc con el usuario (sí que se pueden transportar, pero es un engorro). Con un selector de identidad en el dispositivo móvil, la cosa puede cambiar mucho.

Capítulo 15

Pliego de condiciones y presupuesto

En este capítulo haré una estimación del coste que hubiera tenido para una empresa este proyecto. Al ser éste un proyecto de investigación y desarrollo, hay costes que no se pueden calcular con exactitud, como por ejemplo el tiempo de formación en las nuevas tecnologías y el coste de toda la bibliografía consultada. No obstante, la estimación de tiempos junto con el material utilizado es la parte más significativa.

Ejecución material

Mano de obra

Desarrollo del prototipo

- 7 meses * 20 días/mes * 6 horas/día = 840 horas
- 40 €/hora

TOTAL: 33.600 €

Redacción de la memoria

- 7 meses * 20 días/mes * 5 horas/día = 700 horas
- 40 €/hora

TOTAL: 28.000 €

Material

Equipos

- Estación de trabajo: 1500€
- Cables de red: 20€
- Router: 80€
- Impresora láser: 100€
- Toner: 70€

TOTAL: 1770€

Software

Todo el software utilizado es gratuito y de código abierto.

- Sistema operativo: `GNU/Linux (Debian)`
- Servidor web: `Apache`
- Sistema Gestor de BB.DD.: `PostgreSQL`
- Entorno gráfico: `KDE 3.5`
- Editor de texto: `Kate/Kwrite`
- Intérprete de PHP: `PHP`
- Herramienta de SSO: `simpleSAMLphp`
- Servidor RADIUS: `freeRADIUS`
- Selector de identidad: `DigitalMe`
- Suplicante: `WPA Supplicant`
- Editor LaTeX: `Kile`
- Visor PDF: `KPDF`
- Intérprete de Perl: `Perl`

TOTAL: 0€

TOTAL: 63.370€

Gastos generales y beneficio industrial

Gastos derivados de la utilización de las instalaciones de trabajo y el beneficio industrial. Se estima como el 16 % del coste de ejecución material.

TOTAL: 10.139,2€

Coste de ejecución por contrata

- **Ejecución material** 63.370€
- **Gastos generales y beneficio industrial:** 10139.2€

TOTAL: 73.709,2€

Importe total

- **Coste de ejecución por contrata:** 73.509,2€
- **I.V.A.(16 %):** 11.761,42€

TOTAL: 85.270,64€

El presupuesto total del proyecto asciende a la suma de:
ochenta y cinco mil doscientos setenta euros con sesenta y cuatro céntimos.

Alcalá de Henares, 13 de mayo de 2010
FDO: Samuel Muñoz Hidalgo
Ingeniero Informático

Capítulo 16

Puesta en marcha

16.1. Manual de despliegue

NOTA: *Esta es una guía que describe en términos generales cómo instalar todo el sistema para probarlo. Doy por supuesto que el aventurado probador tiene ciertos conocimientos como para solventar él solo bastantes minucias y probablemente algunos problemas gordos que azarosamente no se me presentaron.*

16.1.1. Arquitectura

Para recrear un entorno básico de acción, se necesitarán dos máquinas. Un PC básico con sistema operativo Linux (preferiblemente aunque no necesariamente Debian) que hará de cliente y servidor, y un punto de acceso (A.P. Router WiFi) que soporte autenticación RADIUS.

16.1.2. Nomenclatura

- **sp:** Service Provider o proveedor de servicios, quien ofrece un servicio, recurso o lo que sea y quiere identificar a los usuarios.
- **idp:** Identity Provider o proveedor de identidad, quien autentica a los usuarios.
- **sts:** Secure Token Service, puede ser una tercera parte que se encarga del intercambio de información.
- **CA:** Certification Authority o autoridad de certificación, de quien me fío a la hora de aceptar certificados.
- **AP:** Access Point o punto de acceso. Será un enrutador WiFi que dará acceso a los clientes de eduroam.

16.1.3. Hosts

El primer paso será definir en el PC las IPs asociadas a cada servicio que vamos a ofrecer. Para ello editamos el fichero '/etc/hosts'y añadimos las siguientes líneas.

```
192.168.0.20   sts.ic
192.168.0.21   idp.ic
192.168.0.22   sp.ic
```

16.1.4. Interfaces virtuales

Editar el fichero '/etc/network/interfaces'y añadir las siguientes 4 interfaces virtuales.

```
#STS
auto eth0:sts
iface eth0:sts inet static
name IFAZ-STS
address 192.168.0.20
netmask 255.255.255.0
breadcast 192.168.0.255
gateway 192.168.0.1
network 192.168.0.0


#IDP
auto eth0:idp
iface eth0:idp inet static
name IFAZ-idp
address 192.168.0.21
netmask 255.255.255.0
breadcast 192.168.0.255
gateway 192.168.0.1
network 192.168.0.0


#SP
auto eth0:sp
iface eth0:sp inet static
name IFAZ-SP
address 192.168.0.22
netmask 255.255.255.0
breadcast 192.168.0.255
gateway 192.168.0.1
network 192.168.0.0


#RADIUS
auto eth0:radius
iface eth0:radius inet static
name IFAZ-RADIUS
address 192.168.0.23
netmask 255.255.255.0
breadcast 192.168.0.255
gateway 192.168.0.1
network 192.168.0.0
```

16.1.5. Certificados

Instalar el paquete openssl para la generación de certificados.
Ejecutar el script de generación. Ver Apéndice D.1
Generaremos 4 certificados. El de la autoridad de certificación y tres firmados por esta: el del IDP, el del STS y otro para el servidor RADIUS.
Por convenio los archivos *.crt son los certificados y los *.key las claves privadas de dichos certificados.
Recuerda que el Common Name (CN) del certificado debe ser el mismo que la raíz de la dirección de la web a llamar. Por ejemplo, el certificado para el idp tendrá un CN = idp.ic que es el que pusimos en el fichero hosts y el sts tendrá CN = sts.ic

16.1.6. Apache 2

Instalamos el servidor web Apache 2.

Módulo SSL

Activamos los módulos de SSL.

consola

```
1 /etc/apache2/mods-enabled# ln -s ../mods-available/ssl.conf ssl.conf
2 /etc/apache2/mods-enabled# ln -s ../mods-available/ssl.load ssl.load
```

Configuración de certificados

Crear la carpeta ssl en el directorio de configuración de apache, en este caso es '/etc/apache2/ssl'. Copiar en esta carpeta los certificados que generamos previamente: CA.crt, idp.crt, idp.key, sts.crt, sts.key.

Configuración de los hosts virtuales

En la carpeta '/etc/apache2/sites-available'crearemos los siguientes archivos que seran los servicios que ofreceremos:

- **idp** (Apéndice D.2.1)
- **sp** (Apéndice D.2.2)
- **sts** (Apéndice D.2.3)

Activamos los sitios enlazándolos simbólicamente en la carpeta '/etc/apache2/sites-enabled'.

consola

```
1 /etc/apache2/sites-enabled# ln -s ../sites-available/sts 001-sts
2 /etc/apache2/sites-enabled# ln -s ../sites-available/idp 002-idp
3 /etc/apache2/sites-enabled# ln -s ../sites-available/sp 003-sp
```

16.1.7. PHP

Habrá que instalar los siguientes módulos de PHP: Hashing function, ZLib, OpenSSL, SimpleXML, XML DOM, RegEx support, MCrypt, uuid, pgsql.
En mi caso, al instalar PHP se configura automáticamente para su uso con apache.

16.1.8. simpleSAMLphp

Copiar la carpeta simplesaml al disco duro. En mi caso a '/home/infocard/simplesaml'.
En la carpeta metadata, dentro del directorio simplesaml, editar los siguientes ficheros:

- **saml20-idp-hosted.php** (Apéndice D.3.1)
- **saml20-idp-remote.php** (Apéndice D.3.2)
- **saml20-sp-hosted.php** (Apéndice D.3.3)
- **saml20-sp-remote.php** (Apéndice D.3.4)

En la carpeta config, dentro del directorio simplesaml, editar los siguientes ficheros. authsources.php config.php config-login-infocard.php

16.1.9. FreeRADIUS

Copiar la carpeta del freeradius a un directorio temporal.
En la carpeta ejecutar:

consola

```
./configure
make clean
make
make install
```

Por defecto, se instala en la carpeta '/usr/share/freeradius'.
Copiar la carpeta de configuración 'freeradius-conf', lo normal sería hacerlo a /etc, pero en este caso, como es de pruebas, la sitúo en '/home/infocard/freeradius-conf'.
Copio el fichero "dictionary.eduroam" a la carpeta '/usr/local/share/freeradius'.
Edito /usr/local/share/freeradius/dictionary y agrego `$INCLUDE dictionary.eduroam`.

NOTA: *Hay que garantizarle los permisos justos a la carpeta de configuración '/home/infocard/freeradius-conf', si todo el mundo es capaz de escribir en ella, el servidor se negará a arrancar.*

PERL

Casi toda la lógica que atañe al servidor RADIUS está escrita en un módulo PERL, por ello habrá que instalar este entorno en la máquina sobre la que se ejecute. Para saber qué módulos instalar, recomiendo mirar las primeras líneas que empiezan por úse'del fichero '/home/infocard/freeradius-conf/perl_IC.pm'.

16.1.10. PostgreSQL

Instalo el sistema gestor de bases de datos PostgreSQL.
Edito el fichero '/etc/postgresql/8.3/main/pg_hba.conf'para permitir el acceso via IP y de forma local (en este caso lo pondría como `trust`) según las necesidades.
Ver ejemplo del fichero pg_hba.conf en el Apéndice D.4

Ejecutar el script de carga 'carga'. Ejemplo en el Apéndice D.5

16.1.11. Punto de acceso

En mi caso, el AP tiene la dirección 192.168.0.119.
Configuramos la seguridad WiFi para trabajar con RADIUS (ver figura: 16.1).

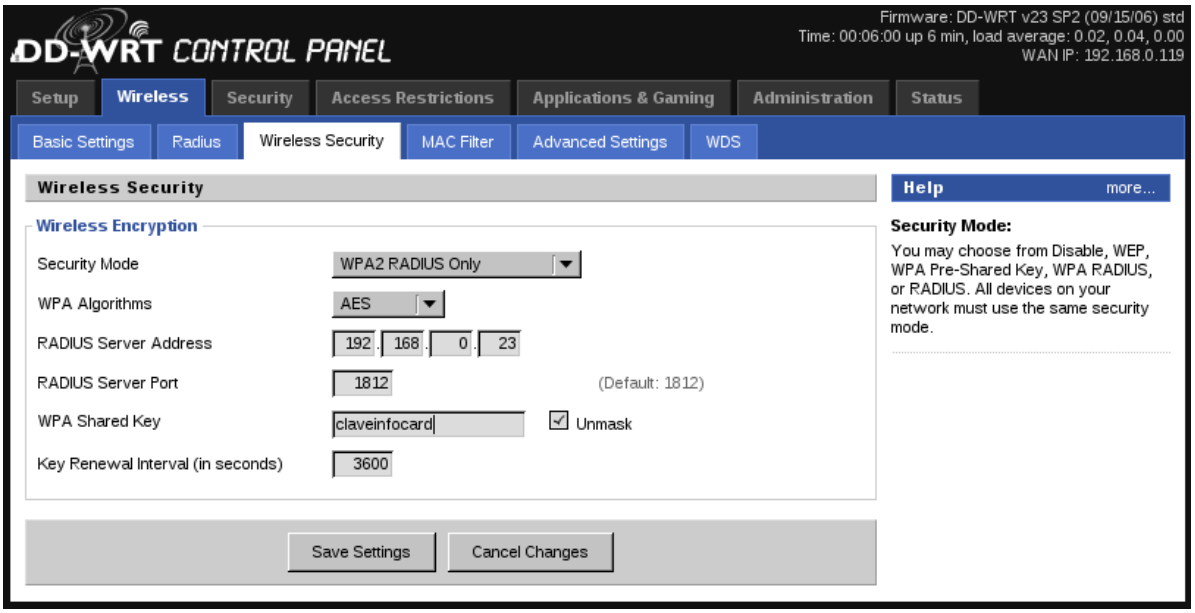

Figura 16.1: Seguridad en el AP

16.1.12. Pasos finales

Configuración del RADIUS

Editamos el fichero '/home/infocard/freeradius-conf/clients.conf'y ponemos que escuche mensajes provenientes de la IP del AP.

```
client InfoCard {
   ipaddr = 192.168.0.119
   secret = claveinfocard
}
```

Editamos '/home/infocard/freeradius-conf/radiusd.conf'y verificamos que el servidor escuche en la interfaz e IP correctas, queda algo como:

```
#Authentication
listen {
   ipaddr = 192.168.0.23
   port = 1812
   type = auth
   interface = eth0
}

#Accounting
listen {
   ipaddr = 192.168.0.23
   port = 1813
   type = acct
   interface = eth0
}
```

Editamos el fichero '/home/infocard/freeradius-conf/perl_IC.pm'y configuramos las variables de conexión a la BB.DD y al servidor simplesaml.
Copiamos los certificados: CA.crt, servidor-radius.crt y servidor-radius.key a la carpeta '/home/infocard/freeradius-conf/certs'. Editar el fichero '/home/infocard/freeradius-conf/eap.conf'y configuramos las siguientes líneas.

```
private_key_file = ${certdir}/servidor-radius.key
certificate_file = ${certdir}/servidor-radius.crt
CA_file = ${cadir}/CA.crt
```

Configuración del simpleSAMLphp

Editar '/home/infocard/simplesaml/config/config-login-infocard.php'.
Configurar la sección de DATABASE OPTIONS.
Configurar la sección STS Card Issuer, especialmente symmetric_key que comparte con el radius y configuredIP para que acepte conexiones de la ip del radius.

Reiniciando servicios

La infraestructura está montada, reiniciaremos los servicios.

consola

```
# /etc/init.d/./apache2 restart
# /etc/init.d/./postgresql-8.3 restart
# /etc/init.d/./networking
# ifup eth0
```

NOTA: *Hasta aquí, el servidor estaría completamente montado y funcionando. Procedo a describir cómo instalar la parte del cliente.*

16.1.13. WPA_Supplicant

Copiamos a nuestra carpeta personal el suplicante WPA. Queda en '/home/infocard/wpa_supplicant-0.6.8'.

16.1.14. PERL

Copiamos la carpeta PERL al directorio personal. Queda '/home/infocard/PERL'. Instalamos el entorno PERL y los módulos correspondientes. Una ayuda para saber qué módulos instalar es abrir el fichero '/home/infocard/PERL/conector.pl'y ver qué líneas empiezan con úse'.
También serán necesarios los siguientes programas: zenity, gksu, iwconfig, etc.
Para configurar este conector, lo editamos y habrá que especificar la dirección del ejecutable del suplicante, del selector de identidad y de la herramienta de configuración de red.

```
#----> GENERAL
$DM_path='/usr/bin/digitalme';
$WPA_SUPPLICANT_path = '/home/infocard/wpa_supplicant-0.6.8/wpa_supplicant/./wpa_supplicant';

$IWCONFIG_path = '/sbin/iwconfig';
```

16.1.15. DigitalME

El DigitalME será el selector de identidad que se usará para manejar las Ínformation Cards'.

Paquete

Vamos a la carpeta DigitalME e instalamos el paquete 'digitalme.deb'.

consola

```
# dpkg -i digitalme.deb
```

Aquí pueden suceder dos cosas:

- Se instala correctamente.
- Empieza a dar errores. Expongo una lista de los más comunes:

- Librerías que no existen. Suele ser el más común. Instálalas y haz enlaces simbólicos con el nombre del fichero que espera encontrar a la versión que has instalado.

- Que falta algún programa como el 'gnome-keyring-manager', el 'gksuó cualquier otro. Instálalo.

- Has probado todo y no funciona. Vas a http://code.bandit-project.org/trac/wiki/DigitalMe, ojeas un poco, te bajas el paquete que te corresponda y haz algún ritual de buena suerte.

Complemento para el navegador

En la carpeta DigitalME, seleccionamos el fichero 'digitalme-firefox-0.5.2571.xpi'y le damos a abrir con Firefox/ICeweasel. Se abrirá el diálogo para instalar esta extensión en el navegador.
En el caso de que no funcionara, te redirijo a la url de los responsables del selector de identidad: http://code.bandit-project.org/trac/wiki/DigitalMe.

16.1.16. Pasos finales

Con todo ya instalado, sólamente tendrías que ejecutar el fichero 'conector.pl', contestar a lo que te pida y te conectará a la red y te dará una InfoCard para que te autentiques en los servicios web que permitan esta opción.

NOTA: *Si has llegado hasta aquí con una conexión exitosa, siéntete satifecho porque es realmente difícil. Es más, apreciaría un correo con tu experiencia y los problemas que te han surgido y soluciones que has encontrado para complementar este manual.*

16.2. Problemas y respuestas

PROBLEMA 1

Al iniciar el RADIUS me sale algo relacionado con PERL del estilo:

consola

```
1 Can't load '/usr/lib/perl/5.10/auto/Data/Dumper/Dumper.so' for module Data::Dumper:
  /usr/lib/perl/5.10/auto/Data/Dumper/Dumper.so: undefined symbol: Perl_sv_cmp at
  /usr/lib/perl/5.10/XSLoader.pm line 64.
2 at /usr/lib/perl/5.10/Data/Dumper.pm line 36
```

RESPUESTA

Este error lo solucioné invocando el RADIUS con la siguiente orden.

consola

```
1 # LD_PRELOAD=/usr/lib/libperl.so radiusd -X -d /home/infocard/freeradius-conf/
```

PROBLEMA 2

El digitalme muestra un mensaje de error al intentar importar una tarjeta.

RESPUESTA

Lo más probable, si el sistema no está en fase de desarrollo, es que se deba a que no tienes importado el certificado de la autoridad de certificación en el fichero 'ca-bundle.crt'. Importarlo es tan sencillo como copiar este certificado al final de dicho fichero. En mi caso:

consola

```
1 # cat /etc/apache2/ssl/CA.crt >> /usr/share/digitalme/certs/ca-bundle.crt
```

PROBLEMA 3

Importé el certificado pero sigue dando error al importar tarjeta.

RESPUESTA

El problema puede deberse a varias causas, que no lea bien el ca-bundle.crt, que la infocard esté mal compuesta (no tenga una estructura correcta de XML), etc. Mira la sección de cómo depurar con el digitalme.

PROBLEMA 4

¿Cómo puedo depurar con el digitalme?

RESPUESTA

Tiene un parámetro (`--loglevel=debug`) que muestra un registro por pantalla.
La siguiente orden hace que el registro de esta ejecución del digitalme se escriba en el fichero dmetraza.txt en la carpeta tmp en vez de en la consola.

consola

```
$ digitalme --loglevel=debug 2> /tmp/dmetraza.txt
```

Apéndices

Apéndice A

Sic transit gloria mundi

A la vista de suficientes ojos, todos los errores resultan evidentes.

Linus Torvalds, (1997)

Hasta un reloj parado, tiene razón 2 veces al día.

Anónimo

No esperes alcanzar la perfección sin haber conocido el error.

anónimo

La razón de este anexo es documentar todas las "novedades" que surgieron después del prototipo y durante la escritura de la memoria. Bien porque exigieran un gran cambio de código o no se ajustaran a las características de algún capítulo, aquí las recopilo.

A.1. Proyecto Moonshot

El proyecto Moonshot es una iniciativa englobada en el marco de Terena que pretende extender los límites del SSO y la identidad federada más allá de la web.
Para más información recomiendo leer el siguiente documento escrito por Josh Howlet, JANET(UK):
http://www.terena.org/mail-archives/mobility/pdfEKnl2kkFsw.pdf

A.2. Selector para OpenID

Siguiendo la estela del selector de identidad para infocard, se ha propuesto la misma idea para OpenID, también llamado cliente activo. Lo que se intenta es extender la funcionalidad de los selectores de identidad ya existentes para incorporar este método.
Se puede encontrar más información en
http://self-issued.info/?p=235.

A.3. Identidad digital y la masa

La clave para que la gente aproveche las ventajas de la identidad digital es que tome conciencia de la privacidad de sus datos. Sin embargo hay todavía visionarios que intentan imponer un punto de vista contrario.

People have really gotten comfortable not only sharing more information and different kinds, but more openly and with more people. That social norm is just something that has evolved over time... But we viewed that as a really important thing, to always keep a beginner's mind and what would we do if we were starting the company now and we decided that these would be the social norms now and we just went for it.

Mark Zuckerberg, creador de Facebook

En la siguiente página se puede ver la evolución en la privacidad que han sufrido los usuarios de Facebook en los últimos cinco años
http://mattmckeon.com/facebook-privacy/.

No quiero hacer una disertación filosófica sobre ello, pero como lo que realmente vende es el miedo, presentaré la siguiente situación.

Ejemplo(23) Supongamos que vivo en el centro de una ciudad, en un barrio de los de toda la vida, que gracias a la migración de las familias a urbanizaciones residenciales, se está llenando de población inmigrante. Como es inevitable, y bajo circunstancias de convivencia entablo una relación de amistad con un vecino (totalmente normal e integrado) de origen marroquí llamado Mohamed. El tío de Mohammed, que vive en Rabat, es un ferviente musulmán que como es normal, profesa su religión en la mezquita local. Hasta aquí todo normal.

Por circunstancias de la vida, me toca hacer un viaje a Israel, Reino Unido, EE.UU. (aquí su país más preocupado por el terrorismo) y en la aduana me retienen. Me explican que estoy vinculado con el terrorismo islámico y que por consiguiente me retienen bajo los acuerdos internacionales de la ley antiterrorista. Empieza un calvario de abogados y cosas peores (¿vacaciones en Guantánamo?).

¿Qué ha pasado?
Podría ser que el imán de la mezquita a la que va el tío de Mohammed estuviera vinculado directamente con una cédula terrorista. ¡Existen tres grados de separación! Suficiente para que se levanten las sospechas sobre mí.

¿Cómo se han dado cuenta?
Mohammed es un amigo mío en Facebook, él también es sospechoso. No es difícil hacer minería de datos en una red social, además es más barato el que la gente te regale sus datos que el contratar investigadores y espías.

¿Qué me va a pasar?
Probablemente nada, ¿quién sabe? ¿Realmente te merece la pena aunque no tengas nada que ocultar el ceder tus datos?

Vale, el ejemplo anterior es de paranoico. ¿Qué puedes esperar de alguien al que amenazaron con detenerle bajo terrorismo por hacer una foto de la estación de Liverpool?

Realmente el problema trasciende de la autenticación a la identidad digital, a nadie se le ocurriría decir: - "Toma mi documentación y mi imagen y haz lo que quieras con ello". Y sin embargo es lo que se hace en ciertas redes sociales. Basta un detonante como el ser acusado de un delito morboso para que el linchamiento mediático comience empezando por los perfiles (fotografías y mensajes) ya sean privados o públicos, amparándose las empresas en el derecho a la información, aun cuando esta

sea explícitamente creada, que no recopilada, para generar más audiencia. Dinero, dinero, dinero. ¿Se valora el factor beneficio/riesgo?

A.4. Sobre tuenti

Por ley, la red social tuenti no debería permitir el registro a menores de 14 años.
Por ley, todo ciudadano mayor de 14 años debe poseer su D.N.I.
¿Tan difícil es instaurar un sistema para comprobar la primera restricción aprovechándose de las características de D.N.I. electrónico? Realmente no, pero estarían perdiendo usuarios.

El segundo caso me lleva a la trazabilidad. Para conseguir registrarse hace falta que alguien te haya invitado a participar en dicha red (las invitaciones son limitadas). Esto genera un árbol (grafo) en el que se pueden trazar relaciones sociales, temporales y hasta geográficas. Tiene sus ventajas como la identificación de perfiles maliciosos, pero en absoluto respeta el anonimato. ¿Soy consciente de la metainformación que genero?

A.5. Rumbo

Para concluir el tema, quiero presentar la siguiente noticia
http://www.elperiodicodearagon.com/noticias/noticia.asp?pkid=576131

Promete.
Por eso, desde el Centro Politécnico Superior (CPS) de la Universidad de Zaragoza y el Instituto de Investigación en Ingeniería de Aragón (I3A), a través de la Cátedra Telefónica, se planteó la idea de crear un Observatorio de la Identidad Digital. El objetivo de este observatorio es el de ofrecer asesoramiento tanto público como privado de un nuevo servicio creado por y para la sociedad de la información, como es la identidad digital.

Confunde.
Salazar plantea que "si la identidad digital de una persona es, entre otras, la forma en que se le conoce en internet, cada uno de los perfiles que se tienen en las redes públicas o los personajes de juegos en línea como Second Life o World of Warcraft son una identidad digital". Sin embargo, especifica, "no era ese tipo de identidad el que queríamos tratar".

Por su seguridad.
Otra aplicación importante es la seguridad, para identificar terroristas o criminales.

Creo que no es forma de presentar un tema tan complejo al gran público, se confunden término y se tergiversa tanto el tema que parece que están haciendo lo mismo de siempre. Contraseñas. Sin duda queda mucho para que el concepto de "identidad digital" sea aceptado y comprendido.

A.6. Consideraciones del lenguaje

Soy consciente de que he hecho uso y abuso de extrangerismos, sin embargo es algo inevitable debido al contexto en el que me manejo. Para ayudar en la tarea de comprensión está el glosario.

Otra peculiaridad es en el uso de las siglas y acrónimo. Creo que es una cuestión de estilo el separar las mayúsculas con puntos o no hacerlo. No suelo hacerlo.

Debo reconocer mi error al confundir el género de credencial. Lo uso en la forma masculina y sin embargo es femenino. Como me enteré casi en la finalización de la redacción de esta memoria, es posible que aparezca de forma errónea en gran parte la misma.

credencial.
(De credencia).

1. *adj. Que acredita.*
2. *f. Real orden u otro documento que sirve para que a un empleado se dé posesión de su plaza, sin perjuicio de obtener luego el título correspondiente.*
3. *f. pl. cartas credenciales.*

R.A.E.

Creo que no me refiero a ninguno de los significados cuando uso la palabra como sustantivo, así que el error se extiende a tomarla como **sinónimo de acreditación** por influencia de la voz inglesa *credential*.

acreditación.

1. *f. Acción y efecto de acreditar.*
2. *f. Documento que acredita la condición de una persona y su facultad para desempeñar determinada actividad o cargo.*

R.A.E.

Algo parecido pasa con la equivalencia *library-librería* cuando la traducción más precisa es biblioteca, aunque según la R.A.E. librería y biblioteca sean sinónimos. Cuestiones de estilo.

A.7. The Venn of Identity

Durante el proyecto se han mencionado las tecnologías de SAML, OpenID e Information Cards. El diagrama de Venn de la identidad (ver figura A.1) muestra cómo se combinan todas ellas para permitir la identidad centrada en el usuario. Bajo mi opinión es un fallo de concepto ya que no operan en el mismo nivel. SAML se utiliza para representar identidades digitales (como token) y relaciones de federación (como protocolo), OpenId es un método específico para la web de gestión de identidades virtuales basado en usuario/contraseña pero no centrado en el usuario, e Information Cards es una envoltura para adaptar cualquier tipo de autenticación y paso de atributos a través de Servicios Web y centrado en el usuario.

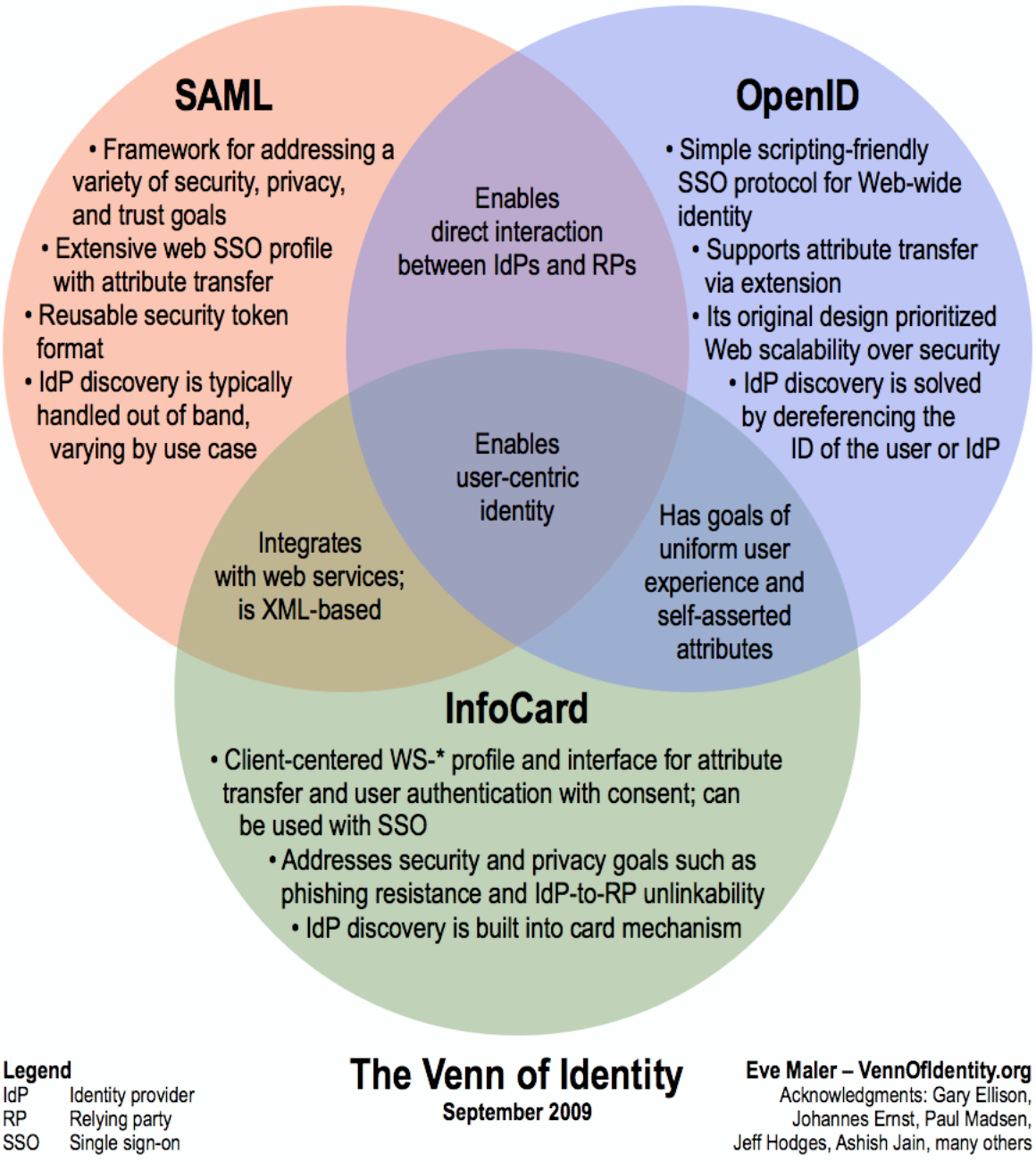

Figura A.1: Diagrama de Venn de la identidad.

A.8. InfoCard 2.0

El 5 de Mayo del 2010 se publicó la noticia de que Microsoft sacaba "AD FS 2.0" (Active Directory Federation Services 2.0).
Ver http://blogs.msdn.com/card/archive/2010/05/05/ad-fs-2-0-is-here.aspx.

Active Directory es el servicio de directorio de Microsoft. Se supone que esta nueva versión era la oportunidad para que Windows tuviera un servidor oficial que ejerciera el rol de proveedor de identidad en el contexto de las Information Cards. El problema es que parece que han roto la retrocompatibilidad.

El descontento de algunos usuarios no se ha hecho esperar. Infocards es una tecnología sobre la que se tienen grandes expectativas. ¿Despegará algún día?

...

On the other hand... We won't know for sure until ADFS 2.0 ships, but from what I and other people have seen from the beta and release candidate versions, Microsoft has broken backward compatibility with CardSpace 1.0. This means that unless Microsoft has taken recent steps to regress their information card issuance code, ADFS 2.0 will ship in information card limbo.

...

But see, people were waiting. Big companies, waiting to run information card pilots. Governments, excited to use ADFS 2.0 to implement higher-assurance consumer identity projects. There weren't a huge number of interested parties, but dammit, they were BIG interested parties. Those interested parties need a sustainable closed circle - a production server and a production client. Not a production server that can only work with a client that "isn't done yet".

...

[Adventures of an Eternal Optimist, 2010]

Apéndice B

Servidor RADIUS

B.1. radiusd.conf

```
1
3   prefix = /usr
    exec_prefix = /usr
5   sysconfdir = /etc
    localstatedir = /var
7   sbindir = ${exec_prefix}/sbin
    logdir = /var/log/freeradius
9   raddbdir = /home/infocard/freeradius-conf
    radacctdir = ${logdir}/radacct
11
    #  Location of config and logfiles.
13  confdir = ${raddbdir}
    run_dir = ${localstatedir}/run/freeradius
15
    # Should likely be ${localstatedir}/lib/radiusd
17  db_dir = $(raddbdir)

19
    libdir = /usr/local/lib
21  pidfile = ${run_dir}/freeradius.pid
    user = freerad
23  group = freerad
    max_request_time = 30
25  cleanup_delay = 5
    max_requests = 1024
27

29  #Authentication
    listen {
31    ipaddr = 192.168.0.23
      port = 1812
33    type = auth
      interface = eth0
35  }

37  #Accounting
    listen {
39    ipaddr = 192.168.0.23
      port = 1813
41    type = acct
```

```
    interface = eth0
}


hostname_lookups = yes
allow_core_dumps = no

regular_expressions = yes
extended_expressions  = yes


log {
    destination = files
    file = ${logdir}/radius.log
    syslog_facility = daemon
    stripped_names = no
    auth = yes
    auth_badpass = yes
    auth_goodpass = yes
}


security {
    max_attributes = 200
    status_server = no
}


proxy_requests  = no
#$INCLUDE ${confdir}/proxy.conf

$INCLUDE ${confdir}/clients.conf

snmp = no
#$INCLUDE  ${confdir}/snmp.conf


thread pool {
    start_servers = 5
    max_servers = 32
    min_spare_servers = 3
    max_requests_per_server = 0
}


modules {
    mschap {
        authtype = MS-CHAP
        use_mppe = yes
        require_encryption = yes
        require_strong = yes
    }

    $INCLUDE eap.conf

    preprocess {
```

```
    huntgroups = ${confdir}/huntgroups
    hints = ${confdir}/hints
    ascend_channels_per_line = 23
    with_ntdomain_hack = no
    with_specialix_jetstream_hack = no
    with_cisco_vsa_hack = no
  }

  files{
    usersfile = ${confdir}/users
    acctusersfile = ${confdir}/acct_users
    preproxy_usersfile = ${confdir}/preproxy_users
    compat = no
  }

  perl {
    #  The Perl script to execute
    module = ${confdir}/perl_IC.pm

    #
    #  The following hashes are given to the module and
    #  filled with value-pairs (Attribute names and values)
    #
    #  %RAD_CHECK            Read-only       Check items
    #  %RAD_REQUEST          Read-only       Attributes from the request
    #  %RAD_REPLY            Read-write      Attributes for the reply
    #
    #  The return codes from functions in the perl_script
    #  are passed directly back to the server.  These
    #  codes are defined in doc/configurable_failover,
    #  src/include/modules.h (RLM_MODULE_REJECT, etc),
    #  and are pre-defined in the 'example.pl' program
    #  which is included.
    #

    #
    #  List of functions in the module to call.
    #  Comment out and change if you want to use other
    #  function names than the defaults.
    #
    func_authenticate = authenticate
    func_authorize = authorize
    func_preacct = preacct
    func_accounting = accounting
    #func_checksimul = checksimul
    #func_pre_proxy = pre_proxy
    #func_post_proxy = post_proxy
    #func_post_auth = post_auth
    #func_xlat = xlat
    #func_detach = detach

    #
    #  Comment out the following line if you whish
    #  to use seperate functions for Start and Stop
    #  accounting packets. In that case, the
    #  func_accounting function is not called.
    #
    #func_start_accounting = accounting_start
    #func_stop_accounting = accounting_stop

    #  Uncomment the following lines if your perl is
    #  compiled with ithreads support.
```

```
        #  the settings bellow are the default one.
167     #
        #max_clones = 32
169     #start_clones = 32
        #min_spare_clones = 0
171     #max_spare_clones = 32
        #cleanup_delay = 5
173     #max_request_per_clone = 0
      }
175   }

177

179   authorize {
        preprocess
181     mschap
        eap
183     perl
      }
185

187

      authenticate {
189     Auth-Type MS-CHAP {
          mschap
191     }
        eap
193   }

195

      preacct {
197     preprocess
      }
199

201

      accounting {
203     perl
      }
205

      #$INCLUDE  ${confdir}/sites-available/inner-tunnel
```

B.2. eap.conf

```
eap {

  default_eap_type = peap

  timer_expire     = 60

  ignore_unknown_eap_types = no

  cisco_accounting_username_bug = no

  mschapv2 {
  }

  ## EAP-TLS
  tls {
    certdir = ${confdir}/certs
    cadir = ${confdir}/certs

    #private_key_password =
    private_key_file = ${certdir}/servidor-radius.key
    certificate_file = ${certdir}/servidor-radius.crt

    CA_file = ${cadir}/CA.crt

    dh_file = ${certdir}/dh

    random_file = /dev/urandom

  }

  peap {.
    default_eap_type = mschapv2
    copy_request_to_tunnel = no
    use_tunneled_reply = no
    proxy_tunneled_request_as_eap = no
    #virtual_server = "inner-tunnel"
  }

}
```

B.3. schema.sql

```
/*
 * Project InfoCard, minimal database
*/


--Where we store user (PERMANENT DATA)
CREATE TABLE USERS (
   name VARCHAR(32),
   cleartext_password VARCHAR(32),
  PRIMARY KEY (name)
);

--Users connected to the system (TEMPORAL DATA)
CREATE TABLE CONNECTED_USERS (
   name VARCHAR(32),
   card_id VARCHAR UNIQUE,
   self_card_id VARCHAR UNIQUE NOT NULL,
   login_time TIMESTAMP DEFAULT NOW(),
   auth_method VARCHAR(32),
  PRIMARY KEY (name),
   FOREIGN KEY (name) REFERENCES USERS (name)
);


--
-- CREATE TABLE ISSUED_CARDS (
--   name VARCHAR(32),
--   credential_card_id VARCHAR,
--   issued_time TIMESTAMP,
--   issued_card_id VARCHAR,
--  PRIMARY KEY (name),
--  FOREIGN KEY (name) REFERENCES USERS (name)
-- );
```

B.4. perl_IC.pm

```
2
use strict;
4
# This is very important ! Without this script will not get the filled  hashesh from
    main.
6 use vars qw(%RAD_REQUEST %RAD_REPLY %RAD_CHECK);
use Data::Dumper;
8

10 ## SAMU
use MIME::Base64;
12 use MIME::Base64::URLSafe;
use LWP::Simple;
14 use Crypt::CBC;
use String::Random qw(random_regex random_string);
16 use Digest::SHA qw(sha256_hex sha256);
use DBI;
18
##------------------------ CONFIG PARAMS
20

22 #----DATABASE CONNECTION PARAMS
  my $DB_name = 'ic_db1';
24   my $DB_username = 'raduser1';
  my $DB_password = 'radpass1';
26
#----ENCRYPTION PARAMS
28   # Encryption $key, 256 bits = 32 bytes, SHA256 of the string $key.
  # If you want NO encryption, only base64 coding for transmission purposes, assign
      this value to undef (no quotes);
30   my $key = 'clave';
  # Where apply for an Infocard-URL
32   my $preurl = 'https://idp.ic/module.php/InfoCard/STS_card_issuer.php';

34

36 # This is hash wich hold original request from radius
#my %RAD_REQUEST;
38 # In this hash you add values that will be returned to NAS.
#my %RAD_REPLY;
40 # This is for check items
# my %RAD_CHECK;
42
#
44 # This the remapping of return values
#
46         use constant    RLM_MODULE_REJECT=>    0;#  /* immediately reject the request
               */
           use constant    RLM_MODULE_FAIL=>      1;#  /* module failed, don't reply */
48         use constant    RLM_MODULE_OK=>        2;#  /* the module is OK, continue */
           use constant    RLM_MODULE_HANDLED=>   3;#  /* the module handled the request,
               so stop. */
50         use constant    RLM_MODULE_INVALID=>   4;#  /* the module considers the request
                invalid. */
           use constant    RLM_MODULE_USERLOCK=>  5;#  /* reject the request (user is
               locked out) */
52         use constant    RLM_MODULE_NOTFOUND=>  6;#  /* user not found */
           use constant    RLM_MODULE_NOOP=>      7;#  /* module succeeded without doing
               anything */
```

```
54          use constant    RLM_MODULE_UPDATED=>    8;#  /* OK (pairs modified) */
            use constant    RLM_MODULE_NUMCODES=>   9;#  /* How many return codes there are
                */
56


58
    ## Request an InfoCard
60  # Input: username, cardid
    # Output: InfoCard-URL
62  sub get_InfoCard {
      # Username/cardid
64    my $username = $_[0];
      my $cardid = $_[1];
66
      my $iv = ''; #Do NOT change it, initialization
68    my ($cipher, $encrequest, $response);

70    if (defined($key)){
        my $enckey = pack("H*",sha256_hex($key));
72      # IV, 16 bytes
        $iv = random_regex('.' x 16);
74      # Initialize cipher
        $cipher = Crypt::CBC->new( -literal_key=>1, -key => $enckey, -cipher => 'Rijndael
            ', -iv => $iv, -header=>'none');
76    }

78    #Encapsulate data (username size + username + cardid size + cardid)
      #Attribute MUST NOT begin with a number
80
      my $request = length($username).$username.length($cardid).$cardid;
82
      # Encrypt request
84    if (defined($key)){
        $encrequest = $cipher->encrypt($request);
86    } else {
        $encrequest = $request;
88    }

90    # Retrieve data
      my $url = $preurl."?data=".urlsafe_b64encode($encrequest)."&iv=".urlsafe_b64encode(
          $iv)."&ident=RADIUS";
92    my $encresponse = decode_base64(get $url);

94    # Decrypt data
      if (defined($key)){
96      $response = $cipher->decrypt($encresponse);
      } else {
98      $response = $encresponse;
      }
100   &radiusd::radlog(1, "URL: $response");
      return $response;
102 }

104


106 # Input: EAP-message
    # Output: Card Id or NULL
108 ## pending improvements PARSE the eap-message in a better way
    sub get_Card_Id {
110   my ($pos, $len, $vendor, $type, $data_length, $data);
      if (uc($RAD_REQUEST{'EAP-Message'})=~/(.{3}7.{4}00534D48.*)$/) {
112
```

```
        $pos = 0;
        $len = 0;

        #TLV_type (2 bytes)
        $len = 4;
        $type = substr($1, $pos, $len);
        $pos += $len;

        #Data Length (2 bytes)
        $len = 4;
        $data_length = hex("0x".substr($1, $pos, $len))*2; #*2 because we are in hex mode
            (2 hex chars = 1 byte)
        $pos += $len;

        #Vendor id (4 bytes)
        $len = 8;
        $vendor = substr($1, $pos, $len);
        $pos += $len;

        #Data
        $len = $data_length;
        $data = pack("H*",substr($1, $pos, $len));

        return $data;
      }

      return "";
    }


    ##MODEL DATA FUNCTIONS

    # INPUT: $RAD_REQUEST('User-Name')
    # OUTPUT: Password, taken from a database, ldap, etc
    sub get_Password{
      # Username
      my $userID = $_[0];
      my $retval;
      my $dbh = DBI->connect("dbi:Pg:dbname=$DB_name", $DB_username, $DB_password) or die
          "ERROR: could not connecto to database";
      my $sth = $dbh->prepare('SELECT cleartext_password FROM users WHERE name = ?') or
          die "Couldn't prepare statement: " . $dbh->errstr;
      $sth->execute($userID) or die "Couldn't execute statement: " . $sth->errstr;
      my @data = $sth->fetchrow_array();
      if (@data==0){
        $retval = ''; #NULL
      } else {
        $retval = $data[0];
      }
      $sth->finish;
      $dbh->disconnect();
      return $retval;
    }


    # INPUT: $RAD_REQUEST('User-Name'), Self issued card ID, authentication-method
    # OUTPUT: Set the user connected in the database
    sub connect_user{
      my $userID = $_[0];
      my $self_card_id = $_[1];
      my $auth_method = $_[2];
```

```
172
    &disconnect_user($userID);
174
    my $dbh = DBI->connect("dbi:Pg:dbname=$DB_name", $DB_username, $DB_password) or die
        "ERROR: could not connecto to database";
176 my $sth = $dbh->prepare('INSERT INTO connected_users (name, self_card_id, auth_
        method) VALUES (?,?,?)') or die "Couldn't prepare statement: " . $dbh->errstr;
    $sth->execute($userID, $self_card_id, $auth_method) or die "Couldn't execute
        statement: " . $sth->errstr;
178 $sth->finish;
    $dbh->disconnect();
180 return 0;
  }
182

184 # INPUT: $RAD_REQUEST('User-Name')
  # OUTPUT: Set the user disconnected in the database
186 sub disconnect_user{
    my $userID = $_[0];
188
    my $dbh = DBI->connect("dbi:Pg:dbname=$DB_name", $DB_username, $DB_password) or die
        "ERROR: could not connecto to database";
190 my $sth = $dbh->prepare('DELETE FROM connected_users where name = ?') or die "Couldn
        't prepare statement: " . $dbh->errstr;
    $sth->execute($userID) or die "Couldn't execute statement: " . $sth->errstr;
192 $sth->finish;
    $dbh->disconnect();
194
    return 0;
196 }

198


200
  #RADIUS HARD STUFF
202 #In this Hash I'll store user status
  my %user_status;
204

206 # Function to handle authorize
  sub authorize {
208   &radiusd::radlog(1, "---------->    AUTORIZACION");

210 # for (keys %RAD_REQUEST) {&radiusd::radlog(1, "RAD_REQUEST: $_ = $RAD_REQUEST{$_}");}
  # for (keys %RAD_REPLY) {&radiusd::radlog(1, "RAD_REPLY: $_ = $RAD_REPLY{$_}");}
212 # for (keys %RAD_CHECK) {&radiusd::radlog(1, "RAD_CHECK: $_ = $RAD_CHECK{$_}");}
  for (keys %user_status) {&radiusd::radlog(1, "user_status: $_ = $user_status{$_}");}
214

216   if (($RAD_REQUEST{'EAP-Type'} eq "MS-CHAP-V2") && (!exists($user_status{$RAD_REQUEST
        {'User-Name'}}))){
      $user_status{$RAD_REQUEST{'User-Name'}} = 1;
218   $RAD_CHECK{'Cleartext-Password'} = &get_Password($RAD_REQUEST{'User-Name'});
      return RLM_MODULE_UPDATED;
220   } else {
      if (($user_status{$RAD_REQUEST{'User-Name'}}==1) && ($RAD_REQUEST{'EAP-Type'} eq "
          PEAP")){
222     $user_status{$RAD_REQUEST{'User-Name'}} = 2;
        return RLM_MODULE_UPDATED;
224   } elsif (($user_status{$RAD_REQUEST{'User-Name'}}==2) && ($RAD_REQUEST{'EAP-Type'}
          eq "MS-CHAP-V2")){
        my ($cardid) = &get_Card_Id;
```

```
226         if ($cardid ne ""){
              &connect_user($RAD_REQUEST{'User-Name'},$cardid,$RAD_REQUEST{'EAP-Type'});
228           &radiusd::radlog(1, "---------->    CARDID $cardid");
              #Request an Infocard for the user to the STS
230           $RAD_REPLY{'EDUROAM-Card-URL'} = &get_InfoCard($RAD_REQUEST{'User-Name'},
                  $cardid);
            }
232
            return RLM_MODULE_UPDATED;
234       } else {
            delete($user_status{$RAD_REQUEST{'User-Name'}});
236       }
        }
238     return RLM_MODULE_NOOP;

240

242 #     if ($RAD_REQUEST{'EAP-Type'} eq "MS-CHAP-V2") {
    #       if(!exists($state{$RAD_REQUEST{'User-Name'}})){
244 #         $state{$RAD_REQUEST{'User-Name'}} = 1;
    #       }
246 #     }
    #
248 #     if ($state{'User-Name'}==0){
    #       if ($RAD_REQUEST{'EAP-Type'} eq "MS-CHAP-V2") { #Set password for mschap auth
250 #         $state=1;
    #         $RAD_CHECK{'Cleartext-Password'} = &get_Password($RAD_REQUEST{'User-Name'});
252 #         return RLM_MODULE_UPDATED;
    #       }
254 #     } elsif ($state==1){
    #       if ($RAD_REQUEST{'EAP-Type'} eq "PEAP") { #Right way to victory
256 #         $state=2;
    #         return RLM_MODULE_UPDATED;
258 #       } else {
    #         $state=0;
260 #       }
    #     } elsif ($state==2){
262 #       if ($RAD_REQUEST{'EAP-Type'} eq "MS-CHAP-V2") { #Issue card url
    #         $state=3;
264 #         my ($cardid) = &get_Card_Id;
    #         if ($cardid ne ""){
266 #           &connect_user($RAD_REQUEST{'User-Name'},$cardid,$RAD_REQUEST{'EAP-Type'});
    #           &radiusd::radlog(1, "---------->    CARDID $cardid");
268 #           #Request an Infocard for the user to the STS
    #           $RAD_REPLY{'EDUROAM-Card-URL'} = &get_InfoCard($RAD_REQUEST{'User-Name'},
    $cardid);
270 #           return RLM_MODULE_UPDATED;
    #         }
272 #       } else {
    #         $state=0;
274 #       }
    #     } elsif ($state==3){
276 #     }
    #
278 #     return RLM_MODULE_NOOP;
    }
280
    # Function to handle authenticate
282 sub authenticate {
          &radiusd::radlog(1, "---------->    AUTENTICACION");
284       # For debugging purposes only
    #     &log_request_attributes;
```

```
286             return RLM_MODULE_OK;
    }
288
    # Function to handle preacct
290 sub preacct {
            # For debugging purposes only
292 #       &log_request_attributes;
      &radiusd::radlog(1, "---------->    PRE ACCOUNTING");
294
            return RLM_MODULE_OK;
296 }

298 # Function to handle accounting
    sub accounting {
300
      &radiusd::radlog(1, "---------->    ACCOUNTING");
302 for (keys %RAD_REQUEST) {&radiusd::radlog(1, "RAD_REQUEST: $_ = $RAD_REQUEST{$_}");}
    for (keys %RAD_REPLY) {&radiusd::radlog(1, "RAD_REPLY: $_ = $RAD_REPLY{$_}");}
304 for (keys %RAD_CHECK) {&radiusd::radlog(1, "RAD_CHECK: $_ = $RAD_CHECK{$_}");}
      return RLM_MODULE_OK;
306 }

308 # Function to handle checksimul
    sub checksimul {
310         # For debugging purposes only
    #       &log_request_attributes;
312
            return RLM_MODULE_OK;
314 }

316 # Function to handle pre_proxy
    sub pre_proxy {
318         # For debugging purposes only
    #       &log_request_attributes;
320
            return RLM_MODULE_OK;
322 }

324 # Function to handle post_proxy
    sub post_proxy {
326         # For debugging purposes only
    #       &log_request_attributes;
328
            return RLM_MODULE_OK;
330 }

332 # Function to handle post_auth
    sub post_auth {
334         # For debugging purposes only
    #       &log_request_attributes;
336         &radiusd::radlog(1, "---------->    POST-AUTH");
      for (keys %RAD_REQUEST) {
338             &radiusd::radlog(1, "RAD_REQUEST: $_ = $RAD_REQUEST{$_}");
        }
340         return RLM_MODULE_OK;
    }
342
    # Function to handle xlat
344 sub xlat {
        # For debugging purposes only
346 #       &log_request_attributes;
```

```
348         # Loads some external perl and evaluate it
            my ($filename,$a,$b,$c,$d) = @_;
350         &radiusd::radlog(1, "From xlat $filename ");
            &radiusd::radlog(1,"From xlat $a $b $c $d ");
352         local *FH;
            open FH, $filename or die "open '$filename' $!";
354         local($/) = undef;
            my $sub = <FH>;
356         close FH;
            my $eval = qq{ sub handler{ $sub;} };
358         eval $eval;
            eval {main->handler;};
360 }

362 # Function to handle detach
    sub detach {
364         # For debugging purposes only
#           &log_request_attributes;
366
            # Do some logging.
368         &radiusd::radlog(0,"rlm_perl::Detaching. Reloading. Done.");
    }
370
    #
372 # Some functions that can be called from other functions
    #
374
    sub test_call {
376         # Some code goes here
    }
378
    sub log_request_attributes {
380         # This shouldn't be done in production environments!
            # This is only meant for debugging!
382         for (keys %RAD_REQUEST) {
                &radiusd::radlog(1, "RAD_REQUEST: $_ = $RAD_REQUEST{$_}");
384         }
    }
```

Apéndice C

Conector - Guión Perl

C.1. conector.pl

```
#!/usr/bin/perl -w

# AUTHOR: Samuel Muñoz Hidalgo
# CONTACT: samuel.mh@gmail.com
# LAST REVISION: 4 June 2009
# DESCRIPTION: PERL connector for the in Eduroam integrated identity metasystem USSO
    project


use LWP::Simple;


# needed programs: zenity,gksu,iwconfig,ifconfig,...,

#-------------CONFIGURACION----------------------

#---> USER SPECIFIC
$LANG = `echo \$LANG`;

$username = undef;
$password = undef;
$iface = undef;
$driver = undef;
$ssid='INFOCARD';

#---> GENERAL
$DM_path='/usr/bin/digitalme';
$WPA_SUPPLICANT_path = '/home/infocard/wpa_supplicant-0.6.8/wpa_supplicant/./wpa_
    supplicant';
$IFCONFIG_path = '/sbin/ifconfig';
$IWCONFIG_path = '/sbin/iwconfig';


$WPA_SUPPLICANT_CONF_file = '/tmp/WPASup_eduroam.conf';
$IC_file = '/tmp/IC-eduroam.crd';

$DM_pipe = '/tmp/DMpipe';
$WPA_pipe = '/tmp/DMpipe';
$WIFI_DRIVERS = 'hostap hermes madwifi MADWIFI atmel wext ndiswrapper broadcom ipw
    wired  bsd  ndis ';
```

```
39  $CONF_CA_CERT = "-----BEGIN CERTIFICATE-----
    MIIDdTCCAt6gAwIBAgIJANFANyG/ki7+MA0GCSqGSIb3DQEBBQUAMIGEMQswCQYD
41  VQQGEwJFUzEPMA0GA1UECBMGTWFkcmlkMRowGAYDVQQHFBFBbGNhbOEgZGUgSGVu
    YXJlczEMMAoGA1UEChMDVUFIMRMwEQYDVQQLFAphdXRvbeF0aWNhMSUwIwYDVQQD
43  ExxGQUtFIENBIEZPUiBURVNUSU5HIFBVUlBPU0VTMB4XDTA5MDUxNzE3MDA1OFoX
    DTEwMDYyMTE3MDA1OFowgYQxCzAJBgNVBAYTAkVTMQ8wDQYDVQQIEwZNYWRyaWQx
45  GjAYBgNVBAcUEUFsY2Fs4SBkZSBIZW5hcmVzMQwwCgYDVQQKEwNVQUgxEzARBgNV
    BAsUCmF1dG9t4XRpY2ExJTAjBgNVBAMTHEZBS0UgQ0EgRk9SIFRFU1RJTkcgUFVS
47  UE9TRVMwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOBM+ryQe57VEhXeCpfY
    m++b4qwxlwg0emBIw0ocxGVqGovdAtfOvx0WnPySq4LJqvtlhLpn+HfiRINsBMAb
49  vEJU31Bu7eN2QOPdiOECzomaXk09F1goESvaRYK7c3gaxPUBTVGGknX19oYkXesk
    x+BqJze0mxZn8ebFf/iZ9ATpAgMBAAGjgewwgekwHQYDVR0OBBYEFCHR6w3uzhJO
51  qW1YQMCga0AzDAaeMIG5BgNVHSMEgbEwga6AFCHR6w3uzhJOqW1YQMCga0AzDAae
    oYGKpIGHMIGEMQswCQYDVQQGEwJFUzEPMA0GA1UECBMGTWFkcmlkMRowGAYDVQQH
53  FBFBbGNhbOEgZGUgSGVuYXJlczEMMAoGA1UEChMDVUFIMRMwEQYDVQQLFAphdXRv
    beF0aWNhMSUwIwYDVQQDExxGQUtFIENBIEZPUiBURVNUSU5HIFBVUlBPU0VTggkA
55  0UA3Ib+SLv4wDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQUFAAOBgQDc+u0ow99i
    TcV9uAMreI9ROSSZHyreVSb8pkNz/SOODNvfqeIfK+0bCFEXWtMVEEuYbtQ9vqCX
57  eFjtZqNai5vCYzzPeud9G1ri3ONgIk9htZETtwF5pwDRTesDtKLfmoaqHfzmpN6j
    mj3VxaLvCb11QQv+Rgp/G6RfDl4f4Zs79Q==
59  -----END CERTIFICATE-----";

61
    #-------------DICTIONARY -----------------------
63
    if ($LANG =~ /ES/i){ #SPANISH (Castellano)
65    $D_W_TITLE = '"Conector InfoCard para Eduroam"';
    }else{               #DEFAULT ENGLISH
67    $D_W_TITLE = '"Eduroam InfoCard Connector"';
    }
69

71
    #------------- FUNCIONES ----------------------
73  sub checkcert {
      if ($CA_cert eq $CONF_CA_CERT){
75      return 1;
      } else {
77      return undef;
      }
79  }

81

83  #-------------- Extraer el CardId de una tarjeta autoemitida ----------------------

85  system "mknod  $DM_pipe p";

87  system 'zenity --info --title='.$D_W_TITLE.' --text="Por favor, seleccione una
        InfoCard autoemitida."';
    $got_cert = undef;
89  while(!$got_cert){
      defined ($DM_pid = fork) or die "ERROR: Cannot fork: $!";
91    if (!$DM_pid){
        #hijo
93      open STDERR, ">>$DM_pipe" or die "ERROR: Cannot redirect";
        exec ("$DM_path", "--loglevel=debug") or die ;
95    }

97    open LOG, "$DM_pipe";

99    #Verificar que tiene cargado  el certificado de una CA que autentique a mi STS
```

```
  $status = 0;
  $CA_cert = '';
  while ($status<2) {
    while (!defined($_=<LOG>)) {}
    if (/Importing .* card/){ #Fin de los certificados
      kill (9, $DM_pid);
      close LOG;
      system "rm $DM_pipe";
      $q_cert = system 'zenity --question --title='.$D_W_TITLE.' --text="ERROR: debes
          importar el certificado de la Autoridad de certificación en el DigitalMe.\n
          Â¿Quieres que lo haga por ti?"';
      if($q_cert) { #NADA
        die "ERROR: DigitalMe could not found your STS certificate"
      } else {
        #system 'zenity --info --title="Conector InfoCard para Eduroam" --text="
            Importando el certificado en el ca-bundle"';
        system "gksu -D $D_W_TITLE \"echo '$CONF_CA_CERT'>>/usr/share/digitalme/certs/
            ca-bundle.crt\"";
        $status = 3;
      }
    }elsif ($status==0) {
      if(/(-----BEGIN CERTIFICATE-----)/) {
        $CA_cert = "$1\n";
        $status = 1;
      }
    } elsif ($status==1){
      if(/(-----END CERTIFICATE-----)/) {
        $CA_cert .= $1;
        if (&checkcert){ #es el certificado
          $status = 2;
          $got_cert = 1;
        } else {
          $status = 0;
        }
      } else {
        $CA_cert .= $_;
      }
    }
  }
}


sleep 1;

#Buscar el ic:CardId de una Infocard autoemitida
$status = 0;
$card_ID = '';
while ($status!=2) {
  while (!defined($_=<LOG>)) {}
  chomp;
  if ($status==0) {
    if(/Importing unmanaged token card .../) {
      $status = 1;
    }
  } elsif ($status==1){
    if(/Private Id \(seed = .*obj = (urn:uuid:\w{8}-(\w{4}-){3}\w{12})\).*/) {
      $status = 2;
      $card_ID = $1;
    } elsif (/(Importing managed token card ...)|(Certificate verification callback)/)
        {
      $status = 0;
    }
```

```
157       }
      }
159
161   kill (9, $DM_pid);
      close LOG;
163   system "rm $DM_pipe";

165

167   #Credenciales de autenticación
      while(!defined($username)){chomp($username = `zenity --entry --title=$D_W_TITLE --text
          ="Introduzca su nombre de usuario."`)};
169   while(!defined($password)){chomp($password = `zenity --entry --title=$D_W_TITLE --hide
          -text --text="Introduzca su contraseña."`)};

171   #Interfaz de conexión, sacado de iwconfig
      while(!defined($iface)){
173     @ifaces_tmp = `gksu $IWCONFIG_path 2>&1`;
        foreach (@ifaces_tmp){
175       if (/^(\w+).*IEEE 802.11 .*/){
            push @ifaces, $1;
177       }
        }
179
        if (@ifaces>1){
181     while(!defined($iface)){chomp($iface = `zenity --title=$D_W_TITLE --list --text="Por
            favor, seleccione su tarjeta WiFi" --column=Interfaz  @ifaces`)};
        } elsif (@ifaces == 1) {
183       $iface = $ifaces[0];
        } else {
185       die "Not Wifi interface found."
        }
187   }

189   #Driver para el suplicante
      while(!defined($driver)){chomp($driver = `zenity --title=$D_W_TITLE --list --text="Por
          favor, seleccione su controlador de tarjeta WiFi" --column=driver  $WIFI_DRIVERS
          `)};
191
      #
193   print "Usuario:\t$username\nPassword:\t$password\nCard ID:\t$card_ID\nDriver: \
          t$driver\nInterfaz:\t$iface\n";

195
      # print "Usuario:\t$username\nDriver: \t$driver\nInterfaz:\t$iface\n";
197


199


201   #------------ LLAMADA AL SUPLICANTE ----------------------
      #Archivo de configuración
203   $WPA_SUPPLICANT_CONF = "
      eapol_version=1
205   ap_scan=1
      fast_reauth=1
207   card_ID=$card_ID

209   network={
        ssid=\"$ssid\"
211     scan_ssid=1
        key_mgmt=WPA-EAP
```

```
213       pairwise=CCMP TKIP
          group=CCMP TKIP
215       eap=PEAP
          phase1=\"peaplabel=0\"
217       phase2=\"auth=MSCHAPV2\"
          identity=\"$username\"
219       password=\"$password\"
      #     ca_cert=\"/etc/apache2/ssl/CA.crt\"
221     }
        ";
223
225     system "mknod  $WPA_pipe p";

227     defined ($WPA_pid = fork) or die "ERROR: Cannot fork: $!";

229     if (!$WPA_pid){
            #hijo
231       open STDERR, ">>$WPA_pipe" or die "ERROR: Cannot redirect";
          system "echo '$WPA_SUPPLICANT_CONF' > $WPA_SUPPLICANT_CONF_file ";
233       print "Lanzando suplicante\n";
          exec "gksu \"$IFCONFIG_path $iface up ; $WPA_SUPPLICANT_path -dd -K -D$driver -
              i$iface -c$WPA_SUPPLICANT_CONF_file\"";
235     }

237

239
        #------------- CARGAR TARJETA GESTIONADA ----------------------
241     #Sacar URL de descarga de la InfoCard

243
        $status = 0;
245     $IC_URL = '';
        open WPALOG, $WPA_pipe;
247

249       print 'Parseando tarjeta';

251     while ($status<3) {
          while (!defined($_=<WPALOG>)) {}
253       if(/EAP-TLV: Vendor Specific/) {
            $status = 1;
255       }elsif ($status==1){
            if(/VENDOR ID: 0534D48/) {
257           $status = 2;
            }
259       }elsif ($status==2){
            if(/DATA: (http.*data=.*iv=.*$')/){
261           $status = 3;
              $IC_URL = $1;
263           $IC = '';
              system "sleep 4";
265           $IC = get($IC_URL);
              if ($IC eq '') {
267             print 'ERROR: Could not get your InfoCard.'
              } else {
269             open(IC_FILE_HANDLE, ">$IC_file") or die 'cannot create IC file!';
                print IC_FILE_HANDLE $IC;
271             close (IC_FILE_HANDLE);
                system "$DM_path $IC_file";
273             unlink $IC_file;
```

```
        }
275       }
    }
277 }

279 close WPALOG;

281 system "rm $WPA_SUPPLICANT_CONF_file";
    system "rm $WPA_pipe";
```

Apéndice D

Ejemplos del manual de despliegue

D.1. Generación de certificados

```
#! /bin/sh

# AUTHOR: Samuel Muñoz Hidalgo
# CONTACT: samuel.mh@gmail.com
# DESCRIPTION: Generates a Certification Authority certificate (root) and three more
#   signed by this one (trusted chain)
# LAST REVISION: 5-Jun-2009


CA='CA'

clear
echo "----- CA Certificate"
openssl req -x509 -new -out $CA.crt -keyout $CA.pem -days 400 -passout pass:clave
openssl rsa -in $CA.pem -out $CA.key -passin pass:clave
rm $CA.pem


clear
SUB='sp'
echo "----- Signed by the CA certificates ($SUB)"
openssl req -new -out $SUB.csr -keyout $SUB.pem -passout pass:clave
openssl rsa  -in $SUB.pem -out $SUB.key  -passin pass:clave
rm $SUB.pem
openssl x509 -in $SUB.csr -out $SUB.crt -req -CAkey $CA.key -CA $CA.crt -days 180 -CAcreateserial
rm $SUB.csr

clear
SUB='idp'
echo "----- Signed by the CA certificates ($SUB)"
openssl req -new -out $SUB.csr -keyout $SUB.pem -passout pass:clave
openssl rsa  -in $SUB.pem -out $SUB.key  -passin pass:clave
```

```
33  rm $SUB.pem
    openssl x509 -in $SUB.csr -out $SUB.crt -req -CAkey $CA.key -CA $CA.crt -days
        180 -CAcreateserial
35  rm $SUB.csr

37  SUB='sts'
    echo "----- Signed by the CA certificates ($SUB)"
39  openssl req -new -out $SUB.csr -keyout $SUB.pem -passout pass:clave
    openssl rsa  -in $SUB.pem -out $SUB.key  -passin pass:clave
41  rm $SUB.pem
    openssl x509 -in $SUB.csr -out $SUB.crt -req -CAkey $CA.key -CA $CA.crt -days
        180 -CAcreateserial
43  rm $SUB.csr

45  SUB='radius'
    echo "----- Signed by the CA certificates ($SUB)"
47  openssl req -new -out $SUB.csr -keyout $SUB.pem -passout pass:clave
    openssl rsa  -in $SUB.pem -out $SUB.key  -passin pass:clave
49  rm $SUB.pem
    openssl x509 -in $SUB.csr -out $SUB.crt -req -CAkey $CA.key -CA $CA.crt -days
        180 -CAcreateserial
51  rm $SUB.csr
```

D.2. Virtual hosts de Apache2

D.2.1. /etc/apache2/sites-available/idp

```
<VirtualHost idp.ic:443>
DocumentRoot /home/infocard/simplesaml/www
ServerAdmin webmaster@localhost

  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
  <Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
    # This directive allows us to have apache2's default start page
                # in /apache2-default/, but still have / go to the right
                    place
                RedirectMatch ^/$ /apache2-default/
  </Directory>


  ErrorLog /var/log/apache2/error.log

  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  LogLevel debug

  CustomLog /var/log/apache2/access.log combined
  ServerSignature On


# SSL CONFIGURATION
  SSLEngine on
  SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+
      eNULL
  SSLCertificateFile /etc/apache2/ssl/idp.crt
  SSLCertificateKeyFile /etc/apache2/ssl/idp.key
  SSLCertificateChainFile /etc/apache2/ssl/CA.crt

</VirtualHost>
```

D.2.2. /etc/apache2/sites-available/sp

```
<VirtualHost sp.ic:80>
DocumentRoot /home/infocard/simplesaml/www
DirectoryIndex index.php

  ErrorLog /var/log/apache2/error.log

  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  LogLevel warn

  CustomLog /var/log/apache2/access.log combined
  ServerSignature On

</VirtualHost>
```

D.2.3. /etc/apache2/sites-available/sts

```
<VirtualHost sts.ic:443>
DocumentRoot /home/infocard/simplesaml/www

  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
  <Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
    # This directive allows us to have apache2's default start page
                # in /apache2-default/, but still have / go to the right
                    place
                RedirectMatch ^/$ /apache2-default/
  </Directory>


  ErrorLog /var/log/apache2/error.log

  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  LogLevel debug

  CustomLog /var/log/apache2/access.log combined
  ServerSignature On


#   SSL CONFIGURATION
SSLEngine on
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+
    eNULL

SSLCertificateFile /etc/apache2/ssl/sts.crt
SSLCertificateKeyFile /etc/apache2/ssl/sts.key
SSLCertificateChainFile /etc/apache2/ssl/CA.crt


</VirtualHost>
```

D.3. Carpeta 'metatada'del simpleSAMLphp

D.3.1. simplesaml/metadata/saml20-idp-hosted.php

```
<?php
/*
 * SAML 2.0 Meta data for simpleSAMLphp
 *
 * The SAML 2.0 IdP Hosted config is used by the SAML 2.0 IdP to identify itself.
 *
 * Required parameters:
 *   - host
 *   - privatekey
 *   - certificate
 *   - auth
 *   - authority
 *
 * Optional Parameters:
 *   - 'userid.attribute'
 *
 *
 * Request signing (optional paramters)
 *    When request.signing is true the privatekey and certificate of the SP
 *    will be used to sign/verify all messages received/sent with the HTTPRedirect
    binding.
 *    The certificate and privatekey from above will be used for signing and
 *    verification purposes.
 *
 *   - request.signing
 *
 */


$metadata = array(
  /*
   *Infocard
   */
  'idp.aut.uah.es' => array(
    'host'        =>    'idp.ic',
    'privatekey'    =>  'server.pem',
    'certificate'   =>  'server.crt',
    'auth'        =>    'InfoCard'
  )
);
?>
```

D.3.2. simplesaml/metadata/saml20-idp-remote.php

```
<?php
/*
 * SAML 2.0 Meta data for simpleSAMLphp
 *
 * The SAML 2.0 IdP Remote config is used by the SAML 2.0 SP to identify trusted SAML
    2.0 IdPs.
 *
 */

$metadata = array(
  /*
   * InfoCard
   */
  'idp.aut.uah.es' =>  array(
    'name'                  => 'idp.ic',
    'description'           => 'Proveedor de identidad',
    'SingleSignOnService'   => 'https://idp.ic/saml2/idp/SSOService.php',
    'SingleLogoutService'   => 'https://idp.ic/saml2/idp/SingleLogoutService.php',
    'certFingerprint'       => 'afe71c28ef740bc87425be13a2263d37971da1f9' //¿Qué es
        esto?
  )
);
?>
```

D.3.3. simplesaml/metadata/saml20-sp-hosted.php

```
<?php
/*
 * SAML 2.0 Meta data for simpleSAMLphp
 *
 * The SAML 2.0 IdP Remote config is used by the SAML 2.0 SP to identify itself.
 *
 * Required fields:
 *  - host
 *
 * Optional fields:
 *  - NameIDFormat
 *  - ForceAuthn
 *
 * Authentication request signing
 *    When request.signing is true the privatekey and certificate of the SP
 *    will be used to sign/verify all messages received/sent with the HTTPRedirect
     binding.
 *    Certificate and privatekey must be placed in the cert directory.
 *    All these attributes are optional:
 *
 *  - 'request.signing' => true,
 *  - 'privatekey' => 'server.pem',
 *  - 'certificate' => 'server.crt',
 */

$metadata = array(
  //Hosted SP
  'sp.ic' => array(
    'host'  => 'sp.ic'
  )
);

?>
```

D.3.4. simplesaml/metadata/saml20-sp-remote.php

```
<?php
/*
 * SAML 2.0 Meta data for simpleSAMLphp
 *
 * The SAML 2.0 SP Remote config is used by the SAML 2.0 IdP to identify trusted SAML
     2.0 SPs.
 *
 * Required parameters:
 *   - AssertionConsumerService
 *   - SingleLogoutService
 *
 * Optional parameters:
 *
 *   - simplesaml.attributes (Will you send an attributestatement [true/false])
 *   - NameIDFormat
 *   - ForceAuthn (default: "false")
 *   - simplesaml.nameidattribute (only needed when you are using NameID format email.
 *
 *   - 'base64attributes' =>  false,
 *   - 'simplesaml.attributes'  =>  true,
 *   - 'attributemap'   =>  'test',
 *   - 'attributes'     =>  array('mail'),
 *   - 'userid.attribute'
 *
 * Request signing
 *    When request.signing is true the certificate of the sp
 *    will be used to verify all messages received with the HTTPRedirect binding.
 *    The certificate from the SP must be installed in the cert directory
 *    before verification can be done.
 *
 *   'request.signing' => false,
 *   'certificate' => "saml2sp.example.org.crt"
 *
 */

$metadata = array(
  'sp.ic' => array(
    'AssertionConsumerService' => 'http://sp.ic/saml2/sp/AssertionConsumerService.php'
        ,
    'SingleLogoutService'      => 'http://sp.ic/saml2/sp/SingleLogoutService.php',
    'name' => 'sp.ic',
    'description' => 'Service Provider'
  )
);
?>
```

D.4. pg_hba.conf

```
# PostgreSQL Client Authentication Configuration File
# ===================================================

# Proyecto InfoCard

# TYPE  DATABASE    USER        CIDR-ADDRESS    METHOD

local   all         all                         trust
host    all         all         127.0.0.1/32    trust
host    IC_db1      raduser1    192.168.0.1/32  password
host    IC_db1      sspuser1    192.168.0.1/32  password
```

D.5. Guión de carga de la base de datos

```
#! /bin/sh -e
# AUTHOR: Samuel Muñoz Hidalgo
# CONTACT: samuel.mh@gmail.com
# DESCRIPTION: Automatic database creation
# LAST REVISION: 17-May-2009

DATABASE='ic_db1'

#Used for connections
SUPERUSER='postgres'

#Radius configuration
RADIUS_USER='raduser1'
RADIUS_PASSWORD='radpass1'
RADIUS_PRIVILEGES="
--  Read credentials
GRANT SELECT ON USERS TO $RADIUS_USER;
-- The server can write to the accounting table.
GRANT ALL on CONNECTED_USERS TO $RADIUS_USER;"

#simpleSAMLphp configuration
SSP_USER='sspuser1'
SSP_PASSWORD='ssppass1'
SSP_PRIVILEGES="
-- The server can update to the accounting table.
GRANT UPDATE on CONNECTED_USERS TO $SSP_USER;"


#COMMANDS
COMMAND="psql -U $SUPERUSER  -d $DATABASE"


#CREATE DATABASE & TABLES
psql -U $SUPERUSER -c "create database $DATABASE"
$COMMAND -f schema.sql


#USERS
$COMMAND -c "CREATE USER $RADIUS_USER WITH NOCREATEDB NOCREATEUSER ENCRYPTED
    PASSWORD '$RADIUS_PASSWORD';"
$COMMAND -c "$RADIUS_PRIVILEGES"
$COMMAND -c "CREATE USER $SSP_USER WITH NOCREATEDB NOCREATEUSER ENCRYPTED
    PASSWORD '$SSP_PASSWORD';"
$COMMAND -c "$SSP_PRIVILEGES"

#INSERTING DATA
$COMMAND -c "INSERT INTO USERS (name, cleartext_password) VALUES ('samu','
    clavesamu');"
$COMMAND -c "INSERT INTO USERS (name, cleartext_password) VALUES ('enrique','
    claveenrique');"
$COMMAND -c "INSERT INTO USERS (name, cleartext_password) VALUES ('user','
    claveuser');"
```

Glosario

AAA: Authentication, Authorization and Accounting. Autenticación, autorización y contabilidad. Funciones para controlar una red.

AAA/H: servidor con capacidad AAA que gestiona la identidad del usuario.

AVP: Attribute Value Pair. Par valor-atributo. Forma de transmitir valores en el protocolo RADIUS.

base64: forma de representar la información únicamente con caracteres ASCII imprimibles.

capa: nivel de abstracción en una arquitectura que ofrece una serie de funcionalidades bien definidas.

IP: Identity Provider. Rol en el metasistema encargado de expedir información sobre identidades digitales.

Idp: IP en el contexto de simplesamlphp.

log: registro o fichero en el que se escriben los mensajes de depuración, avisos, o ejecución de un programa.

marketing: mercadotecnia.

metalenguaje: lenguaje que se usa para hablar del lenguaje (RAE).

metasistema: sistema de control encargado de dar servicio a otros sistemas.

NAS: Network Access Server (punto de acceso, switch). Actúan de pasarela para el método de autenticación.

NAI: Network Access Identifier [RFC4282], Identificador de Acceso a la Red. Consta del identificador de usuario y el dominio que gestiona su identidad o realm.

PPID: Private Personal Identifier, identificador unidireccional que relaciona una information card con una RP.

punto de acceso: dispositivo al que se conecta un terminal cuando quiere acceder a una red inalámbrica.

realm: parte opcional de un NAI que indica a qué dominio enrutar una transacción AAA.

roaming: itinerancia. Capacidad de un disposivo para estar conectado a una red sea cual sea su punto de acceso.

RP: Relying Party. Servicio o entidad que acepta infocards para autenticar a sus usuarios.

script: guión, programa interpetable escrito en texto plano.

software: programa o pieza lógica que ejerce una función.

suplicante: programa para conectarse a una red que use el protocolo IEEE 802.1X.

SSO Single Sign On. Inicio de sesión único en un dominio de aplicaciones web.

TLV: Tag Length Value o Type Length Value [Microsoft MS-PEAP]. Forma de definir datos con el esquema tipo de dato o etiqueta, longitud del valor y el propio valor.

token: testigo, algo que se posee y que prueba algo.

URL: Uniform Resource Locator. A efectos prácticos, la dirección de una página web.

VSA: Vendor-Specific Attribute.

Bibliografía

An implementer's guide to the identity selector interoperability profile v1.5. Technical report, Microsoft Corporation, July 2008. URL http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5_Guide.pdf.

RFC 2759 - Microsoft PPP CHAP Extensions, Version 2, January 2000a. URL http://www.rfc-editor.org/rfc/rfc2759.txt.

RFC 2818 - HTTP Over TLS, May 2000b. URL http://www.rfc-editor.org/rfc/rfc2818.txt.

RFC 2865 - Remote Authentication Dial In User Service (RADIUS), June 2000c. URL http://www.rfc-editor.org/rfc/rfc2865.txt.

RFC 3579 - RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP), September 2003a. URL http://www.rfc-editor.org/rfc/rfc3579.txt.

RFC 3580 - IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines, September 2003b. URL http://www.rfc-editor.org/rfc/rfc3580.txt.

RFC 3748 - Extensible Authentication Protocol (EAP), June 2004. URL http://www.rfc-editor.org/rfc/rfc3748.txt.

RFC 5216 - The EAP-TLS Authentication Protocol, March 2008a. URL http://www.rfc-editor.org/rfc/rfc5216.txt.

RFC 5281 - Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0), August 2008b. URL http://www.rfc-editor.org/rfc/rfc5281.txt.

Postgresql 8.3.6 documentation, 2008.

Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, and Jakub Vrana. *PHP Manual*. the PHP Documentation Group, December 2007. URL http://www.php.net/docs.php.

Adventures of an Eternal Optimist. Cardspace *or* adfs 2.0, May 2010. URL http://eternallyoptimistic.com/2010/05/03/cardspace-or-adfs-2-0/.

Azigo. Welcome to azigo! URL http://www.azigo.com/.

Keith Ballinger, Don Box, Francisco Curbera, Steve Graham, Canyang Kevin Liu, Brad Lovering, Antony Nadalin, Mark Nottingham, David Orchad, Claus von Riegen, Jeffrey Schimmer, John Shewchuk, Greg Truty, and Sanjiva Weerawarana. *Web Services Metadata Exchange (WS-MetadataExchange)*, February 2004.

Omer Bar-or and Benjamin Thomas. How do i log in with openid? URL http://openid.net/get-an-openid/start-using-your-openid/.

Vittorio Bertocci, Garret Serack, and Caleb Baker. *Understanding Windows CardSpace: An Introduction to the Concepts and Challenges of Digital Identities*. Independent technology guides. 2008.

Kim Cameron. The laws of identity, 2005. URL http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf.

Kim Cameron and Michael B. Jones. Design rationale behind the identity metasystem architecture, January 2006. URL http://research.microsoft.com/~mbj/papers/Identity_Metasystem_Design_Rationale.pdf.

Dr Clarie Davies and Matt Shreeve. Federated access management: institutional business case toolkit, July 2007. URL http://www.jisc.ac.uk/media/documents/themes/accessmanagement/cc297d001-1.0%20business%20case%20toolkit.pdf.

Toni de la Fuente Díaz. Wpa+eap-tls+freeradius. URL http://kaslab.net/Telematicas_2005/Telematicas_2005-WPA_EAP_TLS_FreeRadius-toni-transparencias.pdf.

Enrique de la Hoz, Samuel Muñoz, Diego R. López, and Antonio García. An infocard-based proposal for unified single sing on, 2009. URL http://tnc2009.terena.org/schedule/presentations/show9a7d.html?pres_id=44.

Erik Dobbelsteijn. Inventory of 802.1x-based solutions for inter-nrens roaming. URL http://www.terena.nl/activities/tf-mobility/deliverables/delD/DelD_v1.2-f.pdf.

eduGAIN. edugain information. URL http://www.edugain.org/info/#info.

eduroam. Eduroam official site, a. URL http://www.eduroam.org/.

eduroam. Deliverable ds5.1.1: eduroam service definition and implementation plan, b. URL http://www.eduroam.org/downloads/docs/GN2-07-327v2-DS5_1_1-_eduroam_Service_Definition.pdf.

eduroam. Warning: Do not use web logins for eduroam, c. URL http://www.eduroam.org/downloads/docs/advisory/eduroamOT-user-advisory-001.pdf.

Electronic Privacy Information Center E.P.I.C. Privacy and consumer profiling. URL http://epic.org/privacy/profiling/.

Feide. simplesamlphp (página principal). URL http://rnd.feide.no/simplesamlphp/.

Feide-UNINETT. simplesamlphp, 2008. URL http://rnd.feide.no/simplesamlphp.

L. Florio and K. Wierenga. Eduroam, providing mobility for roaming users. Technical report, EUNIS, June 2005. URL http://www.terena.org/activities/tf-mobility/docs/ppt/eunis-eduroamfinal-LF.pdf.

The Apache Software Foundation, editor. *Apache HTTP Server Version 2.2 Documentation*. 2009. URL http://httpd.apache.org/docs/2.2/en/.

Google. Federated login for google account users. URL http://code.google.com/intl/es-ES/apis/accounts/docs/OpenID.html.

google.dirson.com. Google ya es un proveedor de identificación openid. URL http://google.dirson.com/post/4165-google-openid-proveedor-identificacion/.

Jonathan Hassell. *RADIUS*. O'REILLY, October 2002. ISBN 0-596-00322-6.

Samuel Muñoz Hidalgo and Enrique de la Hoz de la Hoz. Integrated in-eduroam infocard identity metasystem for s.s.o., 2009. URL http://it.aut.uah.es/enrique/research/demo.html.

Higgins Developers List. [higgins-dev] use case for self issued token as user credential. URL http://dev.eclipse.org/mhonarc/lists/higgins-dev/msg01330.html.

Internet Identity Workshop 9. Iiwix - book of proceedings, November 2009. URL http://iiw.idcommons.net/File:IIW9Notes.pdf.

Internet2. About. URL http://shibboleth.internet2.edu/about.html.

Michael B. Jones. A guide to using the identity selector interoperability profile v1.5 within web applications and browsers. Technical report, Microsoft Corporation, July 2008. URL http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5_Web_Guide.pdf.

Daniel Koren. Peap & eap-ttls, March 2003. URL www.cs.huji.ac.il/~sans/students_lectures/PEAP-TTLS.ppt.

Linux. 802.1x port-based authentication howto. URL http://www.linux.org/docs/ldp/howto/8021X-HOWTO/intro.html.

MAAWWG. Email metrics report first and second quarters 2009, 2009. URL http://www.maawg.org/sites/maawg/files/news/MAAWG_2009-Q1Q2_Metrics_Report_11.pdf.

José Manuel Macías Luna. Actualidad movilidad, November 2009. URL http://wiki.rediris.es/eduroam/upload/Actualidad_eduroam-Santiago09.pdf.

Jouni Malinen. Linux wpa/wpa2/ieee 802.1x supplicant. URL http://hostap.epitest.fi/wpa_supplicant/.

Microsoft. Kim cameron: Distinguished engineer, 2009. URL http://www.microsoft.com/presspass/exec/de/Cameron/default.mspx.

Microsoft. Microsoft's vision for an identity metasystem. URL http://msdn.microsoft.com/en-us/library/ms996422.aspx.

Microsoft Corporation. [ms-peap]: Protected extensible authentication protocol (peap) specification, March 2010. URL http://download.microsoft.com/download/a/e/6/ae6e4142-aa58-45c6-8dcf-a657e5900cd3/%5BMS-PEAP%5D.pdf.

Miroslav Milinovic. Results of the eduroam supplicant survey, January 2010. URL http://www.terena.org/mail-archives//mobility/pdfCog1yl7EKw.pdf.

Samuel Muñoz Hidalgo. Simplesamlphp infocard module usage (information cards module for simplesamlphp), February 2009. URL http://rnd.feide.no/content/simplesamlphp-infocard-module-usage.

Arun Nanda and Michael B. Jones. Identity selector interoperability profile v1.5. Technical report, Microsoft Corporation, July 2008. URL http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5.pdf.

Pablo Navas Cañete. *Sistema de autenticación de redes inalámbricas basada en criptografía de clave pública*. PhD thesis, Universidad de Alcalá de Henares, 2006.

Identity Metasystem Interoperability Version 1.0. OASIS, July 2009. URL http://docs.oasis-open.org/imi/identity/v1.0/identity.html.

OASIS. Oasis security services (saml) tc, a. URL http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

OASIS. Oasis saml wiki, b. URL http://wiki.oasis-open.org/security.

OASIS. Security assertion markup language (saml) v2.0 technical overview, c. URL http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf.

WS-SecurityPolicy 1.2. OASIS, July 2007a. URL http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/.

WS-Trust 1.3. OASIS, March 2007b. URL http://docs.oasis-open.org/ws-sx/ws-trust/200512.

Web Services Security Username Token Profile 1.1. OASIS, February 2006a. URL http://docs.oasis-open.org/wss/v1.1.

Web Services Security X.509 Certificate Token profile 1.1. OASIS, February 2006b. URL http://docs.oasis-open.org/wss/v1.1.

Open1X. Ieee 802.1x open source implementation. URL http://open1x.sourceforge.net/.

Andy Powell and David Recordon. Openid: Decentralised single sign-on for the web. URL http://www.ariadne.ac.uk/issue51/powell-recordon/.

RedIRIS. Ptyoc - validación de componentes para la infraestructura de autenticación y autorización edugain, a. URL http://www.rediris.es/ptyoc/res/dl03.html.es.

RedIRIS. Eduroam rediris, b. URL http://www.eduroam.es/index.es.php.

RedIRIS. Política de movilidad de la iniciativa eduroam es, c. URL http://www.eduroam.es/politica.es.php.

Michael B. Scher. Sso/rso/uso/ oh no?, December 2004. URL http://www.usenix.org/publications/login/2004-12/pdfs/ohno.pdf.

Randal L. Schwartz and Tom Phoenix. *Learning Perl*. O'REILLY, 3 edition, 2001. ISBN 0-596-00132-0.

Sentillion. New york university medical center selects sentillion for improved access and security for its cpoe initiative, January 2005. URL http://www.sentillion.com/media/press/050125.html.

Andreas Åkre Solberg. simplesamlphp advanced features, February 2008a. URL http://rnd.feide.no/content/simplesamlphp-advanced-features.

Andreas Åkre Solberg. simplesamlphp maintenance and configuration, February 2008b. URL http://rnd.feide.no/content/simplesamlphp-maintenance-and-configuration.

Andreas Åkre Solberg. Using simplesamlphp as an identity provider, October 2007a. URL http://rnd.feide.no/content/using-simplesamlphp-identity-provider.

Andreas Åkre Solberg. Installing simplesamlphp, October 2007b. URL http://rnd.feide.no/content/installing-simplesamlphp.

Andreas Åkre Solberg. simplesamlphp modules, February 2008c. URL http://rnd.feide.no/content/simplesamlphp-modules.

Andreas Åkre Solberg. Using simplesamlphp as a service provider, October 2007c. URL http://rnd.feide.no/content/using-simplesamlphp-service-provider.

Andreas Åkre Solberg. Theming simplesamlphps, January 2009. URL http://rnd.feide.no/content/theming-simplesamlphp.

Andrew S. Tanenbaum. *Redes de computadoras*. Pearson Educación, 4 edition, 2003. URL http://books.google.es/books?id=WWD-4oF9hjEC&printsec=frontcover#v=onepage&q=&f=false.

TERENA. Terena official site. URL http://www.terena.org/.

Universidad de Murcia. Dame project. URL http://dame.inf.um.es/.

Universidad de Sevilla. Ssid eduroam. configuración con open1x/xsupplicant. URL http://reinus.us.es/ssid-eduroam-open1x.es.html.

W3C. Canonical xml version 1.0, a. URL http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

W3C. Guía breve de servicios web, b. URL http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

Nigel Watling and Andy Harjanto. Infocard_explained_final, April 2006. URL http://mschnlnine.vo.llnwd.net/d1/ch9/0/8/0/1/8/1/InfoCard_Explained_Final.wmv.

Wikipedia. Ieee 802.1x, a. URL http://en.wikipedia.org/wiki/802.1X#Vulnerabilities.

Wikipedia. Digital identity, b. URL http://en.wikipedia.org/wiki/Digital_identity.

Wikipedia. Enterprise single sign-on, c. URL http://en.wikipedia.org/wiki/Single_sign-on#Enterprise_single_sign-on.

Wikipedia. Estándar de facto, d. URL http://es.wikipedia.org/wiki/Estandar_de_facto.

Wikipedia. Identity metasystem, e. URL http://en.wikipedia.org/wiki/Identity_Metasystem.

Wikipedia. Infocard, f. URL http://en.wikipedia.org/wiki/Infocard.

Wikipedia. Ingeniería social, g. URL http://es.wikipedia.org/wiki/Ingenieria_social_(seguridad_informatica).

Wikipedia. Openid, h. URL http://en.wikipedia.org/wiki/Openid#Security_and_phishing.

Wikipedia. Password fatigue, i. URL http://en.wikipedia.org/wiki/Password_fatigue.

Wikipedia. Phishing, j. URL http://es.wikipedia.org/wiki/Phishing.

Wikipedia. Single sign-on, k. URL http://en.wikipedia.org/wiki/Single_sign-on.

Wikipedia. Tipo abstracto de datos, l. URL http://es.wikipedia.org/wiki/Tipo_abstracto_de_datos.

Wikipedia. Xml signature, m. URL http://en.wikipedia.org/wiki/XML_Signature.

Wikipedia. Xml schema, n. URL http://es.wikipedia.org/wiki/XML_Schema.

Wikipedia. Acoplamiento, o. URL http://es.wikipedia.org/wiki/Dise%C3%B1o_estructurado#Acoplamiento.

Wikipedia. Eduroam, p. URL http://en.wikipedia.org/wiki/Eduroam.

Wikipedia. Roaming/itinerancia, q. URL http://es.wikipedia.org/wiki/Roaming.

Tomasz Wolniewicz and Maja Górecka-Wolniewicz. Identity management of users in eduroam. URL http://tnc2009.terena.org/core/getfile2527.ppt?file_id=325.